computers

MDPI

Article

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, Charlotte, NC 28223, USA
[2] Research Engineer, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
* Correspondence: vbabaey@charlotte.edu (V.B.); hamid.faragardi@trendplus.se (H.R.F.)

Abstract: The increasing sophistication of web-based services has intensified the risk of zero-day attacks, exposing critical vulnerabilities in user information security. Traditional detection systems often rely on labeled attack data and struggle to identify novel threats without prior knowledge. This paper introduces a novel one-class ensemble method for detecting zero-day web attacks, combining the strengths of Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and stacked autoencoders through latent representation concatenation and compression. Additionally, a structured tokenization strategy based on character-level analysis is employed to enhance input consistency and reduce feature dimensionality. The proposed method was evaluated using the CSIC 2012 dataset, achieving 97.58% accuracy, 97.52% recall, 99.76% specificity, and 99.99% precision, with a false positive rate of just 0.2%. Compared to conventional ensemble techniques like majority voting, our approach demonstrates superior anomaly detection performance by fusing diverse feature representations at the latent level rather than the output level. These results highlight the model's effectiveness in accurately detecting unknown web attacks with low false positives, addressing major limitations of existing detection frameworks.

Keywords: zero-day attacks; tokenization; autoencoder; ensemble classification; neural networks; LSTM; GRU; stacked

Academic Editors: Leandros Maglaras and Helge Janicke

Received: 17 April 2025
Revised: 21 May 2025
Accepted: 22 May 2025
Published: 26 May 2025

Citation: Babaey, V.; Faragardi, H.R. Detecting Zero-Day Web Attacks with an Ensemble of LSTM, GRU, and Stacked Autoencoders. *Computers* **2025**, *14*, 205. https://doi.org/10.3390/computers14060205

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 and 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 for addressing evolving attack patterns [3]. 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. WAFs remain limited in addressing zero-day and obfuscated web attacks. Studies have shown that even modern WAFs such as ModSecurity can fail to detect novel attack patterns, depending on their rule sets and update frequency [4]. These shortcomings highlight the urgent need for more adaptive and intelligent detection mechanisms capable of identifying unknown threats without relying on predefined signatures.

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, we aim to address the limitations of existing detection systems by developing an ensemble-based anomaly detection model that can effectively detect zero-day web attacks without prior exposure to attack patterns. The model 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 model, various web attacks such as SQL injection (SQLi), Cross-Site Scripting (XSS), and Buffer Overflow [5] 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.

Our ensemble method differs from conventional ensemble approaches by combining and compressing latent features rather than relying on simple output aggregation (such as majority voting), significantly enhancing detection performance. 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 web-based services, the detection of web attacks, particularly zero-day exploits, has become a critical cybersecurity challenge [6]. There are two approaches to detect attacks: non-ML (heuristic-based) and machine learning-based approaches. Traditional heuristic approaches detect threats based on manually defined rules [7], but suffer from high false positive rates and are often ineffective against novel attack variants. 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 [8].

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 web attacks such as Structured Query Language Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). It is commonly implemented in signature-based systems, including Web Application Firewalls (WAFs). Supervised models are trained on labeled datasets containing examples of attack patterns alongside normal requests. For instance, a simple SQL injection attack pattern might include inputs like ' OR '1'='1' to manipulate authentication queries, while an XSS attack could involve scripts such as <script>alert('XSS')</script> injected into input fields. By learning these known signatures, supervised models can accurately classify incoming web requests that match documented attack behaviors. However, their dependence on predefined signatures renders them ineffective against zero-day attacks, which introduce novel patterns that are not represented in the training datasets [9].

The unsupervised approach, on the other hand, employs anomaly detection techniques to distinguish between normal and anomalous web activities. Unlike supervised methods, unsupervised models do not require prior knowledge of specific attack patterns, enabling them to identify previously unseen zero-day attacks [9,10]. By modeling the expected behavior of normal web traffic, these approaches can effectively flag deviations indicative of novel attack types, including obfuscated SQLi and XSS payloads, or unauthorized access attempts, making them particularly suitable for detecting evolving and sophisticated threats in dynamic web environments.

The semi-supervised approach lies between supervised and unsupervised methods and leverages exclusively normal web request data to train the model. By focusing on learning only legitimate traffic patterns, the model becomes capable of identifying anomalous behaviors that may signal new or obfuscated web attacks. Since semi-supervised methods eliminate the need for labeled malicious data, they offer a practical solution for early detection of emerging zero-day vulnerabilities while reducing dependency on frequent updates to labeled datasets [9].

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. [9], Ingham et al. [11], Sivri et al. [12], Jung et al. [13], Vartouni et al. [14], Ariu et al. [15], Liang et al. [16], Kuang et al. [17], Tang et al. [18], Indrasiri et al. [19], Gong et al. [20], Tekerek et al. [21], Jemal et al. [22], Alaoui et al. [23], Moarref et al. [24], Yatagha et al. [25], Katbi et al. [26], Tokmak et al. [27], Alqhwazi et al. [28], Thalji et al. [29], Yao et al. [30], Mohamed et al. [31], Shahid et al. [32], Bedi et al. [33], Milosevic et al. [34], Abdelkhalek et al. [35], Yuan et al. [36], Vorobyov et al. [37], Su et al. [38], and Silvestre et al. [39] are notable.

Pu et al. [9] proposed an unsupervised anomaly detection method that combines Sub-Space Clustering (SSC) with a One-Class Support Vector Machine (OCSVM) to detect cyber intrusions, including zero-day attacks, without requiring labeled data. Evaluated on the NSL-KDD dataset, their model achieved a detection rate of 89% with a false alarm rate of 8%, though its performance dropped on low-frequency attack types (e.g., R2L with 52% DR and 8.95% FAR). Additionally, the reliance on Sub-Space Clustering led to increased computational overhead, limiting real-time deployment.

Ingham et al. [11] proposed an anomaly detection system for web servers that models normal HTTP requests using Deterministic Finite Automata (DFA) induction. Their approach uses heuristics to reduce variability in request tokens before feeding them to the DFA learner, enabling the system to detect anomalous web requests without requiring labeled attack data. While effective in identifying various types of web attacks, the method suffers from scalability issues due to large DFA sizes and exhibits high false positive rates in complex environments (e.g., 21 false alarms/day on cs.unm.edu).

Sivri et al. [12] evaluated various machine learning and deep learning models on the CSIC 2012 dataset for web attack detection. They explored character-level representations of HTTP requests and applied classifiers such as LSTM, Convolutional Neural Network (CNN), Extreme Gradient Boosting (XGBoost), and Light Gradient Boosting Machine (LightGBM). For the CSIC 2012 portion of the dataset, the LSTM model achieved the highest performance, with an accuracy of 98.15%, F1-score of 98.16%, and false positive rate of approximately 0.008 (0.8%). Despite strong results, their models were trained in a supervised fashion using both normal and malicious samples, which limits adaptability to zero-day threats. Additionally, their best-performing models required extensive training time (e.g., over 5600 s for LSTM), making them less practical for real-time systems. In contrast, our unsupervised ensemble model operates solely on normal request data, detects zero-day attacks and achieves comparable accuracy with a significantly lower false positive rate of 0.2%, while also offering more efficient training.

Jung et al. [13] proposed a payload feature-based transfer learning (PF-TL) framework to address the problem of insufficient labeled training data in intrusion detection. Their approach enhances feature extraction by combining traditional signature-based features with text vectorization over both the header and payload, aiming to improve representation of domain-specific attack patterns. The transfer process optimizes latent feature sub-spaces between labeled source domains and unlabeled target domains using Principal Component

Analysis (PCA) and similarity-aware projection. They evaluated the method across public datasets including CSIC 2012, showing that payload feature-based transfer learning (PF-TL) significantly outperforms baseline transfer learning (HeTL) and non-transfer baselines, achieving 99.88% accuracy and 99.80% F1-score in detecting attacks like SQLi, XSS, and LDAP injection on the CSIC dataset. However, the approach depends on domain similarity assumptions and requires prior knowledge for effective source–target alignment, limiting its flexibility. Unlike our model, which operates fully unsupervised and does not require labeled data or domain pairing, PF-TL still relies on pre-labeled source domains and structured sub-space mapping.

