Detecting Zero-Day Web Attacks with an Ensemble of LSTM, GRU, and Stacked Autoencoders

Vahid Babaey [1] and Hamid Reza Faragardi[2]

1. Department of Electrical and Computer Engineering, University of North Carolina at Charlotte; vbabaey@charlotte.edu
2. Research Engineer, KTH Royal Institute of Technology, Stockholm, Sweden; Hamid.faragardi@trendplus.se
* Correspondence: vbabaey@charlotte.edu, Hamid.faragardi@trendplus.se

Abstract: The rapid growth in web-based services has significantly increased security risks related to user information, as web-based attacks become increasingly sophisticated and prevalent. Traditional security methods frequently struggle to detect previously unknown (zero-day) web attacks, putting sensitive user data at significant risk. Additionally, reducing human intervention in web security tasks can minimize errors and enhance reliability. This paper introduces an intelligent system designed to detect zero-day web attacks using a novel one-class ensemble method consisting of three distinct autoencoder architectures: LSTM autoencoder, GRU autoencoder, and stacked autoencoder. Our approach employs a novel tokenization strategy to convert normal web requests into structured numeric sequences, enabling the ensemble model to effectively identify anomalous activities by uniquely concatenating and compressing the latent representations from each autoencoder. The proposed method efficiently detects unknown web attacks while effectively addressing common limitations of previous methods, such as high memory consumption and excessive false positive rates. Extensive experimental evaluations demonstrate the superiority of our proposed ensemble, achieving remarkable detection metrics: 97.58% accuracy, 97.52% recall, 99.76% specificity, and 99.99% precision, with an exceptionally low false positive rate of 0.2%. These results underscore our method's significant potential in enhancing real-world web security through accurate and reliable detection of web-based attacks.

Keywords: Zero-Day Attacks, Tokenization, Autoencoder, Ensemble Classification, Neural Networks, LSTM, GRU, Stacked

1. Introduction

In modern digital infrastructure, websites and web-based applications play a crucial role in facilitating economic, educational, recreational, and political activities. However, as the reliance on these platforms increases, so does the risk of security threats, including unauthorized access, data breaches, and service disruptions. One of the primary attack vectors involves manipulating web requests, where adversaries masquerade as legitimate users to exploit vulnerabilities. Consequently, the detection and mitigation of malicious web requests have become vital for ensuring the security of any online service, including websites, web applications, and Content Delivery Networks (CDNs).

To counter such threats, various security mechanisms, including Web Application Firewalls (WAFs) and blacklisting techniques, have been deployed. While these methods offer some level of protection, they remain ineffective against zero-day attacks—novel exploits that lack predefined security signatures [1]. The primary challenge associated with zero-day attacks lies in their unpredictability, as they introduce previously unseen patterns that traditional rule-based detection systems fail to recognize. Addressing these challenges through deep learning-based anomaly detection presents a promising approach, leveraging neural networks to autonomously identify deviations indicative of malicious activity.

Conventional methods for preventing web-based attacks, such as WAFs [2] and blacklisting, exhibit several limitations. For instance, maintaining a blacklist of prohibited keywords within web requests is both time-consuming and insufficient in addressing

evolving attack patterns. Moreover, none of these existing approaches is capable of detecting zero-day attacks, as the strategies and obfuscation techniques employed in these attacks remain unknown. Recent advancements in machine learning and deep learning have demonstrated significant potential in enhancing security through intelligent threat identification, making these techniques highly relevant for modern cyber defense systems.

A critical advantage of anomaly detection models is that they do not require prior exposure to zero-day attacks to effectively detect them. In this study, an ensemble model is proposed that integrates multiple sub-models designed to detect zero-day attacks. Given that the patterns of zero-day attacks are inherently unknown, the model is trained exclusively on normal web request data. By learning the distribution of normal web traffic, the model becomes proficient in identifying deviations, thereby flagging both known and previously unseen attacks as anomalous. This approach ensures that malicious requests, whether originating from known attack types or zero-day exploits, are effectively classified as security threats.

To evaluate the proposed system, various web attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Buffer Overflow [3] are treated as zero-day attacks within the dataset. The model classifies any request with an anomaly score exceeding a predefined threshold as a potential zero-day attack. While the proposed approach does not explicitly categorize different types of attacks, it demonstrates the capability to reliably detect anomalous activities, ensuring a high level of security against emerging threats. The primary objective of this model is to simultaneously address both known and zero-day attacks while maintaining a high detection rate and minimizing false positives.

The rest of this paper is structured as follows: Section 2 presents the foundational concepts and research background. Section 3 provides a review of existing literature on web attack detection. The methodology and architectural design of the proposed model are discussed in Section 4, followed by a performance evaluation in Section 5. Section 6 elaborates on the broader implications of the findings, and Section 7 concludes the paper with final remarks.

The key contributions of this research are as follows:

- **Innovative Ensemble Model Architecture:** This study introduces a novel ensemble approach by integrating LSTM, GRU, and stacked autoencoders for anomaly detection in web requests. Unlike conventional ensemble methods that use simple averaging or majority voting, our approach uniquely concatenates and compresses the latent representations from each autoencoder. This technique significantly improves anomaly detection performance and computational efficiency.
- **Advanced Tokenization and Feature Mapping:** We propose a novel tokenization strategy that classifies tokens based on their character composition (numeric, lowercase, uppercase, and special characters). This structured approach effectively reduces input dimensionality, ensures greater consistency in data representation, and significantly enhances the detection capability of our anomaly detection system.
- **Zero-Day Attack Detection:** Our model is trained exclusively on normal web requests, enabling it to effectively identify and detect previously unseen zero-day attacks by capturing deviations from established normal request patterns.
- **Comprehensive Evaluation Metrics with Emphasis on False Positive Rate (FPR):** Unlike many existing studies, we explicitly evaluate and report the False Positive Rate, achieving a significantly lower FPR of 0.2%. This comprehensive evaluation underscores the practical applicability of our model, addressing an essential aspect often overlooked in anomaly detection research.

By addressing the limitations of traditional detection systems and leveraging anomaly detection through deep learning, this research contributes to advancing cybersecurity measures against evolving web-based threats.

2. Background

Due to the increasing reliance on internet-based services, web attacks pose significant threats to user privacy and can severely disrupt web server operations, affecting users on a global scale. Among these threats, zero-day (previously unknown) attacks are particularly concerning, as they can lead to privacy breaches and denial of service, impacting all users of a targeted website [4]. There are two approaches to detect attacks: non-ML (heuristic-based) and machine learning-based approaches. Heuristic methods play a crucial role in cybersecurity due to their ability to rapidly identify potentially malicious activities based on predefined rules and patterns, without requiring extensive labeled datasets. Heuristic algorithms analyze data patterns and system behaviors, identifying threats by matching observed behaviors against known suspicious patterns or predefined rules. The primary advantage of heuristic approaches lies in their capability to detect novel or zero-day threats promptly, often faster than traditional signature-based methods. However, heuristic approaches can suffer from high false positives or negatives due to their reliance on manually defined rules and parameters, and thus require continuous refinement and adaptation. Despite these limitations, heuristic algorithms remain widely adopted in cybersecurity for their interpretability, speed, and ability to quickly detect anomalies in real-time environments [5]. Machine learning-based approaches for web attack detection typically consist of two key phases: training and detection. During the training phase, the model learns from patterns of normal web requests, while in the detection phase, it utilizes this learned knowledge to identify and mitigate potential web attacks [6].

Web attack countermeasures can generally be categorized into three primary approaches: (1) supervised, (2) unsupervised, and (3) semi-supervised learning. The supervised approach is primarily designed to detect known attacks and is commonly implemented in signature-based systems such as Web Application Firewalls (WAFs). These models rely on labeled datasets containing both historical attack patterns and normal requests, making them highly effective against previously documented threats. However, their reliance on predefined signatures renders them ineffective against zero-day attacks, as these attacks introduce novel patterns that are not represented in the training dataset [7].

The unsupervised approach, on the other hand, employs anomaly detection techniques to distinguish between normal system activities and anomalies. Unlike supervised methods, this approach does not rely on historical attack patterns, allowing it to identify previously unseen zero-day attacks [7,8]. By modeling the expected behavior of normal web traffic, anomaly detection-based methods can effectively flag deviations indicative of malicious activity, making them particularly suitable for dynamic and evolving attack landscapes.

The semi-supervised approach, which lies between supervised and unsupervised methods, leverages normal web request data to train the model. This approach focuses exclusively on learning the characteristics of legitimate requests, enabling the model to differentiate between normal and anomalous activities. Since it involves training on only one category of data—normal requests—it eliminates the need for labeled attack samples while still allowing for anomaly detection in the detection phase [7].

In this research, an unsupervised approach is employed to address the challenge of detecting zero-day attacks, which lack predefined signatures and attack patterns. By learning the distribution of normal web requests, the proposed model can effectively identify requests that deviate from the established normal behavior, thereby flagging them as potential zero-day attacks. This approach enhances the system's ability to detect novel and previously unknown threats, making it a robust solution for mitigating modern web security risks.

3. Related Work

Numerous methods and models have been proposed to counter web attacks, including zero-day attacks. Models by researchers like Pu et al. [7], Ingham et al. [9], Sivri et al. [10], Jung et al. [11], Vartouni et al. [12], Ariu et al. [13], Liang et al. [14], Kuang et al. [15], Tang et al. [16], Indrasiri et al. [17], Gong et al. [18], Tekerek et al. [19], Jemal et al. [20], Alaoui et

al. [21], Mohamed et al. [22], Yuan et al [23], Vorobyov et al. [24], Su et al. [25], Silvestre et al. [26], Yatagha et al. [27], Katbi et al. [28], Tokmak et al. [29], Alqhwazi et al. [30], Thalji et al. [31], Yao et al. [32], are notable.

The model by Pu et al. [7] introduces an unsupervised anomaly detection method that combines Sub-Space Clustering (SSC) and One Class Support Vector Machine (OCSVM). This approach aims at detecting cyber intrusions without prior knowledge of the attacks, making it particularly useful for identifying unknown or zero-day attacks.

The model by Ingham et al. [9] proposes a method for detecting web attacks by focusing on deep learning techniques, specifically utilizing Transformer models. This approach represents a major advancement in web security, providing a more dynamic and intelligent system for detecting and mitigating web-based attacks.

