Received: 20 February 2023 | Revised: 30 October 2023 | Accepted: 3 November 2023

DOI: 10.1111/exsy.13505

ORIGINAL ARTICLE

Expert Systems WILEY

Artificial Intelligence Web Application Firewall for advanced detection of web injection attacks

Jesús-Ángel Román-Gallego | **María-Luisa Pérez-Delgado** | **Marcos Luengo Viñuela** | **María-Concepción Vega-Hernández**

Escuela Politécnica Superior de Zamora, Universidad de Salamanca, Zamora, Spain

Correspondence
Jesús-Ángel Román-Gallego, Escuela Politécnica Superior de Zamora, Universidad de Salamanca, Avenida de Requejo 33, C.P. 49022, Zamora, Spain.
Email: zjarg@usal.es

Funding information
Samuel Solórzano Barruso Memorial Foundation of the University of Salamanca, Grant/Award Number: FS/102015

Abstract

Currently, web services-based applications have an important presence in public and private organizations. The vulnerabilities that these types of applications may have pose an inherent potential risk to the business model of these organizations. These applications have the inherent risk of being used by organizations in such a way that their activity is affected and they become the main entry point for attackers who want to breach their security. The main barrier to this type of attack are web application firewalls (WAF), which are responsible for processing Hypertext Transfer Protocol requests between clients and web servers, classifying them and rejecting malicious requests. This type of (WAF) applications, for the most part, have regular expressions that correspond to general rules and allow detecting malicious requests that follow a pattern contained in them. However, due to the knowledge of these rules by attackers, it is easy to circumvent security and to impersonate a malicious request by an innocuous request. Therefore, in this article, we present a study of different models based on artificial intelligence techniques as Naïve Bayes, k-nearest neighbors, support vector machines, and linear regression to test their effectiveness in detecting malicious requests from a synthetic dataset containing more than 100,000 requests. The results obtained show that the implementation of these methods optimize the detection of malicious requests obtaining results between 92% and 99% of success in their classification.

KEYWORDS
artificial intelligence, injection, machine learning, vulnerability, web application firewall

1 | INTRODUCTION

Currently, the interest in cybersecurity and the importance of software vulnerability analysis is present in all areas of our lives due to the rapid growth of information and communication technologies. Regarding the vulnerability analysis of web application services, according to the report of an external pentesting of the company Ptsecurity of the year 2020, web vulnerabilities account today for a percentage of around 77% (Ptsecurity, 2020) as shows in Figure 1. Regarding the Acunetix report entitled 'Web Application Vulnerability 2019' shows that 46% of web

Abbreviations: HTTP, Hypertext Transfer Protocol; WAF, web application firewall.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2023 The Authors. *Expert Systems* published by John Wiley & Sons Ltd.

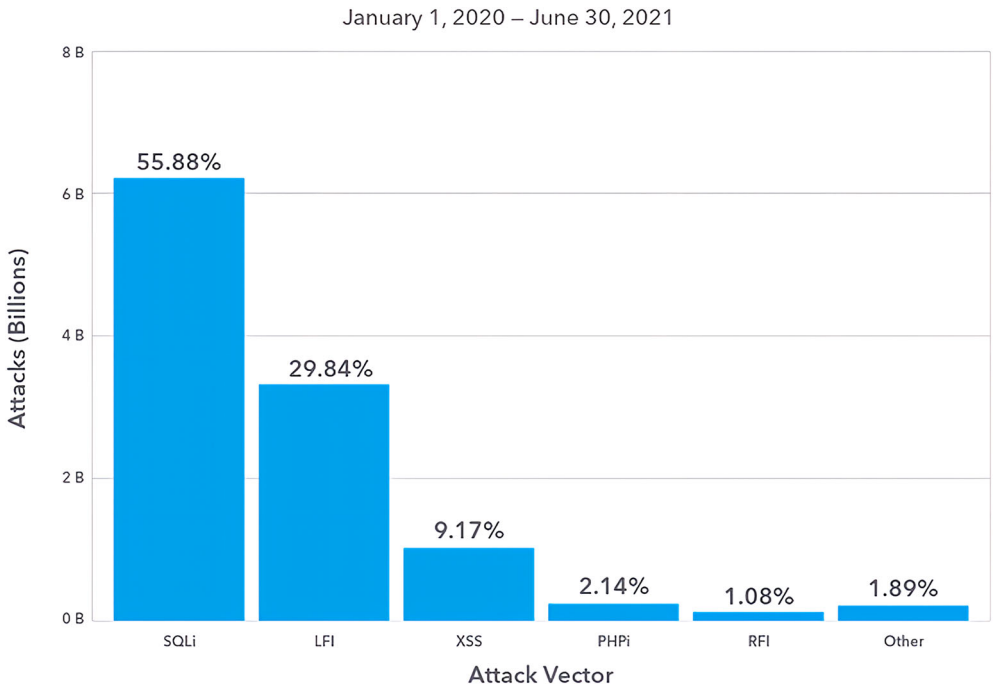

FIGURE 1 Main attack vectors between 2020 and 2021 (Ptsecurity, 2020).

applications have critical vulnerabilities, 30% of web applications are vulnerable to cross site scripting (XSS) and 51% of applications are susceptible to cross site request forgery (CSRF) (Acunetix, 2020). Another Akamai report shows that Structured Query Language (SQL) injection attacks, along with Local File Inclusion (LFI) and XSS attacks accounted for 89.8% of web attacks on the application layer during a 17-month period from 2018 to 2020 (Akamai, 2021). These data show the latent need to implement greater security between web servers and clients, and reflects the obsolete use of web application firewalls (WAFs) by a large number of companies; either due to ignorance of the recommended configuration or due to the limitations of the WAF itself, today they do not have the necessary protection.

The main limitation of conventional WAFs is that they base their detection either on the use of regular expressions in the request to the web server or on query blacklist filtering. First of all, this lacks flexibility in the implementation of the WAF, since rules must vary for each type of application. Second, knowing the rules allows the creation of custom queries that bypass these rules. Third, they do not allow for easy scalability in protection, as creating new rules can sometimes lead to false positive issues and do not provide effective protection.

The last issue to point out is the need for a person to correctly design, update and configure the detection system, as there can be problems associated with human error in this task.

Therefore, based on this real need, this work proposes the design and implementation of a WAF that tries to solve the previous problems. To achieve this goal, artificial intelligence (AI) is presented as the ideal tool in the generation of a model that allows for a high effectiveness in the success rate, great scalability by increasing the source of knowledge and great flexibility due to its ability to generalize possible malicious queries.

Makiou et al. (2014) focus on the problem of detecting complex SQL injections, for which they propose a novel approach to dissect Hypertext Transfer Protocol (HTTP) requests to cover most of the evasion techniques and improve the security rule management process. The proposed system is based on a Bayesian classifier, obtaining results with 97.6% accuracy. On the other hand, Ito and Iyatomi (2018) proposes the implementation of a WAF based on fast and accurate convolutional neural networks which analyzes HTTP requests for SQL injections and buffer overflows, among others, obtaining an accuracy of 98.2%. Betarte et al. (2018) apply machine learning techniques to improve the detection capabilities of WAF Mod Security, giving special importance to the task of reducing the false positives generated. The algorithms evaluated in this article are k-nearest neighbors-3 (KNN), support vector machines (SVM), and random forest (RF) (Breiman, 2001), and results between 90% and 100% accuracy are obtained.

The proposal of ISiker and SoGukpinar (2021) is a machine learning based WAF web attack detection method that uses the n-gram and TFIDF character model (Piskorski & Jacquet, 2020) to obtain feature representations of HTTP requests. The experiments carried out yield results greater than 95% accuracy.