Vartouni et al. [14] proposed an anomaly detection method using a stacked autoencoder (SAE) for feature extraction combined with an Isolation Forest classifier to detect web attacks in HTTP traffic. Features were constructed using a character-level n-gram model, and experiments were conducted on the CSIC 2010 dataset. Their best-performing configuration, using bigram features and SAE with Sigmoid activation and Adam optimizer, achieved 88.32% accuracy, 88.34% detection rate, and 80.79% precision, with specificity of 90.20% and an F1-score of 84.12%. While the approach effectively leveraged unsupervised learning, it suffered from high-dimensional feature vectors (e.g., 2572 features for bigrams), which may introduce computational overhead and noise. In contrast, our model achieves superior precision (99.99%) by combining latent representations from multiple autoencoders, reducing reliance on sparse token-based features and enhancing both efficiency and generalization.

Ariu et al. [15] proposed HMMPayl, an anomaly-based intrusion detection system that uses Hidden Markov Models (HMMs) to model HTTP payloads as sequences of bytes. Their approach offers the expressive power of n-gram analysis while mitigating its computational complexity by processing variable-length payloads through sliding windows and extracting sequence features efficiently. HMMPayl combines multiple HMMs using ensemble techniques like max, min, and mean rules to improve accuracy and robustness. It was evaluated on datasets containing real HTTP traffic and attacks such as SQL injection and XSS, achieving high AUC scores often above 0.97 for shell-code and polymorphic attacks. However, its effectiveness heavily depends on proper tuning of HMM parameters and the window size n, and the system struggles when payloads are short or similar to normal traffic, which can lead to false positives. Unlike our model, which uses high-level semantic tokenization and compresses multi-autoencoder latent representations, HMMPayl operates directly on raw bytes, making it sensitive to payload noise and potentially less adaptable to structural anomalies in web requests.

Liang et al. [16] proposed a deep learning approach for anomaly-based web attack detection using a two-stage model. First, two Recurrent Neural Networks (RNNs), one for the URL path and another for query parameters, are trained in an unsupervised manner using normal GET requests to learn request structure patterns. Then, a supervised Multilayer Perceptron (MLP) classifier is trained on the outputs of the RNNs to distinguish between normal and anomalous requests. Evaluated on a trimmed version of the CSIC 2010 dataset, their model with LSTM achieved 98.42% accuracy, 0.79% false alarm rate and 99.21% specificity, outperforming other approaches like SOM and C4.5. However, the model depends on labeled attack data during the second (supervised) phase, which may hinder generalization to zero-day attacks.

Kuang et al. [17] proposed DeepWAF, a deep learning-based Web Application Firewall that integrates multiple neural network architectures, including CNN, LSTM, CNN-LSTM, and LSTM-CNN, to detect malicious HTTP requests. Their models were evaluated on the CSIC 2010 dataset and achieved a detection rate of approximately 95% with a false alarm rate of around 2%. While DeepWAF demonstrated the effectiveness of deep learning in

web attack detection, its performance still resulted in a noticeable number of false positives, and the approach was trained in a supervised fashion using both normal and attack data. In contrast, our unsupervised ensemble model operates solely on normal traffic, enabling it to detect zero-day attacks while maintaining a significantly lower false positive rate of 0.2%, with higher detection precision and efficiency.

Tang et al. [18] proposed ZeroWall, an unsupervised deep learning model for detecting zero-day web attacks. The system operates alongside traditional Web Application Firewalls (WAFs) and uses an encoder–decoder Recurrent Neural Network to model benign HTTP requests as sequences in a "request language". It is trained exclusively on benign traffic filtered through an existing WAF, then detects anomalies in real time by assessing how well the model can reconstruct incoming requests. A poor reconstruction, measured using the BLEU score, signals potential malicious behavior. The approach was evaluated on eight large-scale real-world HTTP traces and achieved F1-scores above 0.98, outperforming both supervised and other unsupervised baselines.

Indrasiri et al. [19] proposed a robust ensemble learning model called Expandable Random Gradient Stacked Voting Classifier (ERG-SVC) to detect phishing URLs. Their approach utilizes a combination of seven classification algorithms, two ensemble methods, and a clustering algorithm. The system was evaluated on two large datasets with 73,575 and 100,000 URLs, incorporating techniques like heuristic thresholding, extensive feature engineering, and hyperparameter tuning. The ensemble model achieved a high prediction accuracy of 98.27%, and the authors also proposed a lightweight preprocessor for efficiency. However, this work focuses exclusively on phishing URL detection, not full HTTP request analysis, and relies on URL-based features—some of which depend on third-party services like WHOIS and PageRank.

Gong et al. [20] proposed a deep learning-based web attack detection system that introduces model uncertainty to address annotation errors in training data. Their model uses character-level CNNs combined with dropout-based Bayesian inference to quantify uncertainty in predictions. By measuring the variance of the model's output—rather than relying solely on Softmax confidence, they identified mislabeled training samples that could otherwise degrade detection accuracy. The system was evaluated on three datasets, including CSIC 2010, and demonstrated strong performance, with F1-scores up to 94.25%, 97.64% accuracy, 91.11% recall and 97.62% precision. However, their method depends on labeled attack and normal traffic for supervised training and focuses more on correcting label noise than detecting zero-day attacks.

Tekerek et al. [21] proposed a CNN-based deep learning model for anomaly detection in HTTP web traffic. Their system focused on identifying malicious patterns in URLs and payloads extracted from HTTP requests in the CSIC2010v2 dataset. The method uses bag-of-words (BoW) encoding to convert segmented URL and payload parameters into matrix representations, which are then used as grayscale input images to train a CNN. Evaluated over 400 epochs, their model achieved a best accuracy of 97.07%, F1-score of 97.51%, and a false positive rate (FPR) of 3.68%. While the approach effectively detects web attacks through CNN's automatic feature extraction, its reliance on BoW leads to large sparse matrices, and the model may be sensitive to input structure variations. In contrast, our method avoids token sparsity by applying semantic-level tokenization and learning compressed latent representations through an ensemble of autoencoders, achieving comparable detection accuracy with a significantly lower FPR of 0.2%.

Jemal et al. [22] introduced SWAF, a smart Web Application Firewall based on a 5-layer Convolutional Neural Network (CNN) trained using the CSIC 2010 dataset. Their system processes HTTP requests using an ASCII embedding method, which maps each character in the request to its ASCII code, avoiding issues with unseen tokens found in word or

character embedding. SWAF was evaluated using 5-fold cross-validation and achieved an accuracy of 98.1%, with a detection time of just 2.3 ms per request—a notable improvement over earlier CNN-based methods. While effective, the model relies entirely on labeled attack data for supervised training and does not target zero-day attacks directly.

Alaoui et al. [23] proposed a deep learning-based approach for web attack detection using Word2vec embeddings and a stacked generalization ensemble of LSTM networks. The system processes HTTP requests from the CSIC 2010 dataset, applying decoding, generalization, and tokenization to prepare input for embedding. Multiple Word2vec models with different training configurations were used to produce diverse word representations, which were fed into base LSTM models. A shallow neural network was then used as a meta-classifier to combine sub-model predictions. The best configuration of their stacked LSTM ensemble achieved 78.95% accuracy, 81.54% precision, 78.41% recall, and an F1-score of 77.57%. While the model benefits from ensemble diversity and word-level semantics, it remains a supervised approach reliant on labeled attack data.