Sivri et al. [10] used various machine learning and deep learning models. For example, a hybrid model that uses character-level representations to classify HTTP requests as normal or malicious. The study employed models such as XGBoost, LightGBM, LSTM and CNN. The upsampling techniques have been used to balance the dataset, which helps improve classification metrics. As a result, the LSTM model achieved the best accuracy and F1 score, while LightGBM performed better in computation time. This work shows the importance of balancing in real-time web intrusion detection systems.

Jung et al. [11] used a novel approach named Payload Feature-Based Transfer Learning (PF-TL) to cope with insufficient training data in intrusion detection systems. Their method leverages knowledge transfer from a labeled source domain to an unlabeled target domain by extracting features from both the header and payload of network traffic. The technique they use is a hybrid feature extraction, combining signature-based and text vectorization methods, to enhance the representation of attack patterns.

The model by Vartouni et al. [12] uses a deep neural network-based method for feature learning and isolation forest for classification to identify malicious requests. It employs an n-gram model, which represents overlapping subsets of n characters from the data.

Ariu et al. [13] model is an intrusion detection system that represents payloads as byte sequences, analyzed using Hidden Markov Models (HMM). This proposed algorithm ensures the analytical power of n-gram analysis while overcoming its computational complexity, using HMM for feature extraction. However, HMM models are less effective when the sequence length is not appropriate, leading to poorer performance in processing complex requests.

In the Liang et al. [14] model, the approach involves first training two Recurrent Neural Networks (RNNs) with Complex Recurrent Units (LSTM or GRU units) to learn normal request patterns solely from unsupervised normal requests. Then, a supervised neural network classifier is trained, taking the output of the RNN as input to categorize normal and abnormal requests.

Kuang et al. [15] employ deep learning concepts to design a model named DeepWaf, a combination of LSTM and CNN deep neural networks, achieving satisfactory results in detecting web attacks.

In the Tang et al. [16] model, each word in an HTTP request (except for low-value words like 'and', 'or', etc.) is tokenized. Words and tokens are mapped to each other through TokenIDs. The tokenized request is then encoded and decoded using a Short-Term Memory architecture; if the decoded value matches the pre-encoded tokenized value, the request is benign, otherwise, it's malicious. The model primarily targets zero-day attacks, leaving known attack detection to WAF and addressing zero-day attacks through the Zero-Wall model. A limitation of this approach is that new benign requests with different patterns might be incorrectly flagged as malicious and subjected to further scrutiny in the Zero-Wall model before being identified as benign.

In the Indrasiri et al. [17] model, seven classification algorithms, one clustering algorithm, two ensemble methods, and two large standard datasets with 73,575 and 100,000 URLs were used. Two testing modes (percentage split, K-Fold cross-validation) were

employed for experiments and predictions. An ensemble model named ERG-SVC was proposed, using features selected by various feature selection methods.

The model by Gong et al. [18] proposes a method to improve web attack detection by incorporating model uncertainty into deep learning (DL) models, specifically focusing on Convolutional Neural Networks (CNNs). The method aims to address the problem of annotation errors in training data. Annotation errors are common in web attack datasets due to the vast and varied nature of web traffic, making correct labeling challenging.

The model by Tekerek et al. [19] introduces a novel approach for detecting web-based attacks using a deep learning architecture centered on Convolutional Neural Networks (CNNs). Focused on anomaly-based detection, this method preprocesses HTTP request data, particularly URLs and payloads, to identify unusual patterns indicative of potential threats.

The model by Jemal et al. [20] presents a smart web application firewall (SWAF) based on a convolutional neural network. The model is evaluated using a 5-fold cross-validation method. The CNN is characterized by a specific architecture. It can process data at scale and automatically extract and select features and consists of five layers.

The model by Alaoui et al. [21] proposes an approach based on Word2vec embedding and a stacked generalization ensemble model for LSTMs to detect malicious HTTP web requests.

Mohamed et al. [33] propose a deep learning-based multi-class intrusion detection system that classifies different types of web attacks using algorithms like LSTM, Bi-LSTM, CNN, and RNN. Automatic extraction and classification of features from HTTP traffic are their main approaches which overcome limitations related to traditional feature engineering.

The model by Shahid et al. [34] proposes a framework based on an enhanced hybrid approach where Deep Learning model is nested with a Cookie Analysis Engine for web attacks detection, mitigation and attacker profiling in real time.

The model by Moarref et al. [22] tries to focus on enhancing web attack detection via a character-level multichannel multilayer dilated convolutional neural network, processes HTTP request texts at the character level and extract relevant features. The model combines multichannel dilated convolutional blocks with varying kernel sizes to capture diverse temporal relationships and dependencies among characters.

Yatagha et al. [27] proposed a hybrid anomaly detection model combining VAE, LSTM, and OCSVM to detect zero-day anomalies in cyber-physical systems. The model learns normal patterns and flags deviations using reconstruction errors and latent space analysis. An adaptive loss adjustment algorithm ensures continual learning without forgetting. Deployed on a Raspberry Pi, the system effectively detects contextual anomalies in real-time.

Katbi et al. [28] proposed IDSVDD, a novel one-class anomaly detection framework for IoT environments that combines Deep SVDD with an interpolated adversarial autoencoder. The model enhances the structure of the latent space by enforcing convexity and regularization through adversarial interpolation, making it easier to distinguish anomalies from normal data. By learning a compact hypersphere that encloses only normal samples, the system achieves strong zero-day detection performance across multiple IoT datasets while remaining lightweight enough for deployment in resource-constrained environments.

Tokmak et al. [29] presented a deep learning framework for zero-day threat detection that combines Stacked Autoencoders (SAE) for feature selection with an LSTM classifier. Using the UGRansome dataset, the model first performed unsupervised feature extraction with SAE, then applied supervised LSTM layers to capture temporal patterns. The hybrid SAE-LSTM model achieved high accuracy (98%) across signature, anomaly, and synthetic signature attacks, showing strong generalization and effectiveness for detecting both known and novel threats.

Alqhwazi et al. [30] proposed an SQL injection detection system using a Recurrent Neural Network (RNN) Autoencoder, trained on a public Kaggle dataset of SQL queries.

Their architecture uses an autoencoder for dimensionality reduction, followed by an LSTM layer for classification. The model achieved 94% accuracy and 92% F1-score, outperforming traditional ML classifiers like SVM, decision tree, and logistic regression. This approach effectively captures long-term dependencies in SQL queries, making it well-suited for detecting complex or obfuscated injection attacks.

Thalji et al. [31] proposed AE-Net, a novel autoencoder-based feature engineering approach for detecting SQL injection attacks. AE-Net extracts high-level deep features from SQL textual queries, which are then used as input to multiple machine learning and deep learning models. Among the models tested, Extreme Gradient Boosting (XGBoost) achieved the highest performance, reaching a 0.99 accuracy score in k-fold cross-validation. The approach demonstrated the effectiveness of deep unsupervised feature learning in enhancing SQLi detection over traditional methods like BoW and TF-IDF.

Yao et al. [32] proposed a lightweight intrusion detection system for IoT that combines a One-Class Bidirectional GRU Autoencoder with Soft-Voting Ensemble Learning. The autoencoder is trained on only normal data to detect anomalies—including zero-day attacks—based on reconstruction loss. Detected anomalies are then classified using an ensemble of Random Forest, XGBoost, and LightGBM to identify the closest known attack type. The system demonstrated high accuracy and adaptability across three benchmark datasets: WSN-DS, UNSW-NB15, and KDD99.

Apart from machine learning techniques, various heuristic-based approaches have been proposed to detect web-based attacks, such as Yuan et al. [23] who proposed a static SQL injection detection technique based on program transformation to address the limitations of existing tools in handling object-oriented database extensions (OODBE) in PHP applications. Their method, OODBE-SCAN, first transforms object-oriented constructs into semantically equivalent procedural code, enabling accurate identification of source and sink points. The method then performs control flow graph construction and taint analysis to detect vulnerabilities. Compared to tools like RIPS and Seay, OODBE-SCAN demonstrated superior precision and recall in detecting real-world vulnerabilities in OODBE-based web applications.

Vorobyov et al. [24] introduced a novel runtime protection mechanism against SQL injection attacks based on synthesizing fine-grained allowlists from benign SQL queries. Their approach uses an information flow model to decompose SQL queries into semantic units called information tuples, which capture disclosed columns, accessed fields, and related predicates. By generalizing these tuples across a set of safe queries, they create a context-sensitive allowlist that permits future queries only if they disclose no more information than allowed. This method outperforms syntax-based detectors by focusing on semantic disclosure rather than structural similarity, thus reducing false positives and better preventing data exfiltration.

Su et al. [25] proposed Splendor, a static analysis framework for detecting stored Cross-Site Scripting (XSS) vulnerabilities in modern PHP web applications, especially those using Data Access Layers (DAL). The approach introduces a fuzzy token-matching technique to identify database operation triples (table, column, and operation type) from code fragments, even when SQL queries are dynamically constructed or obscured through encapsulation. Splendor then performs a two-phase taint analysis, tracing tainted data from sources to the database writes and from the database reads to sinks. The framework demonstrated strong scalability, identifying 17 real-world zero-day vulnerabilities and outperforming both static (RIPS) and dynamic (Black Widow) tools.

Silvestre et al. [26] introduced FreeSQLi, a novel static analysis tool that detects SQL injection vulnerabilities in PHP applications using session types. Their approach translates PHP code into the FreeST programming language, which supports rich type systems modeling communication protocols. By interpreting interactions between the application and the database as typed communication sessions, FreeSQLi checks for type mismatches—such as sending unsanitized user input (typed as Unsafe) to sensitive sinks—and flags these incon-

sistencies as SQLi vulnerabilities. This method offers formal guarantees and reduces false positives by leveraging strong type checking rather than heuristics or machine learning.

A key limitation of the related works is the omission of one of the most critical evaluation metrics: the False Positive Rate (FPR), which quantifies the proportion of normal web requests misclassified as malicious. This metric is essential for assessing the model's capability to accurately encode and decode normal requests, ensuring minimal disruption to legitimate traffic. In contrast, our proposed model demonstrates superior performance, achieving the lowest False Positive Rate of 0.2%, which is even lower than the values reported in prior studies that included this metric in their evaluation.

4. Proposed model

This section presents the proposed model for detecting zero-day web attacks. The detection process begins with the pre-processing of web requests through tokenization techniques, ensuring standardized input representation. These processed requests are then fed into an ensemble of relatively simple one-class classifiers designed to distinguish between normal and malicious web traffic. The effectiveness of the proposed model is assessed during the detection phase, focusing on its capability to identify and mitigate advanced security threats.