This article addresses possible ways of processing predictive model requests and their characterization for further processing. This requires a review of the possible AI techniques to be used, their implementation and a comparison that allows us to select the model that provides optimal results. Regarding the training set, it is necessary to have enough data to obtain the final model, so it will be necessary to generate these data using an algorithm that allows them to be obtained within the established deadlines.

Once the data for training has been determined, the full development of the WAF models has been tackled, allowing a malicious request to be detected as a result of the processing of a text input that would simulate the content of any field of the HTTP request. A system has been

developed to perform the corresponding tests and determine which is the best option, generating a comparative document with the results obtained that contains the accuracy, completeness and confusion matrix of the different models proposed.

The results obtained in our research have exceeded expectations in the classification of malicious injections in HTTP requests, obtaining percentages high, which indicates that the method proposed in this study is completely feasible for its implementation and deployment in a reverse proxy, an objective that will be addressed as a future line of work.

The rest of the article is organized as follows. Section 2 includes a literature review related to WAFs. Section 3 describes the fundamental theoretical concepts of this work. Section 4 discusses the machine learning techniques used in this work. The experimental results are discussed in Section 5 and, finally, Section 6 shows the conclusions of the article. As an appendix, the Table A1 containing the different abbreviations that appear in this article.

2 | LITERATURE REVIEW

The evolution of threats from the Internet is constant, so the devices that act to try to stop them must evolve in parallel. This section provides a literature review on the evolution of WAFs and detection techniques used to prevent threats.

In Desmet et al. (2006) indicates web applications are the gateway for many attacks in an ICT (Information and Communications Technology) infrastructure, so WAFs play an important role in preventing the exploitation of vulnerabilities in web applications. In this sense, the main contribution of this work is that it shows how, through a combination of static and dynamic verification, WAFs can formally guarantee the absence of certain types of erroneous behaviour in web applications from a prototype of their solution based on a static verification tool.

On the other hand, a security solution using artificial neural networks to protect web applications against SQL injection attacks is proposed in Moosa (2010). This proposal has been tested on sample datasets and has yielded promising results in both accuracy and processing time to distinguish between normal and malicious content, based on a keyword-based approach.

Tekerek and Bay (2019) propose a hybrid learning-based WAF model to prevent web-based attacks using signature-based detection to detect known web attacks and anomaly-based detection to detect anomalous HTTP requests. The second type of detection is done using artificial neural networks. The results presented in the article show that the proposed solution obtained a better performance than the one offered by the existing studies so far (96.59%).

A model that uses that uses machine learning and feature engineering to detect common web attacks was proposed in Shaheed and Kurdy (2022). The model extracts a set of features from incoming requests to the web server: the length of the request, the percentage of allowed characters, the percentage of special characters and the weight of the attack. Such features fully describe the parts of the HTTP request (URL, payload, and headers) and are used to decide whether the request is normal or not. The authors used classification algorithms that have proven to be efficient in binary classification problems, such as logistic regression, decision tree and Naïve Bayes (NB), focusing on the last one.

For this purpose, a character-level convolutional neural network (CLCNN) with a very large maximum global pooling is used to extract the features of HTTP requests and identify them as normal or malicious (Ito & Iyatomi, 2018). The results of the article indicate that the model achieved a classification accuracy of 99.6% with datasets used in other studies, this value being slightly lower for datasets from real web servers. The results obtained on the DATASET CSIC 2010 (Giménez et al., 2010) dataset yield an accuracy of 98.8% with 10-fold cross-validation. However, in this work the malicious requests did not include path-traversal or command injection attacks, among others.

The research described in Işiker and SoĞukpinar (2021) proposes an anomaly-based WAF model that uses natural language processing techniques and linear SVM learning algorithm. The n-word-gram and n-character-gram natural language processing techniques were compared using independent models. SVM, LR, RF, KNN, and decision trees were used to make a comparison among different techniques, with SVM and KNN being the best performing algorithms with an efficiency greater than 99.5%.

Taking into account the work presented by other authors, and seeing which techniques and algorithms have proven to be more efficient and with greater generalization capacity, this work seeks to optimize the result of different techniques based on AI from new data and thus be able to have greater accuracy and greater generalization capacity on this type of threats, so that its implementation in a WAF allows obtaining high performance from.

3 | CONCEPTUAL THEORETICAL FRAMEWORK

HTTP is a transaction-oriented protocol that can be used in any client-server application that includes hypertext, and on which the World Wide Web (WWW) is based. This protocol is an application-level protocol for collaborative, hypermedia, and distributed information systems. It is a generic, stateless, protocol which can be used for the transmission of information with the necessary efficiency to transfer data which can be native text, hypertext, audio, images or any information accessible through the Internet, being its current standard HTTP 1.1 (RFC 2616) (Fielding et al., 1999). The usual form of transmission via HTTP is between a browser and a web server. In this case, to provide reliability, HTTP makes use

of the TCP protocol (Forouzan, 2002) where each transaction is treated independently. The communication between client and server using the HTTP protocol is done through messages, which can be of two types: requests from clients to servers and responses from servers to clients (Fielding et al., 1999). The general structure of the messages is as follows:

- **Request or start line:** identifies the requested resource and the type of message.
- **Response or status line:** provides information on the status of the response.
- **General header:** contains certain fields applicable to the request and response messages, but have no application to the transferred entity.
- **Request header:** provides information about the request and the client.
- **Response header:** contains information about the response.
- **Entity header:** contains information about the resource identified by the request and about the body of the entity.
- **Entity body:** corresponds to the message body.

Based on the previous scheme, and taking into account the information included in Mozilla (2023), the content of the structure of an HTTP message is detailed below, request and response cases are considered:

HTTP requests:

- **Start line:** it is made up of three elements. The first is the HTTP method (GET, PUSH, POST, ...) describing the action to be performed. The second corresponds to the purpose of the request and includes a URL, or the complete address of the protocol, port, and domain. Finally, the third is the HTTP version that defines the structure of the messages and the version expected for the response.
- **Headers:** they can be classified into three groups.
  - General headers, such as via (En-US), which affect the message as a complete unit.
  - Request headers, such as User-Agent or Accept-Type, which modify the request by specifying it in more detail (such as Accept-Language [En-US]).
  - Entity headers, such as Content-Length, which apply to the body of the request and do not need to be transmitted if the message has no body.
- **Body:** it is not mandatory, so it does not need to appear in the request. When it appears, it is divided into two categories.
  - Single data, which is a single file defined in the headers: Content-Type and Content-Length.
  - Request headers, such as User-Agent or Accept-Type, which modify the request by specifying it in more detail (such as Accept-Language [En-US]).
  - Multiple data, which are made up of different contents, are usually associated with Hypertext Markup Language (HTML) forms.

HTTP responses:

- **Status line:** consists of three elements. The first is the HTTP protocol version (e.g., HTTP/1.1) describing the action to be performed. The second corresponds to the status code, which indicates the success or failure of the request (e.g., 200, 404 or 302) and, finally, the third is the status text, which is a brief description, in text, for information purposes, of what the status code means (e.g., HTTP/1.1 404 Not Found).
- **Headers:** they have the same structure as in HTTP requests.
- **Body:** this block does not appear in all. If they do exist, those with a status code such as 201 or 204 (En-US) usually do without it. They are divided into three categories.
  - Single data, which is a single file defined in the headers: Content-Type and Content-Length.
  - Single split data, is a single file of unknown length, and encoded in parts (indicated by the transfer-encoding chunked value).
  - Multiple data, consists of several data, each with a different section of information.