Moarref et al. [24] proposed a character-level multichannel multilayer dilated Convolutional Neural Network (MC-MLDCNN) for detecting web attacks. Their model processes the entire HTTP request text and applies dilated CNNs across multiple channels to capture both short- and long-term dependencies in character sequences. This structure eliminates the need for handcrafted feature engineering, enhancing adaptability to novel attack patterns. Experiments conducted on the CSIC 2010 dataset demonstrated strong performance with 99.36% accuracy, 99.65% precision, 98.80% recall, and an F1-score of 99.22%. However, the approach is supervised and relies on both normal and labeled attack data, limiting its zero-day detection capability in comparison to unsupervised methods.

Yatagha et al. [25] proposed a hybrid anomaly detection framework combining Variational Autoencoders (VAEs), Long Short-Term Memory (LSTM) networks, and a One-Class SVM (OCSVM) to detect zero-day anomalies in cyber–physical systems. The model is trained exclusively on normal time-series data and leverages both reconstruction error and latent space deviation to flag contextual and temporal anomalies. To ensure continual adaptation without catastrophic forgetting, the authors introduced an Adaptive Loss Weight Adjustment Algorithm (ALWAA) aligned with ISO/IEC 42001:2023 standards. Deployed in a real-time industrial setting on a Raspberry Pi 400, the system successfully detected subtle anomalies such as valve failures and pump delays, outperforming baseline methods. While effective for low-dimensional physical sensor data, the model is not tailored for web-based request structures like HTTP traffic.

Katbi et al. [26] proposed IDSVDD, a one-class anomaly detection framework tailored for IoT security, which combines Deep Support Vector Data Description (SVDD) with an interpolated adversarial autoencoder. The model constructs a convex and regularized latent space by enforcing adversarial interpolation constraints, enabling better separation between normal and anomalous behaviors. Trained only on benign data, IDSVDD learns a compact hypersphere that tightly encloses normal samples while rejecting outliers. It was evaluated on multiple benchmark IoT datasets, demonstrating strong zero-day detection performance with minimal computational overhead suitable for resource-constrained devices like Raspberry Pi. However, the system is designed for sensor-driven IoT data and does not address the structure, complexity, or variability of HTTP web requests.

Tokmak et al. [27] proposed a hybrid deep learning framework that combines stacked autoencoders (SAEs) for unsupervised feature extraction with a Long Short-Term Memory (LSTM) classifier for detecting zero-day cyber threats. The model was trained and evaluated using the UGRansome dataset, which includes labeled samples of signature, synthetic signature, and anomaly attacks. The SAE extracts a low-dimensional latent feature set from raw input, which is then passed to the LSTM for supervised temporal classification. The

proposed system achieved a high accuracy of 98.49%, with precision, recall, and F1-score all around 0.985, demonstrating strong generalization to diverse attack categories. However, its reliance on labeled attack data during LSTM training limits its applicability to true zero-day detection. In contrast, our proposed ensemble model operates in a fully unsupervised setting trained solely on normal web requests, enabling it to detect zero-day web attacks without prior exposure to malicious samples, while maintaining a lower false positive rate (0.2%) and higher precision (99.99%).

Alqhwazi et al. [28] proposed a deep learning architecture for detecting SQL injection attacks using a Recurrent Neural Network (RNN) autoencoder. The model was trained on a public Kaggle SQL injection dataset, consisting of over 30,000 SQL queries labeled as benign or malicious. Their architecture includes an autoencoder for dimensionality reduction followed by an LSTM classifier for binary classification. The proposed system achieved 94% accuracy, 95% precision, 90% recall, and an F1-score of 92%, outperforming traditional machine learning models such as SVM, decision tree, random forest, and logistic regression. However, the approach is tailored specifically for SQL injection patterns and depends on labeled data for supervised classification. In contrast, our ensemble model is designed for detecting a wide range of web-based attacks (e.g., XSS, SQLi, Buffer Overflow) using only normal web requests.

Thalji et al. [29] proposed AE-Net, a novel autoencoder-based deep feature extraction approach for detecting SQL injection (SQLi) attacks. The model uses an unsupervised autoencoder to extract high-level semantic features from SQL queries, which are then fed into traditional machine learning models for classification. Evaluated on a public dataset of 46,392 labeled SQL queries, the framework was benchmarked using k-fold cross-validation, with XGBoost achieving the highest performance—an accuracy of 99%, outperforming BoW and TF-IDF-based feature extraction techniques. While the AE-Net successfully enhances feature representation quality, the detection model still depends on labeled attack data and is limited to SQLi detection only, not general web attacks.

Yao et al. [30] 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.

Mohamed et al. [31] proposed a deep learning-based multi-class intrusion detection system that automatically classifies various types of web attacks using LSTM, Bi-LSTM, CNN, and RNN models. Their system uses tokenization and sequence padding to convert HTTP request payloads into numerical input for deep models, avoiding traditional feature engineering. The model was evaluated on three benchmark datasets, CSIC 2012, HTTPPARAM, and ECML-PKDD, demonstrating high classification accuracy. The CNN model achieved 99.28% accuracy, 99.18% precision, 99.28% recall, and 99.22% F1-score on CSIC-2012, while Bi-LSTM achieved 99.66% on HTTPPARAM and 90.6% on ECML-PKDD. While this approach achieves excellent detection and multi-class labeling performance, it requires labeled attack data for supervised training and may struggle with detecting zero-day attacks without retraining. In contrast, our proposed model operates in a fully unsupervised manner using only normal traffic for training, and it leverages latent feature compression across multiple autoencoders to detect unknown web attacks with higher precision (99.99%) and lower false positive rates (0.2%).

Shahid et al. [32] proposed a hybrid deep learning framework for real-time web attack detection and attacker profiling, which nests a Convolutional Neural Network (CNN)

classifier with a custom Cookie Analysis Engine (CAE). The CNN is trained on HTTP request parameters, while the CAE performs integrity and sanitization checks on cookies to enhance profiling and reduce false positives. Their model achieves high accuracy (99.94% on their own dataset and 98.73% on the CSIC 2010 dataset) and claims to reduce classifier invocations through intelligent profiling. However, the framework's real-time efficiency is largely dependent on the CAE's effectiveness, and the approach still triggers deep learning analysis for many cases, raising concerns about scalability.

Bedi et al. [33] addressed the well-known class imbalance problem in intrusion detection systems (IDSs), particularly in datasets like NSL-KDD where classes such as Remote to Local (R2L) and User to Root (U2R) are underrepresented. To overcome the limitations of traditional class balancing methods—such as oversampling (which may cause overfitting), undersampling (which may discard valuable data), and SMOTE (which may introduce synthetic noise)—the authors proposed Siam-IDS, a novel anomaly detection framework based on Siamese Neural Networks. By leveraging few-shot learning, Siam-IDS learns similarity-based representations and avoids reliance on class distribution altogether. Experimental results demonstrated that this approach significantly improved the recall values of minority attack classes compared to deep neural network (DNN) and Convolutional Neural Network (CNN) baselines, highlighting the potential of similarity-based learning in imbalanced intrusion scenarios.

Milosevic et al. [34] addressed the challenge of extreme class imbalance in network intrusion detection using deep learning. Rather than employing typical resampling techniques, they retained the original imbalanced distribution of the CICIDS-2017 and CICIDS-2018 datasets, which include several classes with extremely low instance counts. Their approach focuses on designing a deep neural network (DNN) architecture and leveraging statistical and correlation-based feature selection methods to enhance the model's sensitivity to minority classes. Notably, their analysis reveals that even classes with as few as three instances can be detected accurately when relevant coarse-grained features are preserved. This work demonstrates that careful feature selection and model tuning can partially mitigate the effects of imbalance without modifying the dataset distribution.

Another recent study by Abdelkhalek et al. [35] directly addressed the class imbalance problem in intrusion detection by proposing a hybrid resampling approach combining Adaptive Synthetic Sampling (ADASYN) with Tomek Links undersampling. Using the NSL-KDD dataset, the authors first applied ADASYN to generate synthetic minority class samples based on difficulty of learning, followed by Tomek Links to remove overlapping or noisy instances near class boundaries. This strategy was shown to significantly enhance the detection of minority attack types (such as U2R and R2L), which are typically underrepresented in intrusion datasets. Their method was tested across multiple deep learning models including MLP, DNN, CNN, and CNN-BLSTM, achieving a maximum binary classification accuracy of 99.8% and a multi-class classification accuracy of 99.98%, clearly demonstrating the impact of well-designed data balancing techniques in improving NIDS performance.