4.1. Architecture

The proposed model comprises multiple components, each serving a distinct role in the detection process. These components work together to predict whether an incoming web request is benign or malicious. The overall model architecture is depicted in Figure 1 for the training phase and Figure 2 for the testing phase.
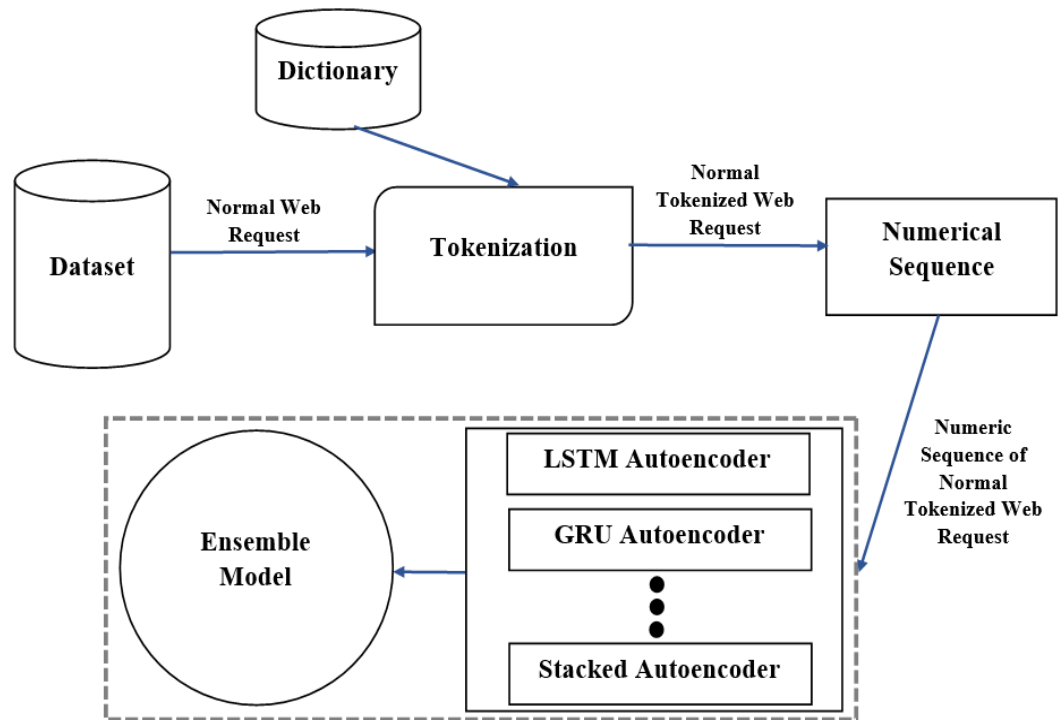

Figure 1. The proposed model in the training phase

During the training phase, the model is exclusively trained on normal web requests to establish a baseline pattern of legitimate traffic. In this phase, the model has full access to the training dataset, allowing it to learn the distribution of normal request patterns [35]. Conversely, in the testing phase, both normal and malicious web requests are input into the model for evaluation. This enables the model to assess its ability to generalize and detect deviations indicative of potential zero-day attacks.
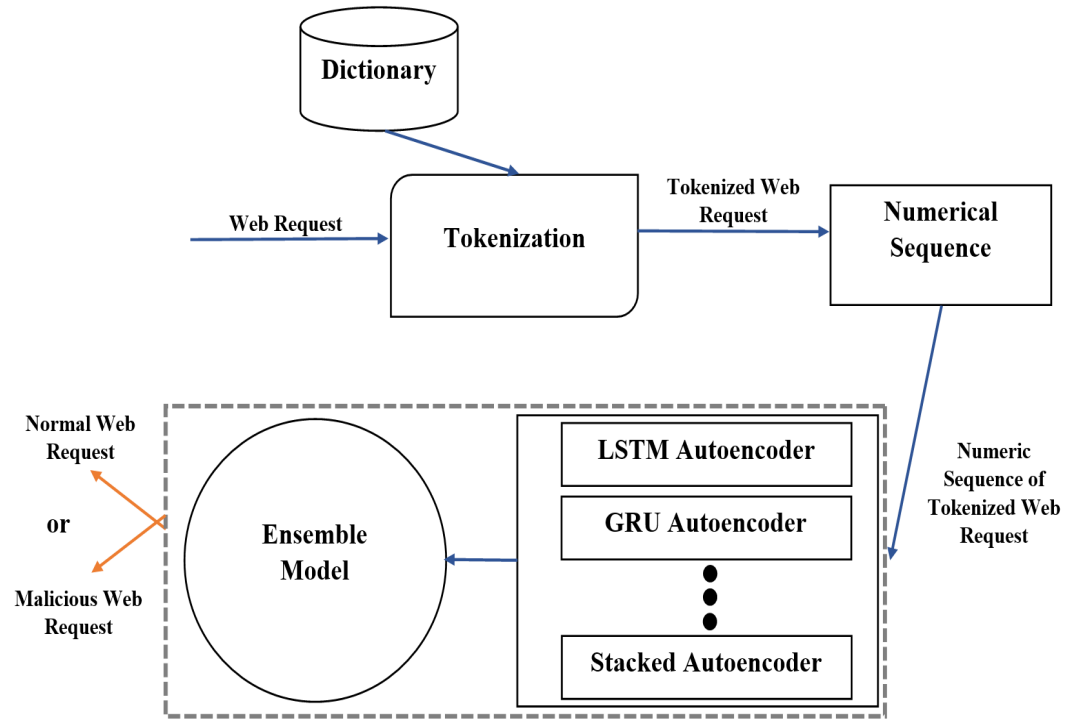
Figure 2. The proposed model in the test phase

4.1.1. Tokenization

A key innovation of the proposed model is the introduction of a novel tokenization technique for web requests, applied at the word level to both normal and malicious inputs [36]. Due to the inherent variability in the length and structure of web requests, this method addresses the challenges of training neural network-based models for web security, which arise from the inherent variability in the length and structure of web requests. Specifically, the model leverages anomaly detection principles to effectively distinguish between legitimate and anomalous web traffic [37].

To ensure consistent data representation, the pre-processing pipeline standardizes normal web requests through a dictionary-based tokenization approach [14]. This process involves segmenting each request at the word level using tools such as Python's Word-PunctTokenizer [38]. The resulting structured pattern is subsequently utilized as the input for training the ensemble model. We will explain the tokenization and data processing workflow by applying it to an example from our dataset. The example request is as follows:

```
POST /tienda1/publico/registro.jsp?modo=registro&login=m6&password=m6
&nombre=m&apellidos=m&email=m&dni=mm&direccion=Calle+Salvatierra+196+
%2C+&ciudad=m&provincia=31&cp=68970&ntc=6987987070987097&B1=Registrar
```

A predefined dictionary is utilized to categorize each character into distinct classes, facilitating structured tokenization. The dictionary includes categories such as *Alpha*, *AlphaNum*, *CapitalAlpha*, and *SpecialChar*, among others. For instance, in Figure 3, the token "login" is assigned the value "m6," which represents a combination of letters and numbers, classifying it under *AlphaNum* according to the predefined dictionary. Similarly, the word "Register," which begins with a capital letter, falls under the *CapitalLowerAlpha* category. These classifications enhance the accuracy of tokenization and facilitate the interpretation of web requests within the proposed model.

The significance of the tokenization approach in this model is twofold:

- **Data volume reduction:** The tokenization process optimizes data representation, reducing the complexity and size of input data, thereby improving computational efficiency.

- **Pattern identification for anomaly detection:** By establishing a structured pattern for normal web requests, the tokenization method enhances the model's ability to differentiate between legitimate and anomalous activities, ensuring higher accuracy in detecting malicious web requests.
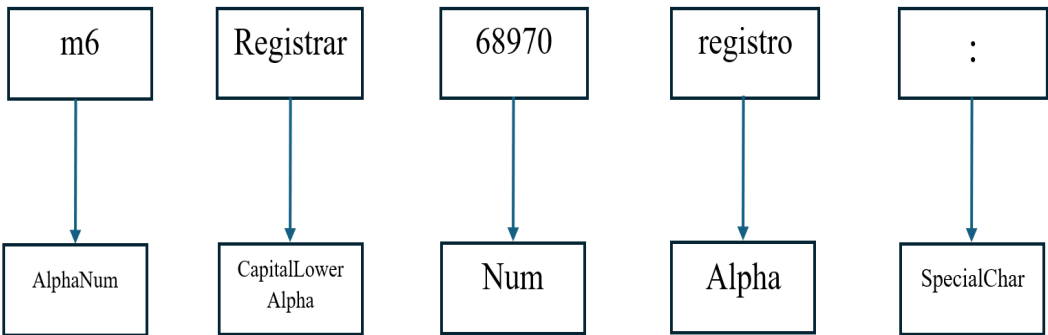
m6	Registrar	68970	registro	:
↓	↓	↓	↓	↓
AlphaNum	CapitalLower Alpha	Num	Alpha	SpecialChar

Figure 3. A Tokenized request.

4.1.2. Numerical Sequence

Following the tokenization of web requests on a word-by-word basis, each token must be mapped to a corresponding numerical value, as neural networks operate on numbers rather than raw text. This transformation is a critical step, as the varying range of input features necessitates data scaling before being processed by the model [39]. The tokenized text is converted into a structured numerical format suitable for input into the neural network. Figure 4 illustrates the mapping process, where individual words are assigned numerical values, facilitating seamless integration into the model.