On the other hand, a client-server architecture is the model that separates the tasks between clients and servers that have to communicate through a computer network. In this model, the client's role is to send a request to another programme to access a service offered by the server. The relationship between client and server corresponds to the request-response pattern and must adhere to a common set of protocols that defines the language, rules or dialog patterns used (Mozilla, 2022). Depending on the type of web service architecture, there can be static or dynamic web services. A static web service always returns a content set on the server. In this case when the server receives a (*HTTP GET*) request, it returns the requested information in a response with a status code. On the other hand, a dynamic web service allows content to be generated and returned based on the URL and specific details of the request.

According to Figure 2, the architecture works as follows:

1. The web browser generates an HTTP GET request to the server from the resource URL.

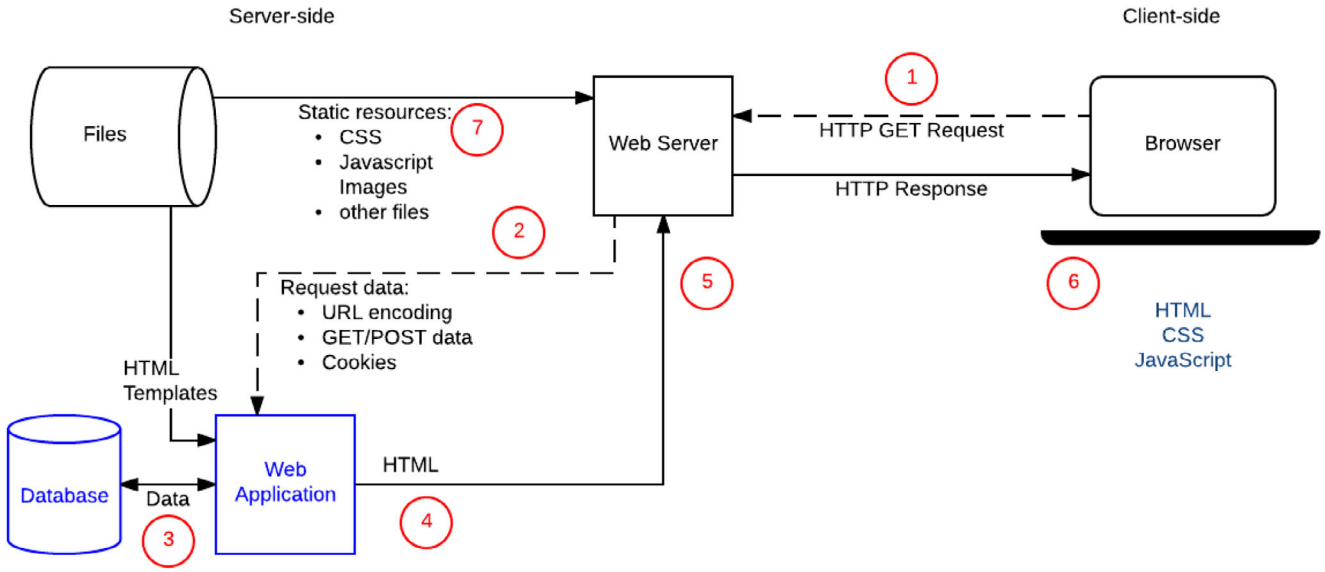

FIGURE 2 Dynamic web server request architecture schematic (Mozilla, 2022).

2. The web server detects the type of request, in this case dynamic, and it is sent to the web application.
3. The web application identifies the client's request and finds the required resource.
4. The web application dynamically creates the HTML code with the required data.
5. The web server returns the generated HTML code to the browser, together with an HTTP status code with value 200 ('success'); otherwise, it will return the corresponding error code.
6. The web browser will process the returned HTML code and send separate requests for style sheets or other files.
7. The web server retrieves the static files and returns them directly to the browser.

In view of the above, HTTP request injections are a type of web application security vulnerability that occurs when HTTP request content or headers are generated dynamically based on user input (Chandramouli et al., 2018). SQL injections consist of the insertion of a text string in the input data from the client to the application, which makes it possible to execute malicious SQL queries (Clarke, 2009; Halfond et al., 2006). The most representative types of SQL injections are listed below:

- **Union-based injection:** in this case, the main idea is to expand the results of the original query by using the UNION SQL operator, which allows you to combine two SELECT statements into a single result that is returned as part of the response.
- **Blind injection:** the attack consists of performing SQL queries that return binary values and determining the response based on these values.
- **Boolean-based injection:** this is performed by queries towards authentication services that trick the database into assuming they have high-level permissions or correct credentials.
- **Error-based injection:** this technique is based in the use of error messages to return the full query results and show sensitive database information.
- **Time-based injection:** this consists of inserting commands that execute operations that take a long time to complete, often many seconds.

Another type of injection is XSS, where malicious scripts are inserted into trusted sites. This type of attack starts from the browser to the end user, where the attacker uses the web application to send malicious code (Fonseca et al., 2007). The most representative XSS injections are listed below:

- Reflected XSS: the application uses non-validated user-supplied data encoded as part of the output HTML or JavaScript.
- Stored XSS: the application stores unvalidated and unsanitized user-supplied data, which are later viewed by another user or an administrator.
- DOM-based XSS: the application processes user-controllable data insecurely.

The aim of command injections is the execution of commands on a host operating system from vulnerable applications. These cases occur when an application sends unsecured user-supplied data (e.g., forms, cookies, http headers, etc.) to the system shell (OWASP, 2022a).

Finally, Path Traversal injections aim to access files and directories that are stored outside the root where the web application is located, by manipulating files and variables with sequences (../), other variants or their absolute paths. In this way, it is possible to access information stored in the file system that includes application codes or critical system configurations (OWASP, 2022b).

As WAF providers try to maintain up-to-date regular expression lists, new ways are found to circumvent this protection by masking the payloads of the malicious instruction, leaving the service vulnerable. In order to analyze common patterns in malicious instruction obfuscations, the main existing methods will be presented below (Demetrio et al., 2020):

- Case-sensitivity method: mixing uppercase and lowercase characters sometimes allows established rules to be circumvented.
- URL and Unicode encoding method: it consists in encoding the content of the request so that the excluded characters are not recognized.
- Junk character or comment inclusion method: the addition of junk characters can make it difficult for the WAF to recognize them.

Figure 3 shows a representation of the proposed architecture and shows in red the AI-WAF where the different models based on the AI techniques proposed in this article are implemented.

4 | MACHINE LEARNING TECHNIQUES

In the present work, different supervised classification techniques have been used to make a comparison and generate a model that allows obtaining optimal results in terms of accuracy and performance. This section presents the different models proposed for this work and their mathematical conceptualization.