Apart from machine learning techniques, various heuristic-based approaches have been proposed to detect web-based attacks, such as Yuan et al. [36], 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 (OODBEs) 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. Its effectiveness is also limited when dealing with complex semantic

constructs in modern web applications, which may hinder accurate transformation and lead to missed vulnerabilities.

Vorobyov et al. [37] 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. However, the model relies heavily on the completeness of the training set. This can result in false positives when encountering legitimate but unseen query patterns, and performance bottlenecks when comparing incoming queries against large allowlists at runtime.

Su et al. [38] 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. [39] 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 inconsistencies as SQLi vulnerabilities. This method offers formal guarantees and reduces false positives by leveraging strong type checking rather than heuristics or machine learning.

A recurring limitation among the reviewed studies is the lack of emphasis on one of the most critical evaluation metrics in real-world deployment: the false positive rate (FPR), which quantifies how often benign web requests are incorrectly flagged as malicious. Despite high accuracy or F1-scores, many models fail to report or sufficiently optimize FPR, leading to increased friction for legitimate users and reduced system trustworthiness. This omission makes it difficult to assess a model's practical effectiveness, especially for zero-day detection scenarios where avoiding false alarms is crucial. In contrast, our proposed model explicitly prioritizes FPR and achieves a significantly lower rate of 0.2%, outperforming even those models that do report this metric. Additionally, unlike many supervised approaches that rely on labeled attack data, our method is trained solely on normal traffic, ensuring greater generalization to unseen attacks while maintaining high precision (99.99%) and efficient inference time. These qualities collectively position our model as a more practical and robust solution for real-world web attack detection.

4. Proposed Model

This section presents the proposed model for detecting zero-day web attacks. The detection process begins with the preprocessing 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. As illustrated in Figure 1, the training pipeline begins with tokenizing normal web requests using a dictionary-based approach, which are then converted into numerical sequences and fed into an ensemble of autoencoders. Figure 2 emphasizes the change during the testing phase, where both normal and malicious requests are fed into the trained model, highlighting the anomaly detection capability.

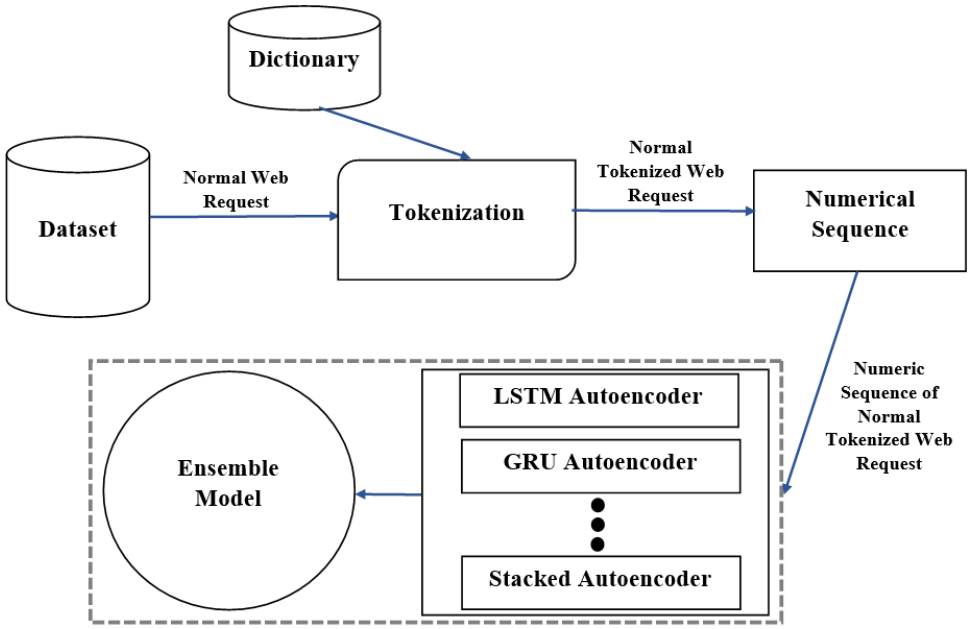

Figure 1. The proposed model in the training phase.

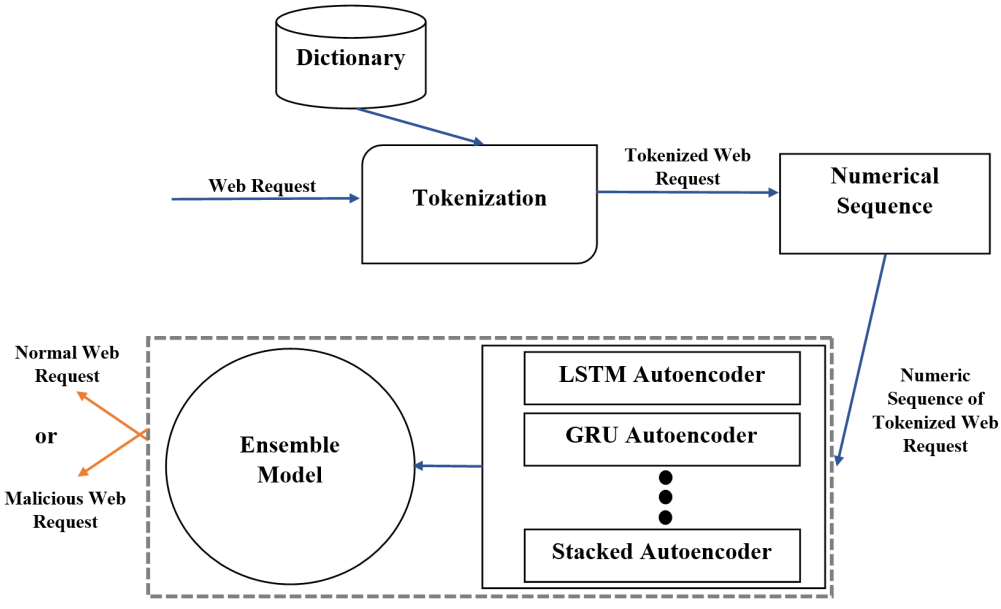

Figure 2. The proposed model in the test phase.

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 [40]. 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.

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 [41]. 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 [42].

Unlike character-level tokenization [14], which treats each character individually and can introduce unnecessary complexity, or n-gram-based methods [15] that can lead to very high dimensionality and computational overhead, our approach focuses on categorizing tokens into predefined classes based on character composition (e.g., *Alpha*, *AlphaNum*, *CapitalAlpha*, and *SpecialChar*). This categorization simplifies input data and reduces dimensionality, directly enhancing computational efficiency.

In contrast, embedding-based methods like Word2Vec [23] or transformer-based tokenization [11] encode semantic relationships between tokens but might not efficiently capture the syntactic patterns critical for identifying subtle anomalies in structured web requests. Our dictionary-based tokenization, however, explicitly preserves critical structural and syntactic information, which is essential for accurately identifying anomalous web requests.

To ensure consistent data representation, the preprocessing pipeline standardizes normal web requests through a dictionary-based tokenization approach [16]. This process involves segmenting each request at the word level using tools such as Python's WordPunctTokenizer [43]. The resulting structured pattern is subsequently utilized as input to train 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. Figure 3 provides a visual example of how the predefined dictionary maps raw tokens to their corresponding semantic categories during the tokenization process. For instance, 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.