To accommodate variations in the length of numerical sequences representing web requests, a padding mechanism is implemented. This ensures that all input sequences maintain a uniform length, preventing inconsistencies in model processing. Padding standardizes shorter sequences to match the fixed input length by appending neutral values, thereby standardizing input dimensions. This step enhances the model's ability to analyze and recognize patterns within the data, ultimately improving detection accuracy and performance.
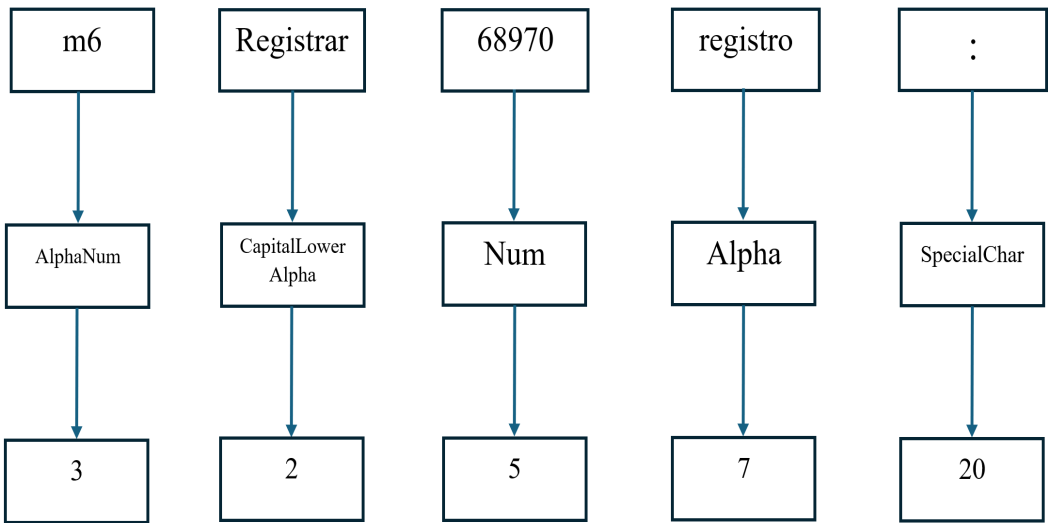
m6	Registrar	68970	registro	:
↓	↓	↓	↓	↓
AlphaNum	CapitalLower Alpha	Num	Alpha	SpecialChar
↓	↓	↓	↓	↓
3	2	5	7	20

Figure 4. Mapping words to numbers.

4.1.3. Ensemble Model

As illustrated in Figure 5, the proposed ensemble model consists of three sub-models: an LSTM autoencoder, a GRU autoencoder, and a stacked autoencoder. The initial phase of the proposed approach involves selecting appropriate sub-models for training and detection, ensuring an optimal balance between detection accuracy and computational efficiency. The primary objective is to design multiple lightweight sub-models that, when combined, enhance the overall detection capability while maintaining a low false positive rate.

To achieve this, various neural network architectures were evaluated. Experimental results demonstrated that employing Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and stacked autoencoders in an ensemble configuration enhances data processing efficiency and improves the accurate identification of both known and zero-day web attacks. The outputs of these sub-models are concatenated and further processed through a dense layer for feature reduction, optimizing the final classification process.

Autoencoders are widely utilized for dimensionality reduction and feature extraction. These models consist of an encoder and a decoder, both comprising multiple layers, which collectively transform an input sequence of symbols (words) into a continuous latent representation. The decoder then reconstructs the original input from this representation, preserving critical features while filtering out noise [40,41]. This reconstruction-based learning approach enables autoencoders to effectively capture underlying patterns in web requests, further improving the robustness of the detection framework.

The encoded sequences of the original input data, denoted as:

$$x = [x_1, x_2, x_3, \ldots, x_n]$$

The encoded sequences are obtained through specific encoding functions corresponding to each autoencoder type. The encoding transformations for the LSTM, GRU, and stacked autoencoders are formally defined as follows:

$$y_L = E_L(x) \tag{1}$$

$$y_G = E_G(x) \tag{2}$$

$$y_S = E_S(x) \tag{3}$$

These equations represent the encoding processes, wherein the input sequence x is mapped to its corresponding latent representation, effectively capturing essential features for the detection process.

For each type of autoencoder, the encoded representations are structured as follows:

$$y_L = \left[y_1^L, y_2^L, y_3^L, \ldots, y_m^L \right]$$

for the LSTM autoencoder,

$$y_G = \left[y_1^G, y_2^G, y_3^G, \ldots, y_m^G \right]$$

for the GRU autoencoder, and

$$y_S = \left[y_1^S, y_2^S, y_3^S, \ldots, y_m^S \right]$$

for the stacked autoencoder.

The output of the encoder serves as the input to the decoder, which reconstructs the original input sequence from the encoded representation. The reconstruction process is defined as follows:

$$\hat{x}_L = D_L(y_L) \tag{4}$$

$$\hat{x}_G = D_G(y_G) \tag{5}$$

$$\hat{x}_S = D_S(y_S) \tag{6}$$

where D_L, D_G, and D_S denote the decoding functions for the LSTM, GRU, and stacked autoencoders, respectively.
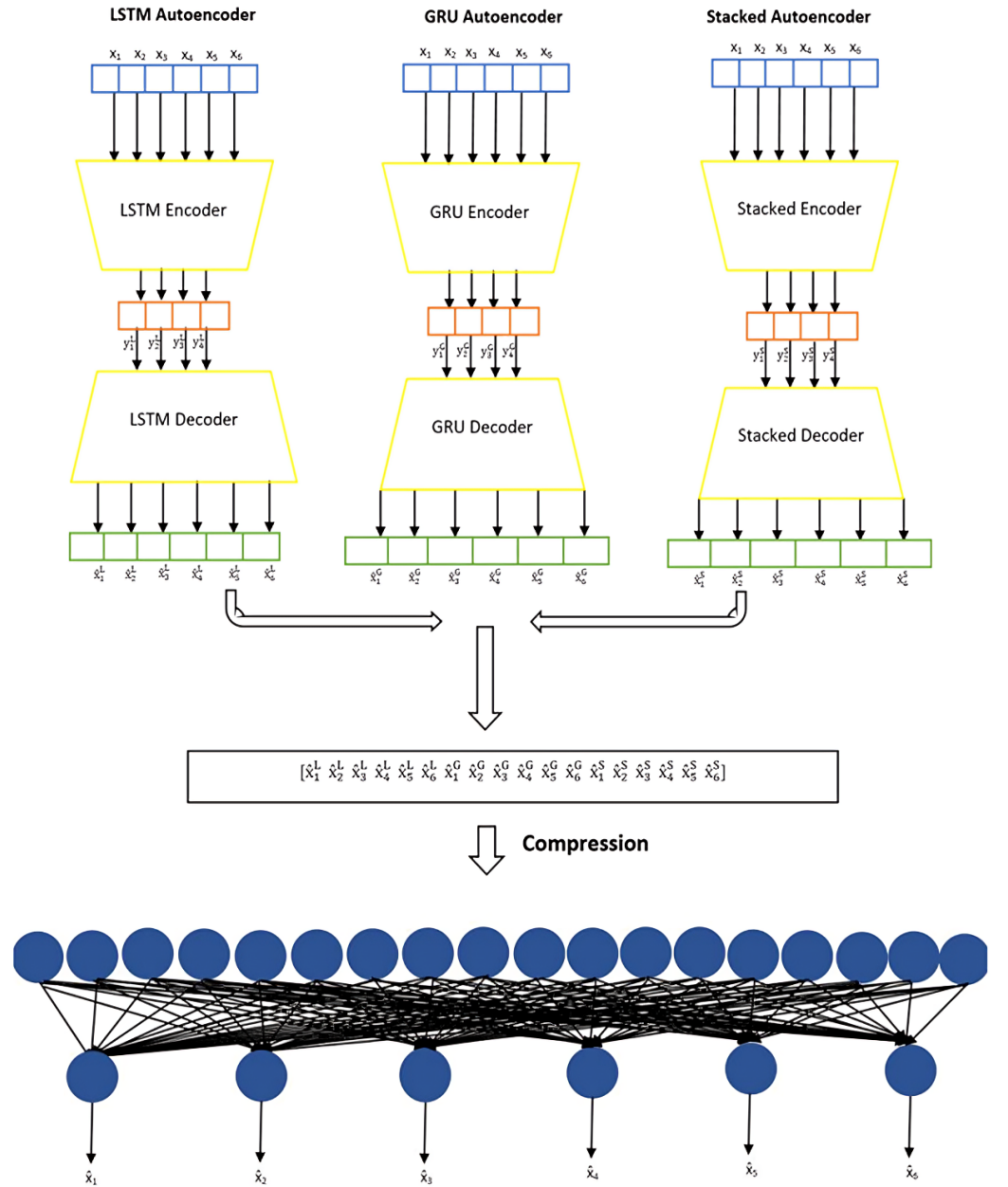

Figure 5. Structure of the proposed model.

The primary objective of the decoding process is to validate the quality and representativeness of the extracted features. The outputs from all three autoencoders are subsequently concatenated to form a unified feature representation, denoted as $\hat{x}$:

$$\hat{x} = \left[x_1^L, x_2^L, x_3^L, x_4^L, x_5^L, x_6^L, x_1^G, x_2^G, x_3^G, x_4^G, x_5^G, x_6^G, x_1^S, x_2^S, x_3^S, x_4^S, x_5^S, x_6^S \right]$$

To maintain consistency with the original input dimensions and optimize computational efficiency, a compression operation is applied to $\hat{x}$. This step ensures that the number of features in the final output aligns with that of the original input data, enabling effective processing and interpretation of web requests within the proposed model:

$$\hat{x} = [\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n]$$

This final transformation refines the extracted feature representations, ensuring that the model retains only the most relevant and informative aspects of the input data while discarding redundant or insignificant components. By optimizing the structure of the encoded sequences, the model enhances its capacity for accurate detection and classification of web requests.

5. Evaluation and Results

This section presents the evaluation of the proposed model and its sub-models based on multiple performance metrics, including accuracy, detection rate, sensitivity, precision, and false positive rate. To assess the effectiveness of the proposed approach, a threshold-based evaluation is conducted using the Mean Absolute Error (MAE), which quantifies the difference between the reconstructed request and the original input (prior to encoding).

In machine learning, MAE is a widely used metric for measuring the absolute difference between predicted values and their actual counterparts. It is computed by averaging the absolute errors across all predictions. MAE was selected as the primary evaluation metric due to its interpretability, robustness, and alignment with the model's objectives. Specifically, MAE measures the average absolute deviation between the original and reconstructed web requests, providing a clear and intuitive indicator of reconstruction accuracy.

Unlike Mean Squared Error (MSE), which disproportionately amplifies the effect of outliers due to its squared loss formulation, MAE is less sensitive to extreme deviations. This stability makes MAE particularly well-suited for anomaly detection, as it emphasizes individual discrepancies without being overly influenced by rare, extreme variations. By leveraging linear reconstruction errors, MAE effectively differentiates between benign and malicious web requests while ensuring an optimal balance between detection rates and false positives. This makes it an appropriate choice for evaluating the performance of web attack detection models.