4.1 | Naïve Bayes (NB)

NB classification algorithms (Saritas & Yasar, 2019; Ting et al., 2011) correspond to a set of supervised techniques based on the application of NB theorem with the naive assumption of independence between each pair of features. Given a class y together with a vector of dependent features $x = (x_1,...,x_n)$, Bayes' theorem establishes the relationship indicated in Equation 1, where P represents conditional probability.

$$P(y|x_1,...,x_n) = \frac{P(y)P(x_1,...,x_n|y)}{P(x_1,...,x_n)}. \quad (1)$$

Taking into account the naive independence assumption (Equation 2), for all i, the relationship described in Equation (1) can be expressed in a simplified way by Equation (3).

$$P(x_i|y,x_1,...,x_{i-1},x_{i+1},...,x_n) = P(x_i|y), \quad (2)$$

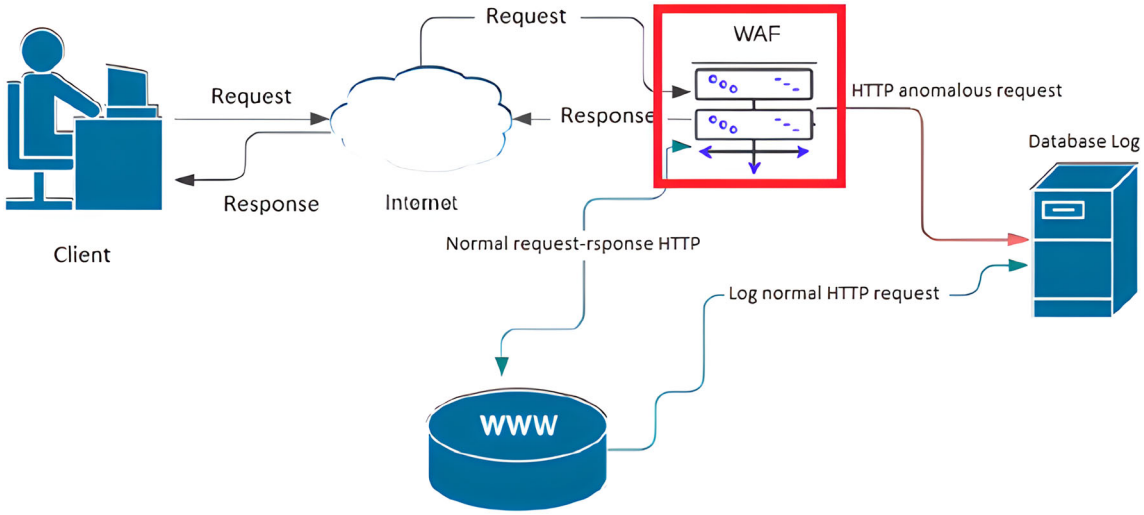

FIGURE 3 Architecture of AI-WAF.

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,...,x_n)}. \tag{3}$$

Since $P(x_1,...,x_n)$ is constant given the input, the following classification rule can be used. $P(x_1,...,x_n)$ is constant given the input sorting rule as shows in (4), and a maximum estimation can be used later to estimate $P(y)$ and $P(x_i - y)$ where the probability $P(y)$ is the relative frequency of the class y in the training set and the probabilities $P(x_1...x_n - y)$ are calculated differently in each version of NB.

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \Rightarrow \widehat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i|y), \tag{4}$$

NB multinomial (Abbas et al., 2019) applies for multinomial distributed data, and is one of the two classical variants of Naïve Bayes used in text classification.

The distribution is parameterized by vectors $\theta_y = (\theta_{y1}, ..., \theta_{yn})$ for each class **y**, where *n* is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i|y)$ that a feature **i** appears in a sample belonging to the class **y**.

The θ_y parameters are estimated using a maximum likelihood smoothed version, that is, the relative frequency count given by Equation (5), where

N_{yi} is the number of times **i** appears in the training set class sample **T** (Equation 6) and N_y is the total count of all features in the class **y** (Equation 7).

$$\widehat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}, \tag{5}$$

$$N_{yi} = \sum_{x \in T} x_i, \tag{6}$$

$$N_y = \sum_{i=1}^{n} N_{yi}. \tag{7}$$

The α parameter, with $\alpha \geq 0$, takes into account features that are not present in the training samples and avoids null probabilities in the subsequent calculations. Laplace smoothing is used when $\alpha = 1$, while Lidstone smoothing is used if $\alpha < 1$ (Chen & Goodman, 1999; Jauhiainen et al., 2017; Vollmer et al., 1999).

4.2 | K-nearest neighbors (KNN)

The k-nearest neighbors (KNN) algorithm (Kozma, 2008) is a supervised classification method used to estimate the density function $F(x|C_j)$ of the predictors *x* for each class C_j. This is a nonparametric classification method, which estimates the value of the probability density function that an element *x* belongs to the class $F(x|C_j)$ from the information provided by the dataset.

The training examples are vectors from a multidimensional feature space. Each example is described by *p* attributes considering *q* classes for classification.

The attribute values of the *i*th example (where $1 \leq i \leq n$) are represented by the following *p*-dimensional vector:

$$x_i = (x_{1i}, x_{2i}, ..., x_{pi}) \in X. \tag{8}$$

The space is divided into regions by locations and labels of the training examples. A point in the space is assigned to class *C* if this is the most frequent class among the *k* nearest training examples. For measurement among elements and classes, the Euclidean distance is generally used. Equation (9) computes the Euclidean distance between examples x_i and x_j.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}. \tag{9}$$

The algorithm is defined by the following two steps:

1. **Training algorithm:** consists of storing the characteristic vectors and class labels of the training examples. For each pair of values $\langle x, f(x)\rangle$ where $x \in X$ this operation adds the vector to the learning structure.
2. **Classification algorithm.** Let x_q be a value to be classified, x_1, ..., x_k its nearest neighbors, and V a spatial vector, $f(x_q)$. This operation returns $\widehat{f}(x_q)$, which is the most common value of $f(x_q)$ among the k nearest neighbors of x_q (given by Equation (10)).

$$\widehat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} [v = f(x_i)]. \tag{10}$$

Choice of k: The choice of parameter k depends mainly on the data, so large values of k reduce the effect of noise on classification, but create boundaries between similar classes. A good choice of parameter k can be made using optimization methods (Zouhal & Denoeux, 1998).

4.3 | Support vector machines (SVM)

SVMs (Noble, 2006; Pisner & Schnyer, 2020) are a set of supervised learning algorithms which given a set of training samples these methods can label classes and train these SVMs to build a model that fit the class of a new sample. SVM is conceptually based on the following: the input vectors are nonlinearly projected onto a very high-dimensional feature space, and a linear decision surface is parameterized on this feature space obtaining a high generalization capability. In this way, a SVM represents the points of a sample in the space, and separates the classes into two spaces as wide as possible, by means of a separation hyperplane which is defined by the two points of the two closest classes, and which is called support vector (Cortes & Vapnik, 1995).

For this, the main idea is to define a hyperplane or set of hyperplanes in a very high dimensional (or even infinite) space that can be used in classification or regression problems where a correct separation between classes will allow an optimal classification, the simplest way to perform the separation being by a straight line, a straight plane or an N-dimensional hyperplane.

The universes to study do not usually correspond to two-dimensional idyllic cases as in the previous example, but an SVM algorithm must deal with:

- More than two predictor variables.
- Non-linear separation curves.
- Cases where the datasets cannot be completely separated.
- Classifications in more than two categories.

Due to the computational limitations of linear learning machines, they cannot be used in most real-world applications. The Kernel function representation offers a solution to this problem by projecting the information to a higher dimensional feature space which increases the computational capacity of the linear learning machine. That is, we will map the input space X to a new feature space of higher dimensionality, called the Hilbert space, according to the conversion indicated by Equation (11).

$$\begin{aligned} &F = \{\phi(x) | x \in X\} \\ &x = \{x1, x2, ..., xn\} \rightarrow \phi(x) = \{\phi 1(x), \phi 2(x), ..., \phi n(x)\} \end{aligned} \tag{11}$$

The base kernel functions can be the linear function (12), the polynomial function (13), the rbf function (14) and the sigmoid function (15).

$$\langle x, x' \rangle, \tag{12}$$

$$(\gamma \langle x, x' \rangle + \tau)^d \text{ where } d \text{ represents the degree,} \tag{13}$$

$$\exp\left(-\gamma \|x - x'\|^2\right), \tag{14}$$

$$\tanh(\gamma \langle x, x' \rangle + \tau). \tag{15}$$

The parameters γ, r and d are referred to as kernel parameters.