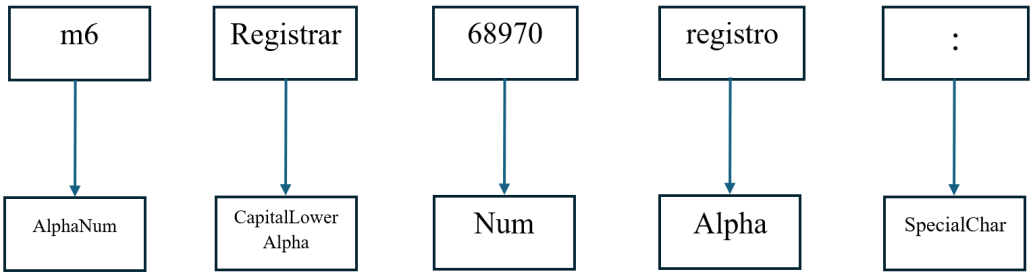

Figure 3. A tokenized request.

While the current tokenization approach effectively handles the structure of the CSIC dataset, its generalization to other datasets is limited due to its dependency on a static token mapping dictionary. Future work will explore adaptive tokenization using LLMs to support broader request formats.

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 [44,45]. The tokenized text is converted into a structured numerical format suitable for input into the neural network. Figure 4 visualizes how the structured tokens from the previous step are translated into numerical values through a predefined mapping dictionary. Each semantic token class (e.g., AlphaNum, Num, SpecialChar) is assigned a unique integer identifier, enabling direct input into the neural network. This process ensures consistency across inputs, allowing the model to recognize semantic patterns in a numerically efficient format. The mapping also supports input standardization via padding, which mitigates the risk of misalignment or irregular dimensions during training and inference.

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 to standardize input dimensions. This step enhances the model's ability to analyze and recognize patterns within the data, ultimately improving detection accuracy and performance.

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. Each sub-model is trained to independently learn compressed representations of normal web requests. The LSTM and GRU autoencoders specialize in capturing sequential dependencies, while the stacked autoencoder focuses on global feature abstraction. After encoding and decoding, the latent representations from all three models are concatenated into a unified vector. This

combined representation captures diverse patterns, helping improve detection accuracy. A dense compression layer is then applied to reduce the dimensionality of the concatenated vector, aligning it with the original input size while retaining the most relevant features. This step is critical for reducing computational overhead and minimizing redundancy, thereby improving both model scalability and interpretability. To mitigate the typical computational overhead associated with ensemble models, our approach strategically employs three lightweight sub-models—LSTM, GRU, and stacked autoencoder—each selected for its balance between performance and efficiency. Unlike complex or deeply layered architectures, these models are compact and require fewer parameters and training iterations. Additionally, during inference, all sub-models run in parallel, which reduces latency and makes the system well-suited for real-time or large-scale deployment. To further enhance efficiency, the outputs of all sub-models are concatenated and compressed using a dense layer, significantly reducing the dimensionality of the final feature representation before classification. This design not only preserves high detection accuracy but also ensures low memory usage and fast runtime, thereby directly addressing the scalability challenge often associated with ensemble-based approaches.

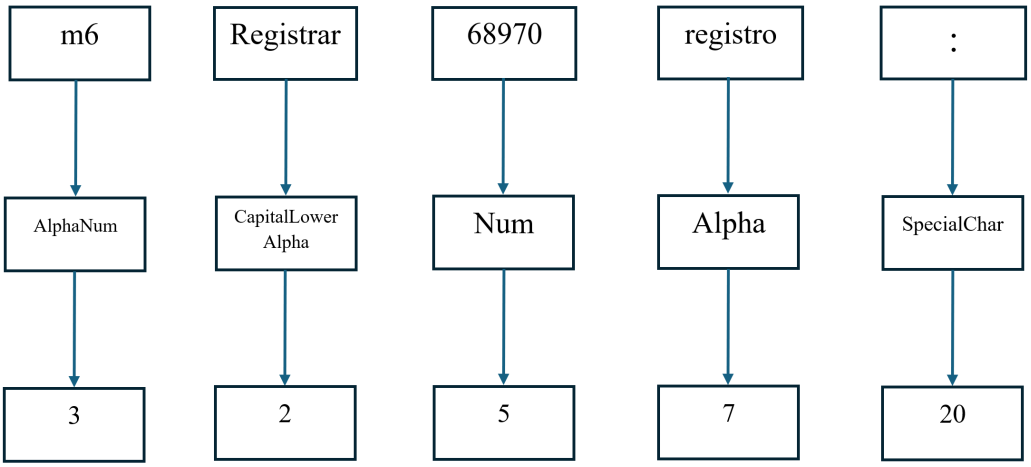

Figure 4. Mapping words to numbers.

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 [46,47]. 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 are 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.

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.

The compression of concatenated latent features serves two main purposes: (1) it aligns the dimensionality of the output with that of the original input, enabling direct interpretability and compatibility for anomaly scoring, and (2) it eliminates redundant or low-information features across the sub-models, thus reducing computational overhead and enhancing inference efficiency. This step is particularly critical when deploying the model in real-time systems, where performance and resource constraints must be considered.

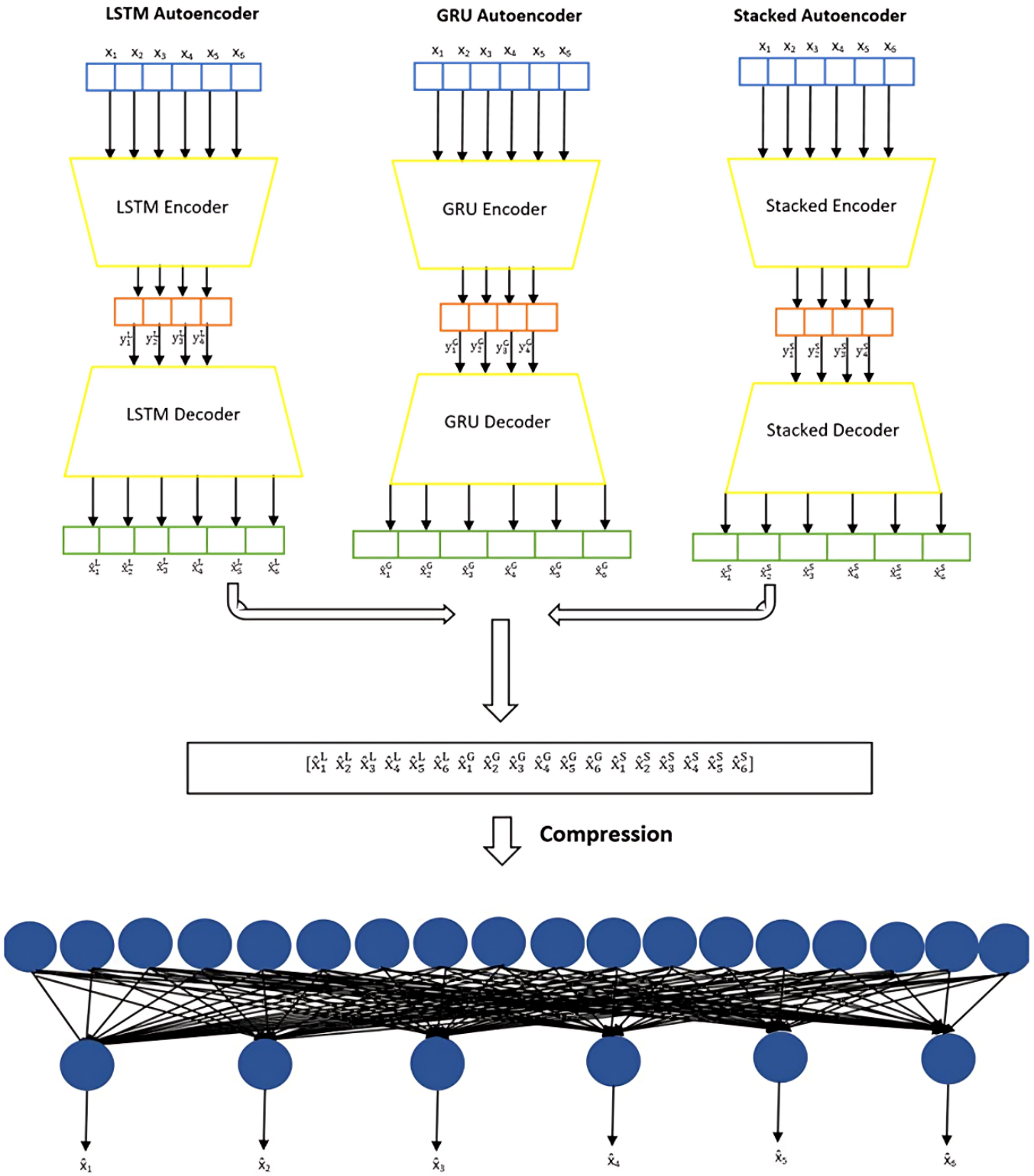

Figure 5. Architecture of the ensemble model showing LSTM, GRU, and stacked autoencoder branches, followed by latent space concatenation and feature compression.