The Mean Absolute Error (MAE) is formally defined as:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{x}_i - x_i| \tag{7}$$

In Formula (7), $\hat{x}_i$ represents the reconstructed value, while x_i denotes the actual value. The classification criterion is based on the computed MAE for a given web request: if the MAE falls below a predefined threshold, the request is classified as normal; otherwise, it is identified as malicious. The threshold value is determined using Figure 6, which visualizes the density distribution of web requests based on the MAE metric. This methodology effectively differentiates benign from malicious web requests, utilizing MAE as a robust anomaly detection measure. Based on the analysis in Figure 6, an optimal threshold value of approximately 4 is identified.

During training, the neural network is trained exclusively on 80% of normal web requests, while malicious requests are introduced only during the detection phase. The optimal threshold is determined empirically through iterative experimentation. The analysis suggests that a threshold value of 4.09 provides optimal detection performance, as requests with an MAE of 5 or higher are observed infrequently. However, setting the threshold too high (e.g., at 7) may improve training results but could fail to detect certain malicious requests with reconstruction errors in the range of 5 to 7. This would compromise the

model's effectiveness in distinguishing anomalous web activity. A practical example from the model's output further illustrates this decision-making process.
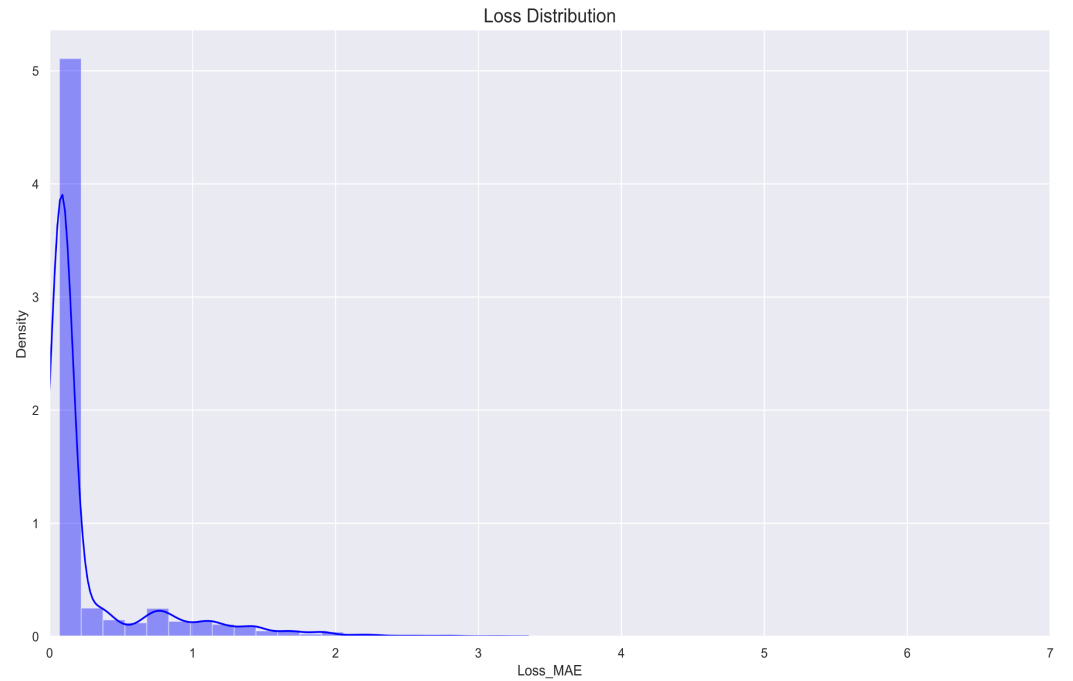

Figure 6. Density diagram according to MAE of the proposed model.

5.1. Data Collection

Previous research on detecting malicious and zero-day web requests has primarily relied on two well-established datasets: CSIC [42] and HTTPPARAMS [42]. These datasets provide a comprehensive representation of both normal and malicious web requests, making them widely used benchmarks in web security research. Additionally, a project hosted on GitHub [43], which employs Convolutional Neural Networks (CNNs), has utilized the CSIC 2012 dataset. Accordingly, the proposed model leverages this dataset for training and evaluation.

The dataset utilized in this study is significant due to the following characteristics:

- It encompasses a diverse range of malicious requests, including SQL Injection (SQLi), Cross-Site Scripting (XSS), and Buffer Overflow attacks.
- It contains normal (benign) web requests, ensuring a balanced distribution of data for effective training and evaluation.

A key consideration in dataset selection is ensuring that malicious requests accurately reflect real-world attack scenarios. The dataset comprises approximately 16,000 instances labeled as anomalous. However, certain anomalies may arise from factors unrelated to direct cyberattacks, such as unusual user behavior, malformed requests, or suspicious data entry attempts. These cases, while indicative of potential security threats, do not strictly conform to defined attack patterns. To maintain data integrity and ensure the model is trained on well-defined attack and normal request samples, such ambiguous anomalies are removed from the dataset prior to training.

5.2. The Ensemble Model Structure

According to Figure 7 the LSTM autoencoder and GRU autoencoder each consist of four layers (two encoder layers of 50 and 25 units respectively, and two symmetric decoder layers of 25 and 50 units), using the default tanh activation function. The stacked autoencoder comprises four dense layers (50, 25, 25, and 50 units) with linear activation. The ensemble model concatenates outputs from these autoencoders into a unified latent

vector, which is further compressed via a dense layer (50 units). All models are trained using the Mean Absolute Error (MAE) loss function, Nadam optimizer, and evaluated based on accuracy metrics.
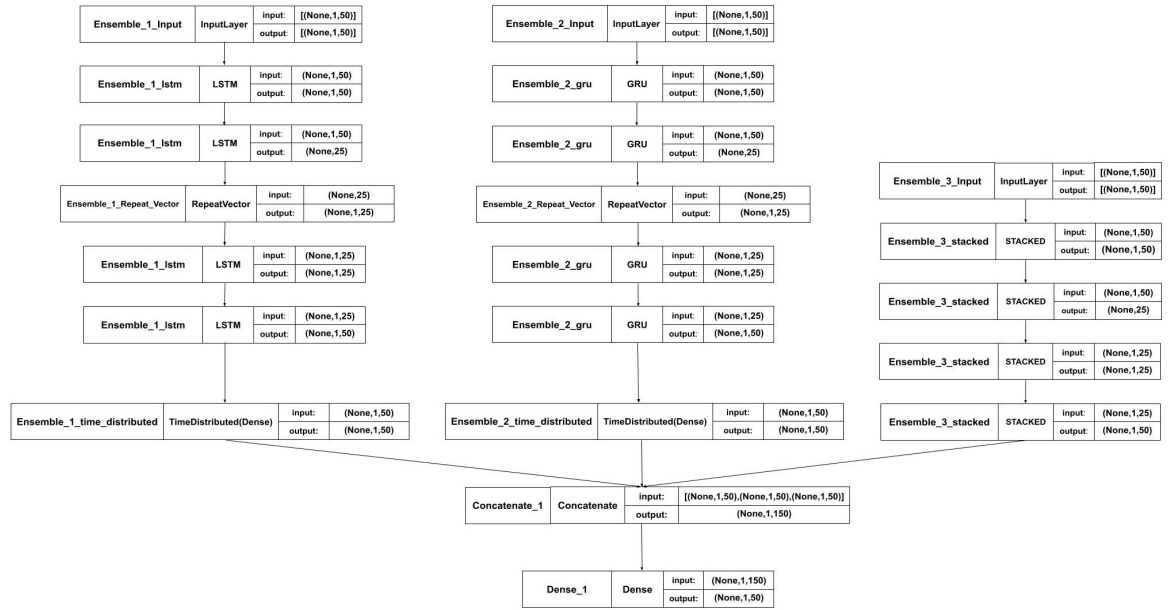

Figure 7. Ensemble model architecture.

The evaluation results of the proposed model, as depicted in Figure 8, illustrate the training process. Over the course of 90 epochs, the Mean Absolute Error (MAE) metric exhibited a significant reduction, decreasing from approximately 3 to around 0.5. This trend indicates that, in the initial training stages, a substantial discrepancy existed between the decoded and original values. However, as the ensemble model progressively learned the patterns of normal web requests, the reconstruction error diminished, reflecting an improvement in model accuracy.

The dataset utilized for training and validation followed an 80-20 split, with 80% of the data allocated for training and the remaining 20% reserved for validation. This partitioning ensures that the model is effectively trained while being rigorously evaluated on an unseen subset of data, enabling a more reliable assessment of its generalization capability.
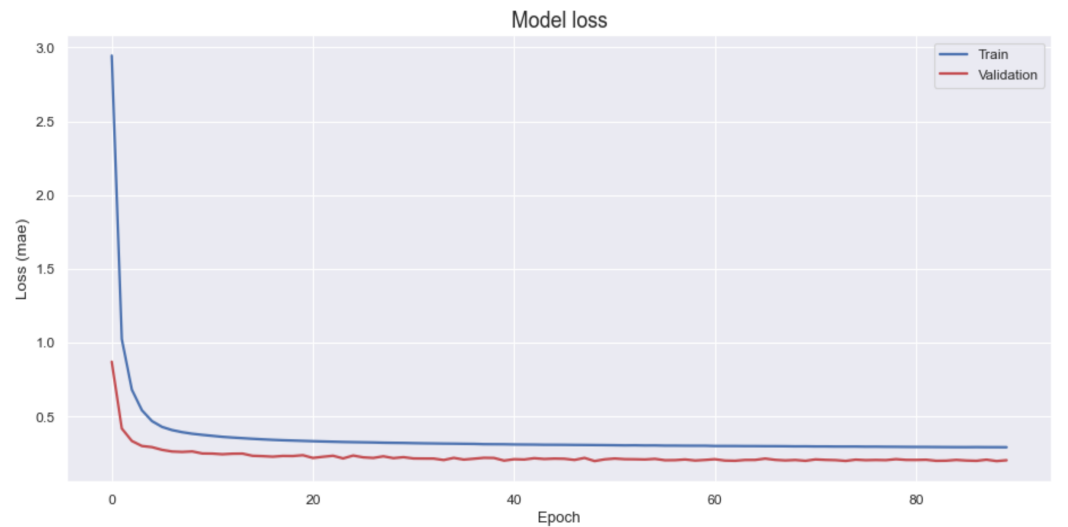

Figure 8. Training Process of the Proposed Model Based on MAE and Number of Epochs.