The definition for the SVC (support vector classifier) sorting algorithm (Rueping, 2010) is as follows:

- Given a set of training vectors $x_i \in \mathbb{R}^p$, $i = 1,..., n$, belonging to two classes, and a vector $y \in \{1, -1\}^n$, the goal is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by the function $\text{sign}(w^T\phi(x) + b)$ is correct for most of the samples.
- SVC solves the following primal problem defined by (16):

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \gamma_i \rightarrow \textbf{Where } y_i\left(w^T\phi(x_i) + b\right) \geq 1 - \gamma_i \textbf{ and } \gamma_i \geq 0, i = 1,...,n, \tag{16}$$

The classifier aims to maximize the margin between classes (by minimizing $\|w\|^2 = w^T w$) while adding a penalty when a sample is misclassified or is in the wrong side of the decision boundary. Ideally, the value $y_i\left(w^T\phi(x_i) + b\right)$ would be greater than or equal to 1 for all samples, indicating perfect prediction. In general, the cases associated with a problem cannot always be separated perfectly with a hyperplane, so this allows some samples to be within a distance γ_i of their correct margin boundary. The penalty term C controls the strength of this penalty.

- The secondary problem that SVC solves is given by the minimization expression indicated in Equation (17), where e is the vector containing ones, Q is a positive semidefinite matrix of size $n \times n$ whose elements are of the form $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel.

$$\min_{\alpha_i} \frac{1}{2} \alpha_i^T Q \alpha_i - e^T \alpha_i \rightarrow \textbf{Where } y^T \alpha_i = 0, \tag{17}$$

The terms α_i are called dual coefficients and are bounded by C, $0 \leq \alpha_i \leq C, i = 1,...,n$. This dual representation highlights the fact that the training vectors are implicitly mapped into a higher dimensional space by the function ϕ.

- Once the optimization problem is solved, the output of the decision function for a given input sample x is obtained by Equation (18).

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b \tag{18}$$

- In the case that the input sample vector is large, the linear SVC classifier can scale almost linearly to millions of samples and/or features. The equivalence for solving the primal problem in the linear classifier is given by Equation (19).

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max\left(0, 1 - y_i\left(w^T\phi(x_i) + b\right)\right). \tag{19}$$

4.4 | Linear regression (LR)

Multinomial logistic regression (also known as softmax regression) generalizes the logistic regression method for multiclass problems, that is, problems with more than two possible discrete outcomes. Its objective is to predict the probabilities of the different possible outcomes of a categorical distribution as a dependent variable, given a set of independent variables (which can be real-valued, binary-valued or categorical-valued among others) (Du et al., 2004; Wang & Yong, 2008). Suppose we have N values, and each value has M labels and can belong to C classes. We have:

- A matrix X defined in $\mathbb{R}^{N \times M}$, where X_{ij} represents the value i labelled j.
- A matrix Y defined in $\mathbb{R}^N$, where Y_i represents the value i belonging to the class k.

The aim is to obtain the weight matrix W defined in $\mathbb{R}^{M\times C}$, where W_{jk} represents the weights associated with the label j and class k. Once W is computed, it is used to predict the class membership of any observation X.

If we know X and W (for example all the elements of W can be set initially to 0), the multiclass logistic regression procedure would be as follows:

1. Calculate the product of X and W, where $Z = -XW$.
2. Take the softmax for each row Z_i based on Equation (20).

$$Z_i : P_i = \text{softmax}(Z_i) = \frac{\exp(Z_i)}{\sum_{k=0}^{C} \exp(Z_{ik})}, \tag{20}$$

After completing this operation, the sum of each row P_i must equal 1.

3. Take the argmax (maximum index value) for each row Z_i and find the class with the highest probability.

Based on the above, the maximum likelihood function that allows a value to be inferred from the set of observations and calculated weights is given by Equation (21).

$$p(Y_i|X_i,W) = P_{i,k=Y_i} = \text{softmax}\left(Z_{i,k=Y_i}\right) = \frac{\exp\left(Z_{i,k=Y_i}\right)}{\sum_{k=0}^{C} \exp(Z_{ik})} = \frac{\exp\left(-X_i W_{k=Y_i}\right)}{\sum_{k=0}^{C} \exp(-X_i W_k)}. \tag{21}$$

- For a given observation, we know the class of this observation, which is Y_i.
- The probability function of Y_i given X_i and W is the probability of the observation i and the class $k = Y_i$, which is the softmax of Z_i, $k = Y_i$.
- The probability function of Y given X and W is the product of all observations and is calculated by Equation (22).

$$p(Y \mid X,W) = \prod_{i=1}^{N} \frac{\exp\left(-X_i W_{k=Y_i}\right)}{\sum_{k=0}^{C} \exp(-X_i W_k)}. \tag{22}$$

5 | EXPERIMENTATION AND RESULTS

To carry out the experimentation with the techniques proposed in Section 4, we propose the creation of a dataset that includes different types of requests. The public datasets on xss injections, command injection, sqli and path-injection obtained from Shah and Kaggle (2021), Shah and Kaggle (2019), and Morzeux-github (2019) have been considered. With these data, a fusion has been performed to obtain the initial database that will be processed to balance the data and adapt them to the needs of the classification algorithms used in this research, Table 1 shows the distribution of the data.

The algorithm training tests have been performed using the Google Compute Engine backend of the Google Platform Colab with python as runtime environment and using a machine with the following specifications: Processor: Intel(R) Xeon(R) CPU @ 2.30 GHz, Ram Memory: 12.69 GB, SSD Disk: 78.19 GB.

When creating the merged dataset, the possibility that the input data have some type of obfuscation was considered to avoid being detected by the classification model as malicious. In our case, the data to be generated correspond to requests with obfuscation by URL character encoding or URL encoding and original data without obfuscation. The Algorithm 1 was used to perform the data processing proposed above.

TABLE 1 Distribution of input data according to the type of attack.

Label	Quantity
TOTAL	119,883
VALID	60,623
SQLI	26,306
XSS	19,460
COMMAND INJECTION	6922
PATH-TRAVERSAL	6572

5.1 | Vectorization methods

Term analysis is one of the main fields of application of machine learning algorithms. However, the raw data corresponds to a sequence of symbols that cannot be directly processed by the algorithms. This is because most of them expect numerical feature vectors with a fixed size instead of raw text documents with variable length.

To solve this problem, utilities are required to extract numeric features from text content:

- **Tokenization** of these strings and assignment of an integer identifier for each possible token.
- **Count** of token occurrences in each document.
- **Normalization and weighting** of the tokens that appear in most of the samples/documents.

In this vectorization scheme, features and samples are defined as follows:

- Each individual frequency of token occurrence (normalized or not) is treated as a **feature**.
- The vector of all token frequencies for a given document is treated as a **multivariate sample**.
- The set of documents is represented by a matrix with one row per document and one column for each token that appears in the vocabulary.

Algorithm 1 **Generation of obfuscated data.**

Input: *DataName* : filename of value-tag pairs in JSON format for loading from memory
Output: *DataOutput* : list of value-label pairs containing the total data
Begin
DataList: ⇐ ReadJSON(DataName, DataType)
ObfuscatedUrl: ⇐ DataList
ObfuscatedUrl ⇐ DeleteValuesValidType(ObfuscatedUrl)
ObfuscatedBase64 ⇐ Base64(ObfuscatedUrl)
for each index in *ObfuscatedUrl* **do**
$aux1 \leftarrow CodeUrl(ObfuscatedUrl[value][index])$
$ObfuscatedUrll[value][index] \leftarrow aux1$
end for
for each index in *ObfuscatedBase64* **do**
$aux2 \leftarrow CodeBase64(ObfuscatedBase64[value][index])$
$ObfuscatedBase64[value][index] \leftarrow aux2$
end for
$CompleteData \leftarrow ConcatenateList(DataList, ObfuscatedUrl, ObfuscatedBase64)$
$SaveJSON(CompleteDataList, OutputDataName)$
Return *CompleteData*

Based on this, vectorization can be defined as the general process of converting a collection of text documents into numeric feature vectors. This specific strategy (tokenization, counting and normalization) and the documents are described by word occurrences, without their relative position being a relevant factor in the document analysis. The two most relevant vectorization techniques are vectorization by counting scheme (Goyal, 2021) and vectorization by TFIDF weighting (Yamout & Lakkis, 2018).