It is important to explicitly address how our proposed model handles the challenge of class imbalance, which is frequently problematic in anomaly detection scenarios. Unlike traditional supervised learning models, our approach uses an unsupervised anomaly detection paradigm that relies exclusively on normal web request data for training. Specifically, we train the model solely on 80% of the 6492 normal samples with 5193 from the CSIC

2012 dataset, ensuring that the model learns only legitimate web request patterns and behaviors. As a result, the imbalance issue—typically characterized by an insufficient number of malicious examples for learning—is effectively circumvented during training, as our model does not require exposure to malicious patterns to establish baseline behavior.

Furthermore, we carefully consider class imbalance in the evaluation phase. Our test set includes a notably imbalanced distribution: 1299 normal requests versus 50,176 malicious requests. Despite this imbalance, our evaluation explicitly incorporates a diverse range of metrics—accuracy, precision, recall, specificity, F1-score, and particularly the false positive rate (FPR)—to ensure that the model's performance is assessed comprehensively and fairly. By explicitly including metrics like the FPR and recall, we precisely capture how well the model avoids false alarms and missed detections, even in the presence of extreme class imbalance. These metrics directly reflect the model's practical viability for real-world deployment, where imbalance is inevitable but should not compromise detection reliability.

In short, our unsupervised training approach inherently avoids the class imbalance issue during model training, while our thorough evaluation methodology ensures the model's robustness and practical effectiveness under realistic, highly imbalanced testing conditions.

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 the 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

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

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 each web request. To determine the optimal threshold for distinguishing between benign and malicious requests, we analyzed the distribution of MAE scores across normal training requests, as visualized in Figure 6. Based on this distribution, we observed that MAE values greater than approximately 4 were uncommon among normal requests, suggesting that these higher values represent potential anomalies. Consequently, we selected a threshold of 4.09, just above this observed point, as our decision boundary. Additional experiments

with alternative threshold values around 4 confirmed that this selection offered a balanced trade-off between sensitivity to malicious requests and avoiding excessive false positives. Although this approach is heuristic, future work could systematically evaluate the threshold using techniques such as percentile-based analysis or automated hyperparameter tuning methods to further refine the detection performance.

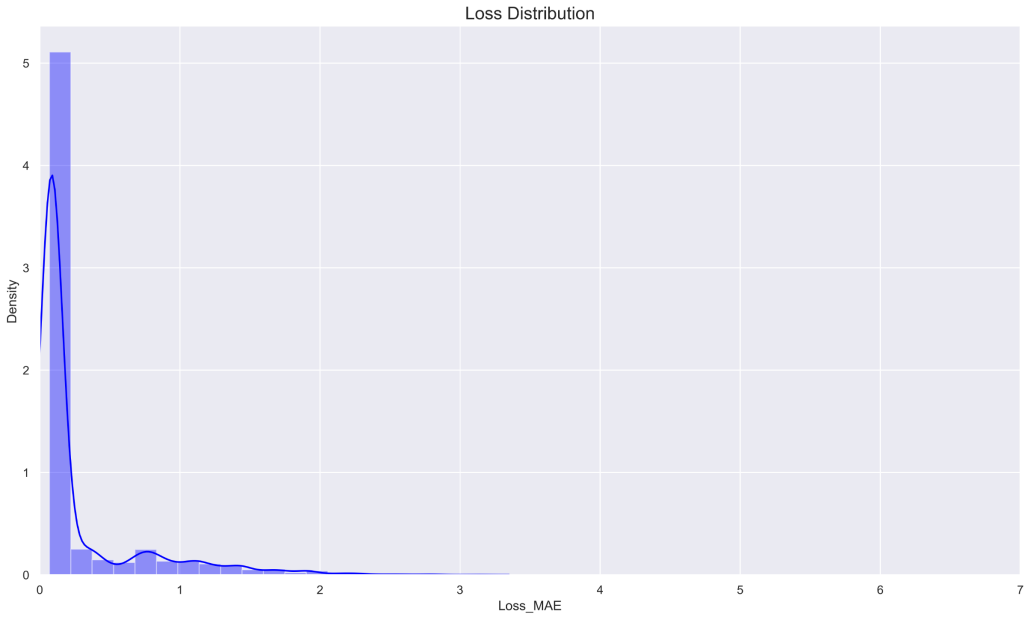

Figure 6. Density diagram according to the MAE of the proposed model.

5.1. Data Collection

Previous research on detecting malicious and zero-day web requests has primarily relied on two well-established datasets: CSIC [48] and HTTPPARAMS [48]. 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 [49], which employs Convolutional Neural Networks (CNNs), 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, the Nadam optimizer, and evaluated based on accuracy metrics.

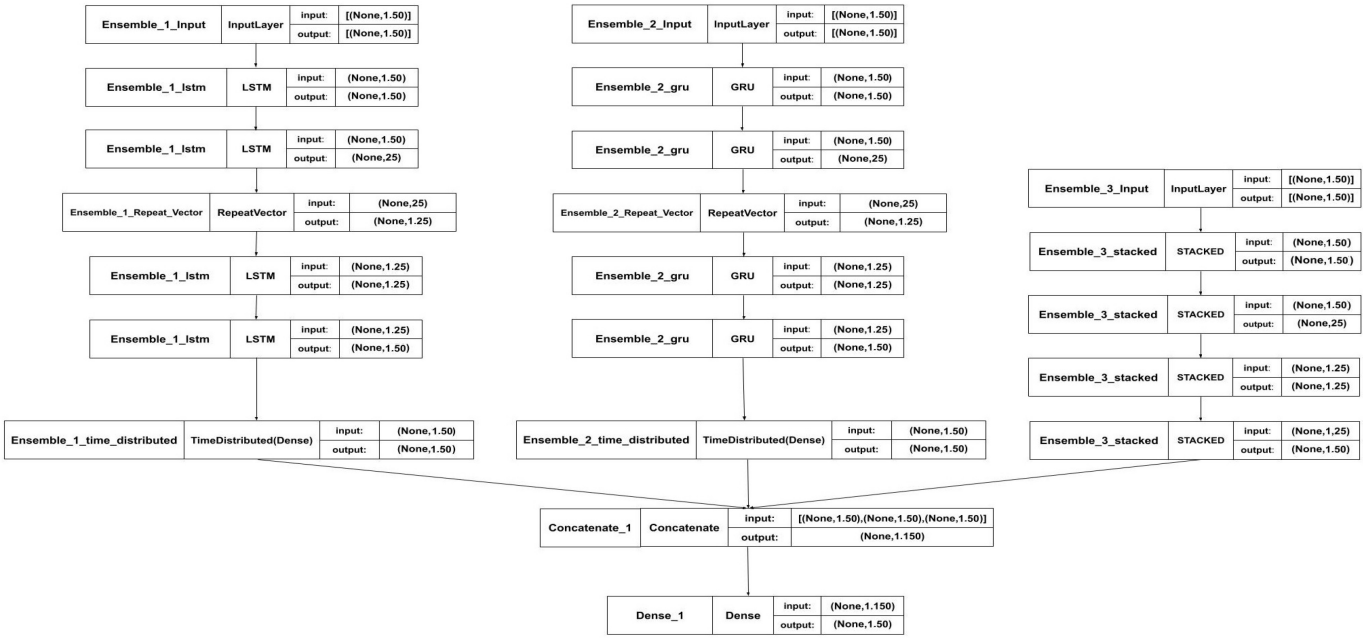

Figure 7. The architectural structure of the proposed ensemble model. Each branch represents one of the three autoencoder sub-models—LSTM, GRU, and stacked autoencoder with their respective layer configurations. The outputs from these branches are concatenated and passed through a compression dense layer to form a unified representation used for anomaly detection.