Following the detection phase, the Mean Absolute Error (MAE) for each web request is computed and compared against the predefined threshold. As illustrated in Figure 9, the calculated MAE values for individual web requests are represented in blue, while the threshold value is indicated by a red line. Web requests with an MAE exceeding the threshold (blue points above the red line) are classified as malicious, whereas those with an MAE below the threshold (blue points below the red line) are categorized as normal requests.

The ensemble model, incorporating LSTM, GRU, and stacked autoencoder sub-models, demonstrates superior performance across all evaluation metrics compared to each sub-model individually. The reported results represent the average performance obtained over six independent runs of the model.

The system utilized for evaluating the proposed model consists of key components, as shown in Table 1, including LSTM, GRU, and Stacked Autoencoder for neural network-based processing, running on Windows 11 with Python v3.12 as the programming language. The implementation leverages Scikit-learn v1.6.0 for machine learning functionalities and WordPunctTokenizer from the Natural Language Toolkit (NLTK) for splitting a text into a sequence of words. Additionally, the Tokenizer class is employed for converting text data into numerical sequences, ensuring compatibility with neural networks. The model's performance is evaluated using Mean Absolute Error (MAE), as previously defined, to quantifies the difference between predicted and actual values, providing an effective measure for anomaly detection. The training phase required approximately 20 seconds, while the test phase was completed in 5 seconds. The model itself was implemented using Python version 3.12 with the Keras framework.

Table 1. System components used in the evaluation setup.

System	Details
Neural Networks	LSTM, GRU, and Stacked Autoencoder
Operating System	Windows 11
Programming Language	Python v3.12
Python Library	Scikit-learn v1.6.0
Natural Language Toolkit (NLTK) Library	WordPunctTokenizer
Feature Extraction and Tokenization Tool	Tokenizer class in python
Evaluation Metric for measuring prediction accuracy	Mean Absolute Error (MAE)

5.3. Results

Table 2 defines the key terms used to compute the evaluation metrics. The performance assessment of the proposed model involves the computation of six primary metrics: accuracy, precision, sensitivity, detection rate, false positive rate, and F1 score.

The proposed ensemble model consistently outperforms the individual sub-models across all evaluation metrics. While the LSTM and GRU autoencoders achieve high accuracy, sensitivity, and precision, they exhibit a higher false positive rate, incorrectly classifying several normal requests as malicious. Conversely, the stacked autoencoder reduces the false positive rate effectively but shows comparatively weaker precision and recall. Combining these sub-models into a unified ensemble framework leverages their complementary strengths, thereby significantly improving overall detection performance.

A detailed analysis reveals that both LSTM and GRU sub-models misclassified 14 out of 1,299 normal requests as malicious—an undesirable outcome in real-world scenarios. Incorporating the stacked autoencoder into the ensemble mitigates this issue by
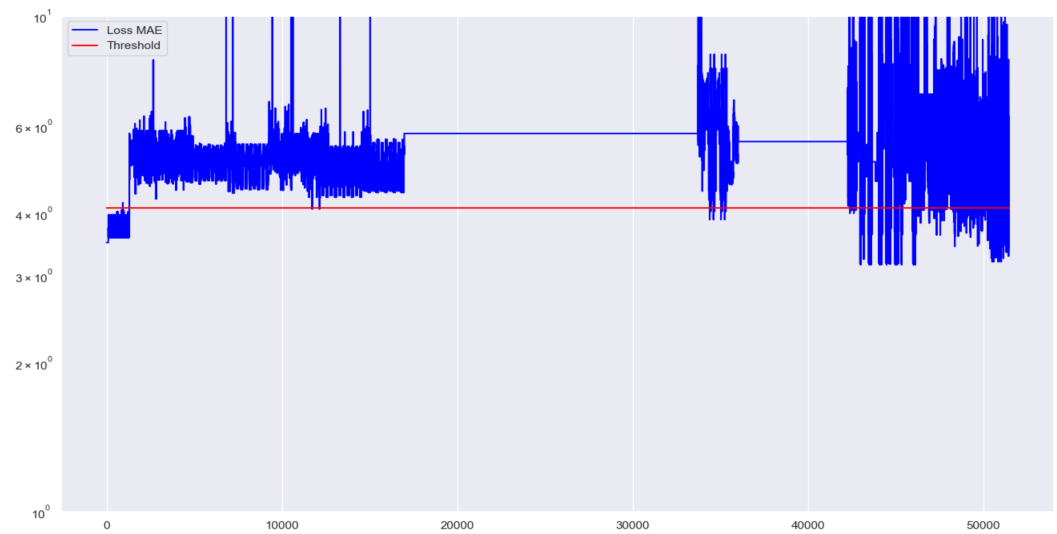
Figure 9. Comparison of the Calculated MAE Value for Each Web Request with the Threshold Value.

reducing false positives, albeit at the expense of slightly lower accuracy and recall when used independently. Table 3 clearly illustrates the comparative performance of each individual sub-model against the proposed ensemble approach and Figure 10 illustrates this comparison in the form of a bar chart.

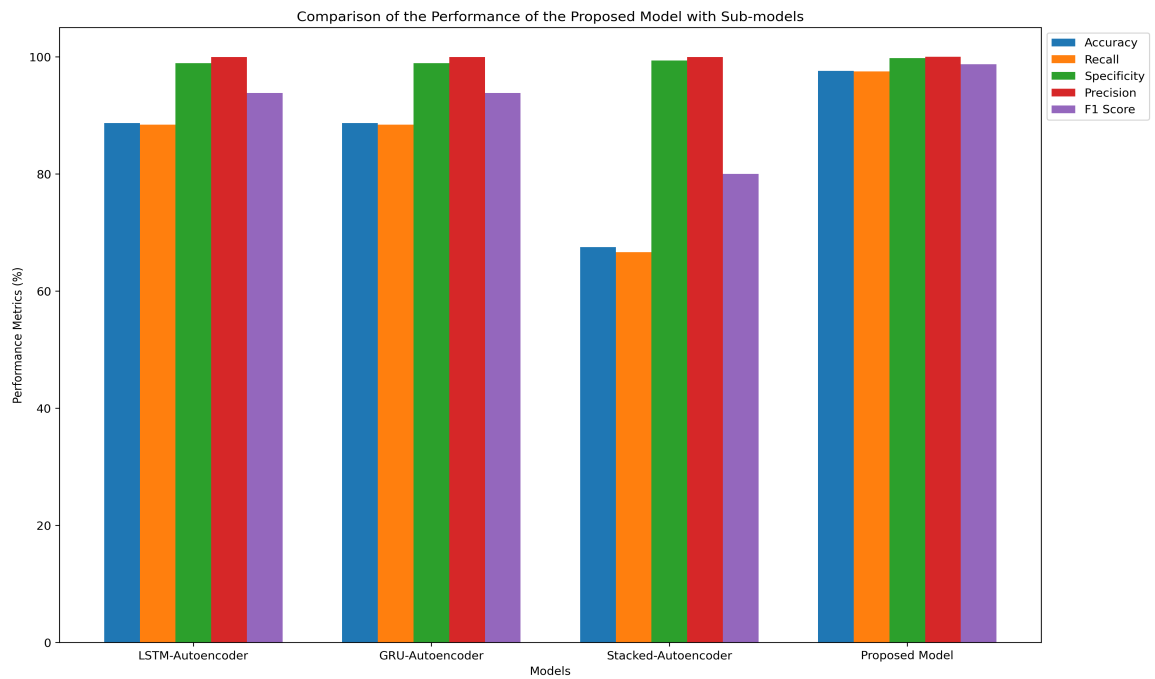**Table 2.** Performance metrics used for the proposed model.

Metric	Definition/Calculation	Value
Total (T)	Total number of requests	51473
Correct Predictions (CP)	Number of correctly predicted requests	50230
Negatives	Normal requests	1299
Positives	Malicious requests	50174
True Negatives (TN)	Normal requests correctly predicted as normal	1296
False Positives (FP)	Normal requests incorrectly predicted as malicious	3
False Negatives (FN)	Malicious requests incorrectly predicted as normal	1240
True Positives (TP)	Malicious requests correctly predicted as malicious	48934
Accuracy	$\frac{TN+TP}{TP+TN+FP+FN}$	0.9758
Recall (Sensitivity)	$\frac{TP}{TP+FN}$	0.9752
Specificity	$\frac{TN}{TN+FP}$	0.9976
Precision	$\frac{TP}{TP+FP}$	0.9999
False Positive Rate	$1 - \text{Specificity}$	0.002
F1 Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	0.9874

These results demonstrate that the ensemble strategy effectively balances the distinct advantages and limitations of each sub-model, providing a robust and reliable solution for detecting zero-day web attacks.

Table 4 presents a comparative analysis of the proposed model's performance against various models from previous research that have utilized the CSIC2010 and CSIC2012 datasets. This comparison provides insights into the effectiveness of the proposed approach relative to existing solutions in the field. One notable limitation in prior studies is the omission of the False Positive Rate (FPR) in their evaluation results. This metric is crucial, as it quantifies the number of normal requests misclassified as malicious, directly impacting the practical applicability of detection models. Figure 11 illustrates this comparison in the

Table 3. Comparison of the Performance of the Proposed Model with the Sub-models.

Models	Accuracy	Recall	Specificity	Precision	False Positive Rate	F1 Score
LSTM-Autoencoder	88.68%	88.41%	98.92%	99.96%	1%	93.83%
GRU-Autoencoder	88.69%	88.42%	98.92%	99.96%	1%	93.83%
Stacked-Autoencoder	67.48%	66.65%	99.38%	99.97%	0.6%	79.98%
Proposed Model	**97.58%**	**97.52%**	**99.76%**	**99.99%**	**0.2%**	**98.74%**

Figure 10. Comparison of the Performance of the Proposed Model with the Sub-models in the form of a bar chart.

form of a bar chart. In the figure, models that utilized the CSIC2012 dataset are highlighted in red.