5.2 | Results

The experimentation is carried out with the proposed algorithms on the data set indicated in Table 1. For each of the techniques it is intended to optimize their performance, so the two vectorizations proposed in Section 5.1 are applied, seeking a parameter adjustment that generates the best results for each of them. The evaluation is carried out using the following metrics: recall (Equation 23), precision (Equation 24), accuracy (Equation 25) and F1-score (Equation 26) (Sokolova et al., 2006).

The values used to compute the previous methods are as follows:

- FP (false positives): is the number of predicted positives that were incorrect. It represents the cases identified as an attack that do not really correspond to an attack.
- FN (false negatives): the cases identified as no attack that really correspond to an attack.
- TN (true negatives): the cases identified as no attack that really correspond to no attack.
- TP (true positives): is the number of predicted positives that were correct. It represents the cases identified as an attack that really correspond to an attack.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{23}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{24}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{25}$$

$$\text{F1-score} = 2\frac{\text{Precision Recall}}{\text{Precision} + \text{Recall}}. \tag{26}$$

The previous measures will provide the evaluation of each of the models, allowing their performance to be compared.

The total number of samples used in all models to obtain the results is 29,971, divided as follows:

- command injection: 1730 samples.
- path-traversal: 1643 samples.
- sqli: 6577 samples.
- valid: 15,156 samples.
- xss: 4865 samples.

5.2.1 | Results for LR model

Table 2 shows the results obtained with the LR model with vectorization by TFIDF. In this case, the parameters were set as follows: **ngram-range: (1, 4), max-iter: 300, solver: newton-cg**. When each of the dataset is considered separately, it is observed that results that exceed 90% are obtained in most cases. In addition, the model obtains an accuracy value of 98% when the total dataset is considered.

A confusion matrix allows us to visually check the degree of accuracy in classifying the data. Each column of the matrix indicates the prediction number for each class and each row represents the elements of the real class to which they belong.

Figure 4 shows the confusion matrix for the tests performed with the LR model using the following libraries (liblinear, newton-cg, lbfgs, sag and saga) to determine this model as the best LR model.

TABLE 2 Detailed LR results with vectorization by TFIDF.

Dataset	Precision	Recall	F1-score
command injection	0.97	0.85	0.91
path-traversal	0.94	0.94	0.94
sqli	0.99	0.98	0.99
valid	0.98	1.0	0.99
xss	1.0	0.99	0.99

5.2.2 | Results for Naïve Bayes multinomial model

The results of the NB multinomial model with vectorization by TFIDF are shown in Table 3. The best results of this method were obtained with the value of the parameter **ngram-range: (1, 4)**. In this case, the results obtained are worse than those generated by the LR model, obtaining an accuracy result of 89% for the total dataset. Figure 5 shows the confusion matrix for the NB multinomial model.

In this case, the best results obtained by this model correspond to the sqli, valid and xss datasets, with values exceeding 90% for the three error measures, while in the previous model (LR) the best results are obtained for the xss dataset. The worst results are obtained with the command injection dataset, not exceeding 60% in any case.

The 0 values that appear in the confusion matrix (Figure 5) indicate that the model has not classified any erroneous data at the intersection of the two datasets.

5.2.3 | Results for K-nearest neighbors model

Regarding the KNN classifier, the best results are obtained with the TFIDF vectorization and with the following parameters: **ngram-range: (1, 4), p:1, Dist-measure: Minkowski, n-neighbors = 5, algorithm: Brute-Force** (Table 4). In this case, the results of all the metrics exceed 92% for each dataset, and an accuracy of 99% is obtained for the global dataset. Figure 6 shows the confusion matrix for the KNN model.

5.2.4 | Results for support vector machines model

Finally, Table 5 shows the results obtained with SVM with rbf Kernel vectorized by TFIDF and with the following parameters: **ngram-range: (1, 4), kernel:rbf, C:10**. This model shows the best results of all models, obtaining in each of the datasets values greater than 97% and an accuracy of 99% in the complete dataset. The SVM tests have been performed with different Kernels (linear, rbf, polynomic and sigmoid), being the rbf kernel the one that obtains the best results. Figure 7 shows the confusion matrix for the SVM model with rbf kernel.

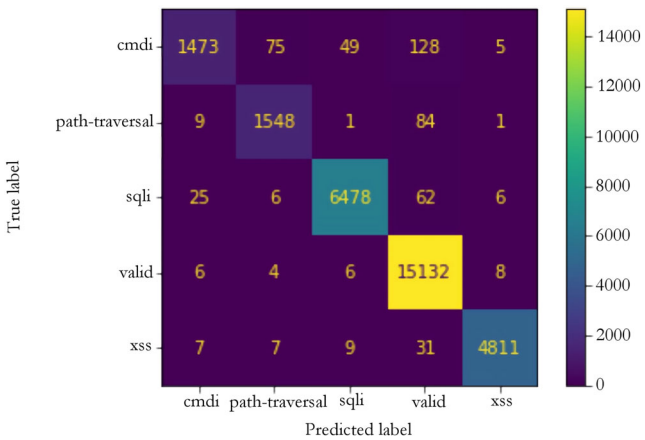

FIGURE 4 Confusion matrix for LR model.

TABLE 3 Detailed NB results with vectorization by TFIDF.

Dataset	Precision	Recall	F1-score
command injection	0.60	0.55	0.57
path-traversal	0.88	0.61	0.72
sqli	0.94	0.90	0.92
valid	0.88	0.98	0.93
xss	0.99	0.84	0.91

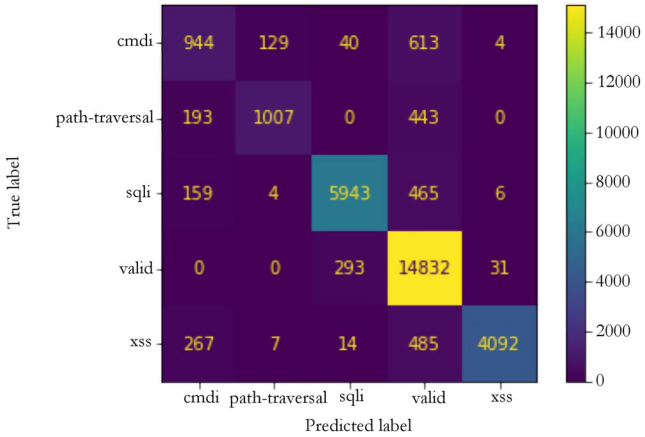

FIGURE 5 Confusion matrix for NB model.

TABLE 4 Detailed KNN results with vectorization by TFIDF.