The specific number of layers and units was primarily chosen to match the dimensionality of our input features (50 features). Additional experiments with alternative configurations (e.g., different layer sizes) confirmed that this selection provided the optimal balance between accuracy, computational efficiency, and effective representation of the structured input data.

Figure 8 shows the learning behavior of the proposed model across 90 epochs. Both the training and validation MAE decrease rapidly during early epochs and converge to low, stable values, indicating effective learning of the normal request distribution. The gap between the training and validation curves remains small throughout, suggesting minimal overfitting. This result confirms that the ensemble model generalizes well to unseen data while preserving low reconstruction error, which is critical for robust anomaly detection.

Following the detection phase, the Mean Absolute Error (MAE) for each web request is computed and compared against the predefined threshold. Figure 9 provides a detailed view of how the Mean Absolute Error (MAE) for each individual request compares to the detection threshold. The X-axis represents the sequence of requests processed during evaluation, while the Y-axis shows the MAE score on a logarithmic scale. Blue dots above the red threshold line indicate detected anomalies. This visualization demonstrates the model's ability to sharply distinguish between normal and malicious traffic, validating the appropriateness of the selected MAE threshold (4.09) for anomaly detection.

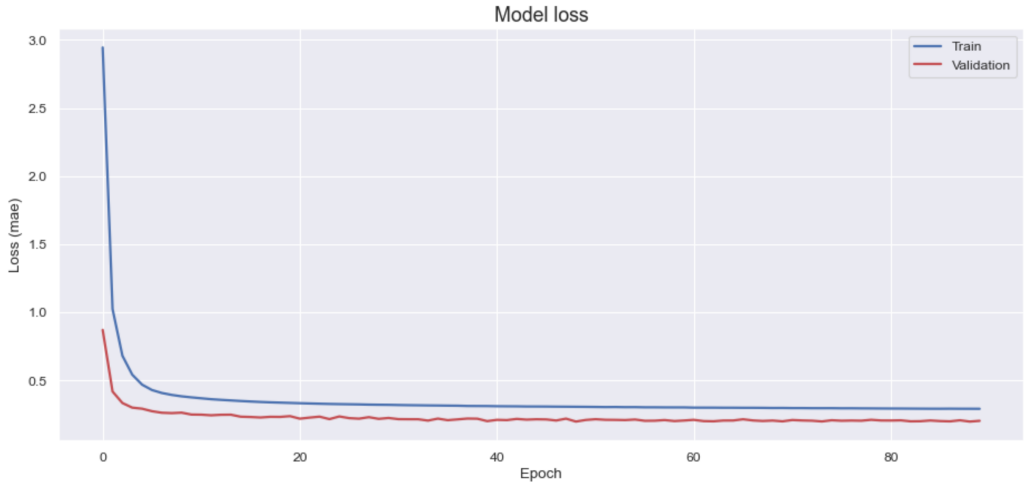

Figure 8. Training and validation process of the proposed model based on MAE and number of epochs.

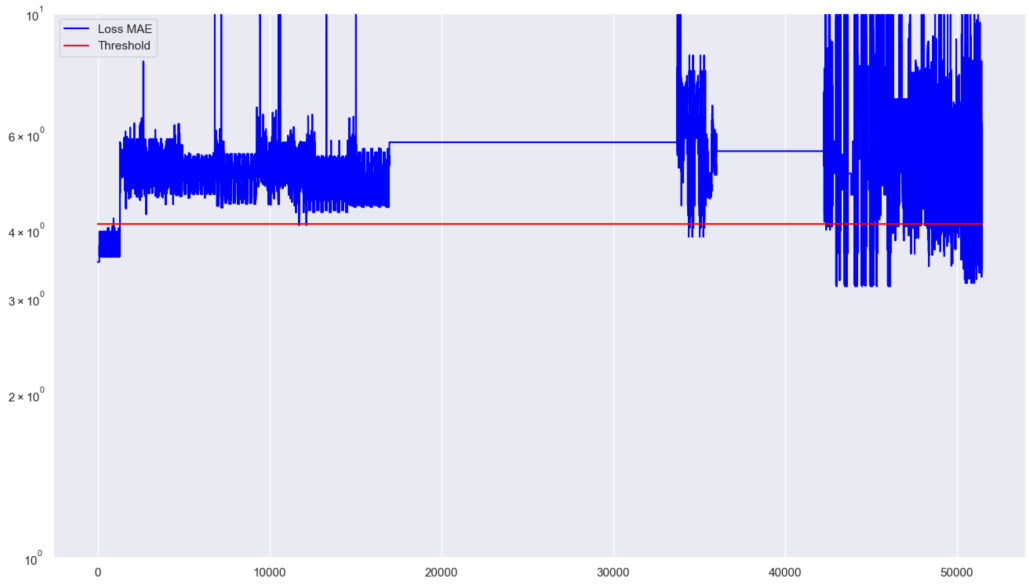

Figure 9. Comparison of the calculated MAE value for each web request with the threshold value.

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 quantify the difference between predicted and actual values, providing an effective measure for anomaly detection. The training phase required approximately 20 s, while the test phase was completed in 5 s. 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 1299 normal requests as malicious—an undesirable outcome in real-world scenarios. Incorporating the stacked autoencoder into the ensemble mitigates this issue by reducing false positives, albeit at the expense of slightly lower accuracy and recall when used independently. Table 3 and Figure 10 provide a detailed comparative analysis of the performance metrics for each individual sub-model—LSTM, GRU, and stacked autoencoder—as well as the overall ensemble model. The ensemble model significantly outperforms all individual components in terms of accuracy (97.58%), recall (97.52%), and F1-score (98.74%), and exhibits a notably low false positive rate of just 0.2%. This improvement reflects the ensemble's ability to capture diverse latent representations and mitigate the weaknesses of standalone models. The bar chart in Figure 10 visually reinforces these findings, showing that the proposed model maintains high precision and recall simultaneously—indicating a balanced and effective detection mechanism suitable for real-world deployment.

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. The bar chart in Figure 11 visually reinforces these findings, illustrating that the proposed model maintains high precision, recall and low FPR simultaneously—showing a balanced among metrics and effective detection mechanism suitable for real-world deployment. In the figure, models that utilized the CSIC2012 dataset are highlighted in red.

Table 2. Performance metrics used for the proposed model.