The primary comparison focuses on studies that have employed the CSIC2012 dataset [10, 11,33], as they provide the most directly comparable benchmark. However, to offer a broader perspective, we also include studies based on the CSIC2010 dataset [12,14,15,18–22,34]. It is important to note that differences in dataset characteristics may influence the comparability of results.
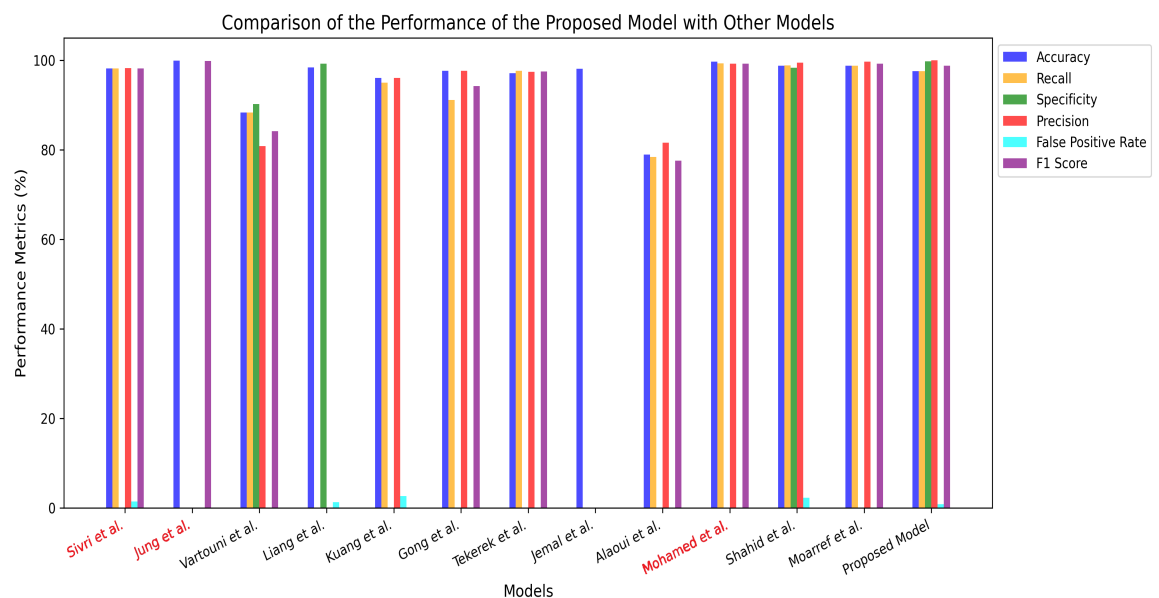
The CSIC2010 and CSIC2012 datasets are widely recognized benchmarks for evaluating web application security models, particularly for detecting SQL Injection (SQLi) and other web-based attacks. The CSIC2010 dataset, developed earlier, contains a diverse set of normal and anomalous HTTP requests. While it provides a solid foundation for studying web attack detection, it lacks the complexity and evolving attack patterns characteristic of modern cybersecurity threats.

To address these limitations, the CSIC2012 dataset was designed with more sophisticated and realistic attack scenarios, along with a broader range of normal traffic. This makes CSIC2012 a more representative dataset for contemporary web security challenges. Additionally, CSIC2012 includes refined labeling and a larger volume of data, enhancing its suitability for training and evaluating advanced machine learning models.

These distinctions underscore the importance of selecting CSIC2012 for research targeting modern web application threats, as it serves as a more rigorous and up-to-date evaluation benchmark compared to its predecessor.

Table 4. Comparison of the Performance of the Proposed Model with Previously Designed Models.

Models	Accuracy	Recall	Specificity	Precision	False Positive Rate	F1 Score
Sivri et al. [10]	98.15%	98.15%	-	98.20%	0.8%	98.16%
Jung et al. [11]	99.88%	-	-	-	-	99.80%
Vartouni et al. [12]	88.32%	88.34%	90.20%	80.79%	-	84.12%
Liang et al. [14]	98.42%	-	99.21%	-	0.7%	-
Kuang et al. [15]	96%	95%	-	96%	2%	-
Gong et al. [18]	97.64%	91.11%	-	97.62%	-	94.25%
Tekerek et al. [19]	97.07%	97.59%	-	97.43%	-	97.51%
Jemal et al. [20]	98.1%	-	-	-	-	-
Alaoui et al. [21]	78.95%	78.41%	-	81.54%	-	77.57%
Mohamed et al. [33]	99.66%	99.28%	-	99.18%	-	99.22%
Shahid et al. [34]	98.73%	98.87%	98.33%	99.41%	1.67%	99.13%
Moarref et al. [22]	99.36%	98.80%	-	99.65%	-	99.22%
Proposed Model	**97.58%**	**97.52%**	**99.76%**	**99.99%**	**0.2%**	**98.74%**

Figure 11. Comparison of the Performance of the Proposed Model with Previously Designed Models in the form of a bar chart.

6. Discussion

According to the obtained results, the proposed model demonstrates strong performance across all evaluation metrics, particularly in terms of the False Positive Rate (FPR). The model achieves the lowest FPR compared to related work referenced in this study, highlighting its ability to accurately differentiate between normal and malicious web requests. This is a critical aspect of web security, as a high FPR would indicate an increased likelihood of blocking legitimate users, thereby compromising usability.

6.1. Effectiveness of the Ensemble Model

The proposed ensemble model comprises three fundamental sub-models: LSTM, GRU, and Stacked Autoencoder. Each sub-model plays a distinct role in detecting malicious web requests, contributing unique advantages:

- **LSTM Autoencoder:** Captures long-term dependencies in web request sequences, enhancing the ability to recognize complex request structures.
- **GRU Autoencoder:** Provides computational efficiency while preserving strong sequential pattern recognition capabilities.

- **Stacked Autoencoder:** Focuses on dimensionality reduction and latent feature extraction, improving anomaly detection in subtle attack patterns.

By integrating these sub-models, the ensemble model achieves a well-balanced performance across multiple evaluation metrics, effectively mitigating the weaknesses of individual sub-models by an innovative concatenation and feature-compression mechanism. This structured ensemble method distinctly differs from traditional ensemble strategies and clearly enhances computational efficiency and detection accuracy. Additionally, our advanced tokenization method provides a unique and consistent input representation that significantly improves anomaly detection compared to conventional tokenization strategies. Our explicit evaluation of the False Positive Rate further distinguishes our work by providing practical insights often neglected in related literature. The ensemble approach significantly enhances accuracy, recall, and precision while substantially reducing false positives compared to each sub-model independently.

6.2. Practical Considerations and Limitations

Although the proposed model demonstrates robust performance on benchmark datasets, its real-world deployment in web security applications requires additional considerations. One of the primary challenges is the variability in web request patterns across different applications and domains. While the tokenization and feature extraction methods effectively standardize the CSIC2012 dataset, these techniques should be adapted and applied to other datasets, such as FWAF and HTTPParams [44], to enhance generalization.

Additionally, while the model is designed to detect zero-day attacks, it does not explicitly classify attack types (e.g., SQL Injection, Cross-Site Scripting (XSS), Buffer Overflow). Future research should focus on integrating an attack-type classification mechanism alongside anomaly detection to improve forensic analysis and threat response capabilities.

7. Conclusions

In this study, each web request was initially segmented into individual words and then tokenized using a predefined vocabulary. This preprocessing step aimed to standardize and simplify web requests while establishing a structured pattern for normal web traffic. In the final stage of preprocessing, each tokenized word was mapped to a unique numerical representation, facilitating its input into the neural network. The proposed model employs an ensemble approach comprising three relatively simple sub-models: LSTM, GRU, and Stacked Autoencoders. The ensemble operates by independently processing the input data through each sub-model and then outputs are explicitly concatenated into a combined latent feature set, ensuring the ensemble benefits from the diverse representation capabilities of each sub-model. After concatenation, a dedicated Dense layer compresses the resulting features into a unified, optimized representation, significantly reducing the dimensionality from a larger combined vector to a manageable size. A novel structured tokenization method significantly enhancing detection performance, and explicit evaluation of critical metrics including False Positive Rate.

During the training phase, only normal web requests were provided as input to the ensemble model, enabling it to learn the underlying patterns of legitimate requests. Upon completion of training, the model effectively captured and recognized these patterns. In the detection phase, both normal and malicious web requests were introduced for evaluation. The Mean Absolute Error (MAE) was employed as the primary metric to quantify the difference between the reconstructed and original values of each request. The threshold for classification was determined based on the MAE values computed during the training phase. In the detection phase, if the MAE of a web request was below the threshold, it was classified as normal; otherwise, it was identified as malicious.

During evaluation, the ensemble model's performance was compared against each of its sub-models individually. The results demonstrated that the ensemble approach achieved superior performance, particularly in terms of an increased detection rate and a reduced false positive rate. Additionally, the proposed model was benchmarked against

prior research, where it consistently outperformed existing approaches, further validating its effectiveness in detecting web-based threats.

8. Future Work

One potential direction for future research involves enhancing the tokenization and feature extraction process [45] across diverse web attack datasets. This improvement can be achieved through the application of Generative AI, leveraging Large Language Models (LLMs) [46,47]. Specifically, prompt engineering [48,49] can be employed to construct a structured prompt that systematically guides the LLM in preprocessing each dataset sample.

A few-shot learning approach can be utilized to accomplish this by providing representative examples from each dataset, detailing the exact tokenization process required for each sample. This method ensures that the LLM internalizes dataset-specific preprocessing rules while maintaining uniformity across different datasets.

Another approach involves automating the preprocessing pipeline by generating customized script code [50] for each dataset. This method leverages LLMs to autonomously generate preprocessing scripts based on structured prompts that define tokenization rules and feature extraction strategies. While manual scripting was employed in this study for data preprocessing, future research will focus on automating this process using LLMs, thereby minimizing human intervention and improving scalability across diverse datasets.

Beyond preprocessing improvements, future efforts will explore the implementation of advanced neural architectures and anomaly detection approaches, such as Bidirectional LSTM, GRU, and Convolutional Neural Networks (CNNs) [51], to develop a more robust ensemble model for web attack detection. Additionally, feature selection techniques will be applied to retain high-information-value features while eliminating less significant ones, effectively reducing input dimensionality and enhancing computational efficiency. Regarding advanced anomaly detection approaches, For instance, [52] introduced VulnSage, a framework leveraging structured reasoning strategies such as Chain-of-Thought and Think and Verify to improve zero-shot vulnerability detection in software systems. Inspired by this work, future studies could explore the use of similar structured reasoning strategies in our web attack detection framework, potentially improving the detection of complex, multi-component web vulnerabilities by incorporating explicit logical analysis and self-verification mechanisms.

Furthermore, the integration of a voting-based ensemble learning approach will be investigated to improve detection accuracy by combining multiple classifiers. This strategy aims to leverage the strengths of individual models, resulting in a more generalized and resilient web attack detection framework.