Dataset	Precision	Recall	F1-score
command injection	0.97	0.92	0.94
path-traversal	0.98	0.96	0.97
sqli	1.0	0.98	0.99
valid	0.98	1.0	0.99
xss	1.0	0.99	1.0

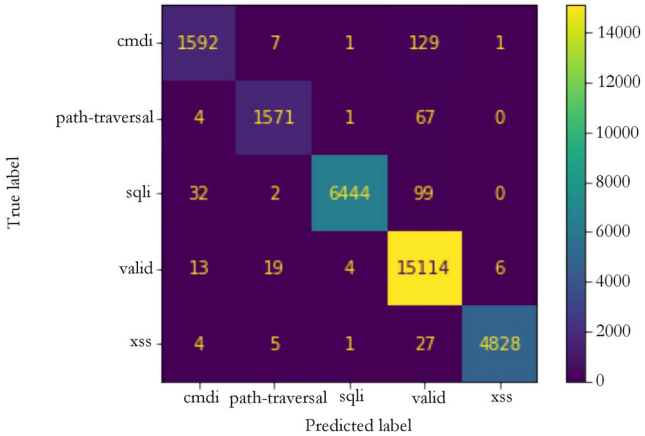

FIGURE 6 Confusion matrix for KNN model.

5.2.5 | Discussion

Considering the constant growth of cybersecurity threats and the importance of safeguarding information systems, this work aims to include AI-based models capable of differentiating an attack from a valid request and ultimately protect itself to the attack. For this purpose,

TABLE 5 Detailed SVM (rbf) results with vectorization by TFIDF.

Dataset	Precision	Recall	F1-score
command injection	0.97	0.98	0.97
path-traversal	0.99	0.98	0.98
sqli	1.0	0.99	1.0
valid	1.0	1.0	1.0
xss	1.0	1.0	1.0

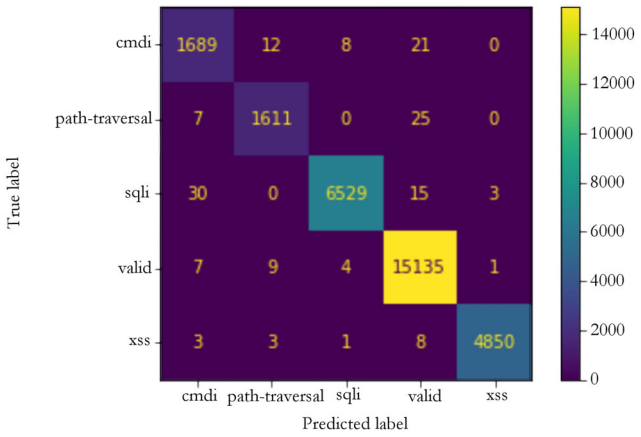

FIGURE 7 Confusion matrix for SVM (rbf) model.

different machine learning techniques such as linear regression, Naïve Bayes, KNN and SVM are introduced, from which an analysis and comparison will be made to obtain the best model for this type of application. To obtain a strong training and tests set, the public datasets on xss injections, command injection, sqli and path-injection obtained from Shah and Kaggle (2021), Shah and Kaggle (2019), and Morzeux-github (2019) have been considered, and the parameters of the different models have been selected so that the results of each of them are optimal. For the comparison of the different models, a series of measures such as recall, precision, accuracy and F1-score have been established, in order to have a wider view of the behaviour of the model on the different datasets individually, as well as globally. The results obtained by the models are very good, however, the SVM is very remarkable as it obtains the best results, both individually with each dataset and globally with all the data.

6 | CONCLUSIONS

Information technology security is a current and fundamental concept in all its aspects that has special relevance in the flow of information on the Internet. Given the growing interest in cybersecurity, the analysis of vulnerabilities in systems susceptible to compromise is a very important issue for any organization, whether public or private. The possibility of implementing AI techniques and their use in terms of filtering requests and being able to determine whether they are malicious makes it possible to prevent the consequences of attacks on current systems with success rates close to 100%. This means optimizing security even on modified data that would be much more difficult to detect by a human expert. In this work, machine learning methods have been applied to a large dataset that has been processed using vectorization and obfuscation techniques, so that the models obtained can optimize their performance. All the models used generate good results. However, the model that yields the best results is the SVM with rbf kernel since it exceeds 99% accuracy. That said, the research whose results are shown in this work has allowed comparing different techniques and determining an optimal model to be implemented in a real environment with an optimal perspective in terms of the results that can be obtained. Based on these results, it is proposed that taking into account the effectiveness of this model, it should be implemented in a WAF capable of filtering requests, so the next step in this work is the design and implementation of a reverse proxy that allows filtering and processing requests, generating usage data and possible attacks.

AUTHOR CONTRIBUTIONS

Conceptualization: Jesús-Ángel Román-Gallego, María-Luisa Pérez-Delgado, and Marcos Luengo Viñuela. *Methodology*: Jesús-Ángel Román-Gallego, María-Luisa Pérez-Delgado, and Marcos Luengo Viñuela. *Software*: Jesús-Ángel Román-Gallego and Marcos Luengo Viñuela. *Validation*:

María-Luisa Pérez-Delgado and María-Concepción Vega-Hernández. *Formal analysis*: María-Luisa Pérez-Delgado and María-Concepción Vega-Hernández. *Investigation*: Jesús-Ángel Román-Gallego and Marcos Luengo Viñuela. *Resources*: Jesús-Ángel Román-Gallego, María-Luisa Pérez-Delgado, and Marcos Luengo Viñuela. *Data curation*: Jesús-Ángel Román-Gallego, María-Luisa Pérez-Delgado, Marcos Luengo Viñuela, and María-Concepción Vega-Hernández. *Writing – original draft preparation*: Jesús-Ángel Román-Gallego and María-Luisa Pérez-Delgado. *Writing – review and editing*: Jesús-Ángel Román-Gallego and María-Luisa Pérez-Delgado. *Project administration*: María-Luisa Pérez-Delgado. *Funding acquisition*: María-Luisa Pérez-Delgado. All authors have read and agreed to the published version of the manuscript.

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available in Kaggle at https://www.kaggle.com/. These data were derived from the following resources available in the public domain: Sql injection dataset, https://www.kaggle.com/syedsaqlainhussain/sql-injection-dataset - Xss injection dataset, https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning - Http params dataset, https://github.com/Morzeux/HttpParamsDataset.

ORCID

Jesús-Ángel Román-Gallego https://orcid.org/0000-0002-2058-6219
María-Luisa Pérez-Delgado https://orcid.org/0000-0003-1810-0264
María-Concepción Vega-Hernández https://orcid.org/0000-0003-4266-4908

REFERENCES

Abbas, M., Memon, K. A., Jamali, A. A., Memon, S., & Ahmed, A. (2019). Multinomial Naïve Bayes classification model for sentiment analysis. *IJCSNS International Journal of Computer Science and Network Security*, *19*(3), 62–67.

Acunetix. (2020). Web application vulnerability report. https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2020/

Akamai. (2021). State of The Internet Security. https://www.akamai.com/resources/state-of-the-internet/soti-security-api-the-attack-surface-that-connects-us-all

Betarte, G., Pardo, A., & Martínez, R. (2018). Web application attacks detection using machine learning techniques. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1065–1072). IEEE.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Chandramouli, S. P., Bajan, P. M., Kruegel, C., Vigna, G., Zhao, Z., Doupe, A., & Ahn, G.-J. (2018). Measuring e-mail header injections on the world wide web. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (pp. 1647–1656). ACM.

Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, *13*(4), 359–394.

Clarke, J. (2009). *SQL injection attacks and defense*. Elsevier.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Demetrio, L., Valenza, A., Costa, G., & Lagorio, G. (2020). Waf-a-mole: evading web application firewalls through adversarial machine learning. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (pp. 1745–1752). ACM.

Desmet, L., Piessens, F., Joosen, W., & Verbaeten, P. (2006). Bridging the gap between web application firewalls and web applications. In *Proceedings of the 4th ACM Workshop on Formal Methods in Security* (pp. 67–77). ACM.

Du, W., Han, Y. S., & Chen, S. (2004). Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM International Conference on Data Mining* (pp. 222–233). SIAM.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext Transfer Protocol—HTTP/1.1. https://www.w3.org/Protocols/rfc2616/rfc2616.html

Fonseca, J., Vieira, M., & Madeira, H. (2007). Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)* (pp. 365–372). IEEE.

Forouzan, B. A. (2002). *TCP/IP protocol suite*. McGraw-Hill Higher Education.

Giménez, C. T., Villegas, A. P., & Marañón, G. Á. (2010). *HTTP dataset CSIC 2010* (64) [Data set]. Information Security Institute of CSIC (Spanish Research National Council).

Goyal, R. (2021). Evaluation of rule-based, CountVectorizer, and Word2Vec machine learning models for tweet analysis to improve disaster relief. In *2021 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 16–19). IEEE.

Halfond, W. G., Viegas, J., & Orso, A. (2006). A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering* (pp. 13–15). IEEE.

ISiker, B., & SoGukpinar, I. (2021). Machine learning based web application firewall. In *2021 2nd International Informatics and Software Engineering Conference (IISEC)* (pp. 1–6). IEEE.

Ito, M., & Iyatomi, H. (2018). Web application firewall using character-level convolutional neural network. In *2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA)* (pp. 103–106). IEEE.

Jauhiainen, T., Lindén, K., & Jauhiainen, H. (2017). Evaluation of language identification methods using 285 languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics* (pp. 183–191). ACL Antology.

Kozma, L. (2008). *k nearest neighbors algorithm (KNN)* (p. 32). Helsinki University of Technology.

Makiou, A., Begriche, Y., & Serhrouchni, A. (2014). Improving web application firewalls to detect advanced SQL injection attacks. In *2014 10th International Conference on Information Assurance and Security* (pp. 35–40). IEEE.
Moosa, A. (2010). Artificial neural network based web application firewall for SQL injection. *International Journal of Computer and Information Engineering*, *4*(4), 610–619.
Morzeux-github. (2019). Http params dataset. https://github.com/Morzeux/HttpParamsDataset
Mozilla. (2022). Client-server overview. https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview
Mozilla. (2023). HTTP messages. https://developer.mozilla.org/es/docs/Web/HTTP/Messages
Noble, W. S. (2006). What is a support vector machine? *Nature Biotechnology*, *24*(12), 1565–1567.
OWASP. (2022a). CMD injection (command injection). https://owasp.org/www-community/attacks/Command_Injection
OWASP. (2022b). CMD injection (path traversal). https://owasp.org/www-community/attacks/Path_Traversal
Piskorski, J., & Jacquet, G. (2020). TF-IDF character N-grams versus word embedding-based models for fine-grained event classification: A preliminary study. In *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020* (pp. 26–34). European Language Resources Association (ELRA).
Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In *Machine learning* (pp. 101–121). Elsevier.
Ptsecurity. (2020). External pentests results. https://www.ptsecurity.com/ww-en/analytics/external-pentests-results-2020/
Rueping, S. (2010). SVM classifier estimation from group probabilities. In *Proceedings of the 27 th International Conference on Machine Learning*. ICML.
Saritas, M. M., & Yasar, A. (2019). Performance analysis of ANN and Naïve Bayes classification algorithm for data classification. *International Journal of Intelligent Systems and Applications in Engineering*, *7*(2), 88–91.
Shah, S. S. H., & Kaggle. (2019). Xss injection dataset. https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning
Shah, S. S. H., & Kaggle. (2021). sql injection dataset. https://www.kaggle.com/syedsaqlainhussain/sql-injection-dataset/
Shaheed, A., & Kurdy, M. (2022). Web application firewall using machine learning and features engineering. *Security and Communication Networks*, *2022*, 5280158.
Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. In A. Sattar & B. Kang (Eds.), *Australasian Joint Conference on Artificial Intelligence* (pp. 1015–1021). Springer.
Tekerek, A., & Bay, O. (2019). Design and implementation of an artificial intelligence-based web application firewall model. *Neural Network World*, *29*, 189–206.
Ting, S., Ip, W., & Tsang, A. H. (2011). Is Naïve Bayes a good classifier for document classification. *International Journal of Software Engineering and Its Applications*, *5*(3), 37–46.
Vollmer, J., Mencl, R., & Mueller, H. (1999). Improved Laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, *18*, 131–138.
Wang, Z. Z., & Yong, J. H. (2008). Texture analysis and classification with linear regression model based on wavelet transform. *IEEE Transactions on Image Processing*, *17*(8), 1421–1430.
Yamout, F., & Lakkis, R. (2018). Improved TFIDF weighting techniques in document retrieval. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)* (pp. 69–73). IEEE.
Zouhal, L. M., & Denoeux, T. (1998). An evidence-theoretic K-NN rule with parameter optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *28*(2), 263–271.

AUTHOR BIOGRAPHIES

Jesús A. Román is an Associate Professor in the Department of Computer Science and Automatics at the University of Salamanca, Spain. He obtained his degree in Computer Science Engineering at the Pontifical University of Salamanca in 2006, MCS in Intelligent Systems in 2009 and PhD. in 2016 at the University of Salamanca. His research interests are in the areas of Intelligent Systems, Security Systems, Artificial Intelligence and Computer Science. He has published several research papers and book chapters related to these areas.

María-Luisa Pérez-Delgado is an associate professor in the Department of Computer Science and Automation at the University of Salamanca. She has a degree in Computer Science from the University of Salamanca and a degree in Computer Engineering from the University of Valladolid. She also holds a PhD in Mathematics from the University of Salamanca. She has participated in several regional, national and international research projects. She has also published several articles in scientific journals.

Marcos Luengo Viñuela is a part-time professor at the Escuela Politécnica Superior de Zamora. He holds a degree in Computer Engineering in Information Systems from the University of Salamanca and a Master in Cybersecurity Research from the University of León. His area of interest is Computer Security in all its fields, focusing on intrusion detection methods and DDoS attacks. He works in different national and international projects as a security consultant.

María Concepción Vega Hernández is a professor in the Department of Statistics at the University of Salamanca, where she received her PhD in Applied Multivariate Statistics from the University of Salamanca, and a Master in Advanced Multivariate Data Analysis. Her area of interest is applied multivariate statistics and her lines of research have been mainly linked to the areas of education and psychology, learning styles and strategies, information and communication technologies (ICT), and emotional intelligence. He has published research articles and book chapters related to these areas.

How to cite this article: Román-Gallego, J.-Á., Pérez-Delgado, M.-L., Viñuela, M. L., & Vega-Hernández, M.-C. (2024). Artificial Intelligence Web Application Firewall for advanced detection of web injection attacks. *Expert Systems*, e13505. https://doi.org/10.1111/exsy.13505

APPENDIX A

TABLE A1 Acronyms used in the article.

AI	Artificial intelligence
CSRF	Cross site request forgery
FP	False positives
FN	False negatives
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
KNN	K-nearest neighbors
LFI	Local file inclusion
LR	Linear regression
NB	Naïve Bayes
PHPI	php injection
RF	Random forest
RFI	Remote file inclusion
SQL	Structured query language
SQLI	Structured query language injection
SVC	Support vector classifier
SVM	Support vector machines
TCP	Transmission control protocol
TFIDF	Term frequency—inverse document frequency
TN	True negatives
TP	True positives
URL	Uniform resource locator
WAF	Web application firewall
WWW	World wide web
XSS	Cross site scripting