References

1. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; Tawalbeh, L. Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review* **2023**, *56*, 10733–10811.
2. Dawadi, B.R.; Adhikari, B.; Srivastava, D.K. Deep learning technique-enabled web application firewall for the detection of web attacks. *Sensors* **2023**, *23*, 2073.
3. Yang, J.; Chen, Y.L.; Por, L.Y.; Ku, C.S. A systematic literature review of information security in chatbots. *Applied Sciences* **2023**, *13*, 6355.
4. Calzavara, S.; Conti, M.; Focardi, R.; Rabitti, A.; Tolomei, G. Machine learning for web vulnerability detection: the case of cross-site request forgery. *IEEE Security & Privacy* **2020**, *18*, 8–16.
5. Kalla, D.; Mohammed, A.S.; Boddapati, V.N.; Jiwani, N.; Kiruthiga, T. Investigating the Impact of Heuristic Algorithms on Cyberthreat Detection. In Proceedings of the 2024 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT), 2024, Vol. 1, pp. 450–455.
6. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* **2016**, *60*, 19–31.
7. Pu, G.; Wang, L.; Shen, J.; Dong, F. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Science and Technology* **2020**, *26*, 146–153.
8. Long, H.V.; Tuan, T.A.; Taniar, D.; Can, N.V.; Hue, H.M.; Son, N.T.K. An efficient algorithm and tool for detecting dangerous website vulnerabilities. *International Journal of Web and Grid Services* **2020**, *16*, 81–104.

9. Ingham, K.L.; Somayaji, A.; Burge, J.; Forrest, S. Learning DFA representations of HTTP for protecting web applications. *Computer Networks* **2007**, *51*, 1239–1255.

10. Sivri, T.T.; Akman, N.P.; Berkol, A.; Peker, C. Web intrusion detection using character level machine learning approaches with upsampled data. *Annals of Computer Science and Information Systems* **2022**, *32*.

11. Jung, I.; Lim, J.; Kim, H.K. PF-TL: Payload feature-based transfer learning for dealing with the lack of training data. *Electronics* **2021**, *10*, 1148.

12. 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). IEEE, 2018, pp. 131–134.

13. Ariu, D.; Tronci, R.; Giacinto, G. HMMPayl: An intrusion detection system based on Hidden Markov Models. *computers & security* **2011**, *30*, 221–241.

14. Liang, J.; Zhao, W.; Ye, W. Anomaly-based web attack detection: a deep learning approach. In Proceedings of the Proceedings of the 2017 VI International Conference on Network, Communication and Computing, 2017, pp. 80–85.

15. Kuang, X.; Zhang, M.; Li, H.; Zhao, G.; Cao, H.; Wu, Z.; Wang, X. DeepWAF: detecting web attacks based on CNN and LSTM models. In Proceedings of the Cyberspace Safety and Security: 11th International Symposium, CSS 2019, Guangzhou, China, December 1–3, 2019, Proceedings, Part II 11. Springer, 2019, pp. 121–136.

16. Tang, R.; Yang, Z.; Li, Z.; Meng, W.; Wang, H.; Li, Q.; Sun, Y.; Pei, D.; Wei, T.; Xu, Y.; et al. Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 2020, pp. 2479–2488.

17. Indrasiri, P.L.; Halgamuge, M.N.; Mohammad, A. Robust ensemble machine learning model for filtering phishing URLs: Expandable random gradient stacked voting classifier (ERG-SVC). *Ieee Access* **2021**, *9*, 150142–150161.

18. Gong, X.; Lu, J.; Zhou, Y.; Qiu, H.; He, R. Model uncertainty based annotation error fixing for web attack detection. *Journal of Signal Processing Systems* **2021**, *93*, 187–199.

19. Tekerek, A. A novel architecture for web-based attack detection using convolutional neural network. *Computers & Security* **2021**, *100*, 102096.

20. Jemal, I.; Haddar, M.A.; Cheikhrouhou, O.; Mahfoudhi, A. SWAF: a smart web application firewall based on convolutional neural network. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN). IEEE, 2022, pp. 01–06.

21. Alaoui, R.L.; et al. Web attacks detection using stacked generalization ensemble for LSTMs and word embedding. *Procedia Computer Science* **2022**, *215*, 687–696.

22. Moarref, N.; Sandıkkaya, M.T. MC-MLDCNN: Multichannel Multilayer Dilated Convolutional Neural Networks for Web Attack Detection. *Security and Communication Networks* **2023**, *2023*, 2415288.

23. Yuan, Y.; Lu, Y.; Zhu, K.; Huang, H.; Yu, L.; Zhao, J. A Static Detection Method for SQL Injection Vulnerability Based on Program Transformation. *Applied Sciences* **2023**, *13*.

24. Vorobyov, K.; Gauthier, F.; Krishnan, P. Synthesis of Allowlists for Runtime Protection against SQLi. In Proceedings of the Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results. Association for Computing Machinery, 2024, p. 16–20.

25. Su, H.; Li, F.; Xu, L.; Hu, W.; Sun, Y.; Sun, Q.; Chao, H.; Huo, W. Splendor: Static Detection of Stored XSS in Modern Web Applications. In Proceedings of the Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. Association for Computing Machinery, 2023, p. 1043–1054.

26. Silvestre, A.; Medeiros, I.; Mordido, A. Towards a SQL Injection Vulnerability Detector Based on Session Types. In Proceedings of the Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE. INSTICC, SciTePress, 2024, pp. 711–718.

27. Yatagha, R.; Nebebe, B.; Waedt, K.; Ruland, C. Towards a Zero-Day Anomaly Detector in Cyber Physical Systems Using a Hybrid VAE-LSTM-OCSVM Model. In Proceedings of the Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 5038–5045.

28. Katbi, A.; Ksantini, R. One-class IoT anomaly detection system using an improved interpolated deep SVDD autoencoder with adversarial regularizer. *Digital Signal Processing* **2025**, p. 105153.

29. Tokmak, M.; Nkongolo, M. Stacking an autoencoder for feature selection of zero-day threats. *arXiv preprint arXiv:2311.00304* **2023**.

30. Alghawazi, M.; Alghazzawi, D.; Alarifi, S. Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model. *Mathematics* **2023**, *11*, 3286.

31. Thalji, N.; Raza, A.; Islam, M.S.; Samee, N.A.; Jamjoom, M.M. Ae-net: Novel autoencoder-based deep features for sql injection attack detection. *IEEE Access* **2023**, *11*, 135507–135516.

32. Yao, W.; Hu, L.; Hou, Y.; Li, X. A lightweight intelligent network intrusion detection system using one-class autoencoder and ensemble learning for IoT. *Sensors* **2023**, *23*, 4141.

33. Mohamed, S.M.; Rohaim, M.A. Multi-Class Intrusion Detection System using Deep Learning. *Journal of Al-Azhar University Engineering Sector* **2023**, *18*, 869–883.

34. Shahid, W.B.; Aslam, B.; Abbas, H.; Khalid, S.B.; Afzal, H. An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling. *Journal of Network and Computer Applications* **2022**, *198*, 103270.

35. Thomas, S.; Koleini, F.; Tabrizi, N. Dynamic defenses and the transferability of adversarial examples. In Proceedings of the 2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA). IEEE, 2022, pp. 276–284.
36. Khalid, M.N.; Farooq, H.; Iqbal, M.; Alam, M.T.; Rasheed, K. Predicting web vulnerabilities in web applications based on machine learning. In Proceedings of the Intelligent Technologies and Applications: First International Conference, INTAP 2018, Bahawalpur, Pakistan, October 23-25, 2018, Revised Selected Papers 1. Springer, 2019, pp. 473–484.
37. Levene, M.; Poulovassilis, A.; Davison, B.D. Learning web request patterns. *Web Dynamics: Adapting to Change in Content, Size, Topology and Use* **2004**, pp. 435–459.
38. Vijayarani, S.; Janani, R.; et al. Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)* **2016**, *3*, 37–47.
39. Rashvand, N.; Hosseini, S.S.; Azarbayjani, M.; Tabkhi, H. Real-Time Bus Arrival Prediction: A Deep Learning Approach for Enhanced Urban Mobility. *arXiv preprint arXiv:2303.15495* **2023**.
40. Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems* **2017**.
41. Rashvand, N.; Witham, K.; Maldonado, G.; Katariya, V.; Marer Prabhu, N.; Schirner, G.; Tabkhi, H. Enhancing automatic modulation recognition for iot applications using transformers. *IoT* **2024**, *5*, 212–226.
42. Shaheed, A.; Kurdy, M.B. Web application firewall using machine learning and features engineering. *Security and Communication Networks* **2022**, *2022*, 5280158.
43. DuckDuckBug. CNN Web Application Firewall. https://github.com/DuckDuckBug/cnn_waf, 2023. Accessed: January 29, 2025.
44. Jagat, R.R.; Sisodia, D.S.; Singh, P. Detecting web attacks from HTTP weblogs using variational LSTM autoencoder deviation network. *IEEE Transactions on Services Computing* **2024**.
45. Abshari, D.; Fu, C.; Sridhar, M. LLM-assisted Physical Invariant Extraction for Cyber-Physical Systems Anomaly Detection. *arXiv preprint arXiv:2411.10918* **2024**.
46. Zibaeirad, A.; Koleini, F.; Bi, S.; Hou, T.; Wang, T. A comprehensive survey on the security of smart grid: Challenges, mitigations, and future research opportunities. *arXiv preprint arXiv:2407.07966* **2024**.
47. Abshari, D.; Sridhar, M. A Survey of Anomaly Detection in Cyber-Physical Systems. *arXiv preprint arXiv:2502.13256* **2025**.
48. Babaey, V.; Ravindran, A. GenSQLi: A Generative Artificial Intelligence Framework for Automatically Securing Web Application Firewalls Against Structured Query Language Injection Attacks. *Future Internet* **2025**, *17*, 8.
49. Babaey, V.; Ravindran, A. GenXSS: an AI-Driven Framework for Automated Detection of XSS Attacks in WAFs. *arXiv preprint arXiv:2504.08176* **2025**.
50. White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D.C. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* **2023**.
51. Graves, A.; Jaitly, N.; Mohamed, A.r. Hybrid speech recognition with deep bidirectional LSTM. In Proceedings of the 2013 IEEE workshop on automatic speech recognition and understanding. IEEE, 2013, pp. 273–278.
52. Zibaeirad, A.; Vieira, M. Reasoning with LLMs for Zero-Shot Vulnerability Detection. *arXiv preprint arXiv:2503.17885* **2025**.