Metric	Definition/Calculation	Value
Total (T)	Total number of requests	51,473
Correct Predictions (CPs)	Number of correctly predicted requests	50,230
Negatives	Normal requests	1299
Positives	Malicious requests	50,174
True Negatives (TNs)	Normal requests correctly predicted as normal	1296
False Positives (FPs)	Normal requests incorrectly predicted as malicious	3
False Negatives (FNs)	Malicious requests incorrectly predicted as normal	1240
True Positives (TPs)	Malicious requests correctly predicted as malicious	48,934
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

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%**

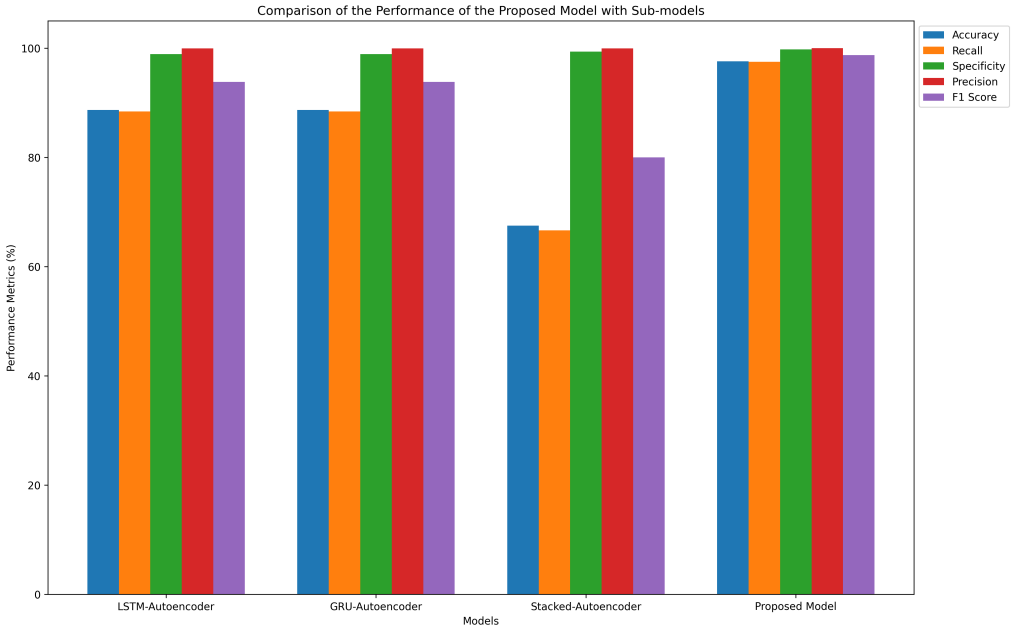

Figure 10. Comparison of the performance of the proposed model with the sub-models in the form of a bar chart.

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. [12]	98.15%	98.15%	-	98.20%	0.8%	98.16%
Jung et al. [13]	99.88%	-	-	-	-	99.80%
Vartouni et al. [14]	88.32%	88.34%	90.20%	80.79%	-	84.12%
Liang et al. [16]	98.42%	-	99.21%	-	0.7%	-
Kuang et al. [17]	96%	95%	-	96%	2%	-
Gong et al. [20]	97.64%	91.11%	-	97.62%	-	94.25%
Tekerek et al. [21]	97.07%	97.59%	3.68%	97.43%	3.68%	97.51%
Jemal et al. [22]	98.1%	-	-	-	-	-
Alaoui et al. [23]	78.95%	78.41%	-	81.54%	-	77.57%
Moarref et al. [24]	99.36%	98.80%	-	99.65%	-	99.22%
Mohamed et al. [31]	99.66%	99.28%	-	99.18%	-	99.22%
Shahid et al. [32]	98.73%	98.87%	98.33%	99.41%	1.67%	99.13%
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. Prior works include those by Sivri et al. [12], Jung et al. [13], Vartouni et al. [14], Liang et al. [16], Kuang et al. [17], Gong et al. [20], Tekerek et al. [21], Jemal et al. [22], Alaoui et al. [23], Mohamed et al. [31], Shahid et al. [32], and Moarref et al. [24].

The primary comparison focuses on studies that have employed the CSIC2012 dataset [12,13,31], as they provide the most directly comparable benchmark. However, to offer a broader perspective, we also include studies based on the CSIC2010 dataset [14,16,17,20–24,32]. 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.

6. Discussion

The proposed ensemble model demonstrates strong performance across all evaluation metrics, particularly in terms of the false positive rate (FPR). Achieving the lowest FPR compared to related works, the model highlights its capability to accurately distinguish between normal and malicious web requests. Maintaining a low FPR is critical in real-world web security applications to avoid blocking legitimate user activities and to preserve usability.

6.1. Effectiveness and Error Correlation in the Ensemble Model

Our ensemble approach integrates three complementary sub-models: LSTM autoencoder, GRU autoencoder, and stacked autoencoder. Each of these contributes uniquely to the ensemble:

- **LSTM Autoencoder**: Excels at capturing complex, long-term sequential dependencies in web requests.
- **GRU Autoencoder**: Offers computational efficiency alongside robust recognition of sequential patterns, making it well-suited for real-time scenarios.
- **Stacked autoencoder**: Specializes in extracting compact and representative latent features, aiding in subtle anomaly detection and dimensionality reduction.

An essential strength of the ensemble is its capability to mitigate correlated errors. Individual autoencoders often produce detection errors due to specific limitations—LSTM and GRU autoencoders may generate false positives due to their sensitivity to sequence complexity, while the stacked autoencoder may produce false negatives when anomalies present subtle deviations. By concatenating and compressing latent representations from multiple sub-models, our ensemble significantly reduces the correlation of errors across models. Specifically, cases misclassified by one sub-model often receive correct classifications by others, thereby collectively enhancing detection robustness. Future analyses could systematically quantify the correlation among sub-model errors using metrics such as Cohen's Kappa or correlation matrices, providing deeper insights into ensemble effectiveness.

Additionally, our advanced tokenization approach ensures a consistent and structured representation of web requests, which notably improves anomaly detection capabilities compared to conventional tokenization methods, as confirmed by our evaluation metrics. The explicit consideration and optimization of the false positive rate further differentiates our approach, addressing practical operational challenges often overlooked in the literature.

6.2. Generalizability, Adversarial Robustness, and Practical Challenges

Generalizability across different web domains remains a significant challenge. Our current tokenization and feature extraction pipeline was tailored specifically to standardize requests from the CSIC 2012 dataset, limiting its immediate applicability to more diverse real-world web requests. Adapting the tokenizer to handle varied formats found in datasets such as FWAF and HTTPParams [50] or real-world environments is an essential step toward broader generalization. Future research could significantly benefit from adaptive tokenization techniques leveraging Large Language Models (LLMs) and prompt engineering, enhancing the model's robustness across diverse and evolving web request structures.

The robustness of anomaly detection models against adversarial examples and obfuscation attacks is another crucial consideration that was not explicitly evaluated in our

current work. Attackers often employ sophisticated evasion techniques, including payload obfuscation or data drift—changes in web request patterns over time—to bypass detection mechanisms. To address these challenges, future research should explicitly test our model against adversarial and obfuscated web attacks, applying adversarial training or robustness testing methods to ensure resilience in adversarial scenarios. Moreover, continuous learning strategies could be adopted to manage data drift, dynamically adapting model parameters as web request patterns evolve.

Another limitation is the reliance on annotated benchmark datasets such as CSIC 2012. Although widely used for reproducibility and benchmarking, they do not fully replicate the complexity and variability found in live web systems. Further research should thus validate the model's performance within realistic simulated environments or operational settings, which could reveal insights about its practical robustness, scalability, and performance under real-time constraints.

While the current ensemble design achieves a balance between accuracy and efficiency, the scalability of such models in high-throughput or resource-constrained environments remains a practical challenge. Future research should explore optimization techniques such as model pruning, knowledge distillation, and quantization to further reduce inference time and memory consumption. Additionally, adaptive ensemble techniques where only a subset of sub-models are activated based on the input request's complexity could offer a dynamic trade-off between speed and accuracy.

Lastly, the current model focuses on anomaly detection without explicit categorization of attack types (e.g., SQL injection, Cross-Site Scripting, Buffer Overflow). Integrating an explicit attack-type classification mechanism would further enhance the practical utility and forensic capabilities of the proposed model, providing more actionable insights for cybersecurity professionals.

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.

Practical deployment within real-world security frameworks, such as Web Application Firewalls (WAFs) and real-time Intrusion Detection and Prevention Systems (IDS/IPS), is feasible given the computational efficiency of the proposed ensemble approach. Specifically, the pre-trained ensemble model can be integrated as a detection engine within WAF modules or IDS/IPS components, processing web requests in real time to promptly identify anomalous behavior based on reconstruction errors. To optimize performance in real-time scenarios, further efforts should explore model quantization, pruning, or efficient inference methods to ensure minimal latency without compromising detection accuracy.

8. Future Work

One potential direction for future research involves enhancing the tokenization and feature extraction process [51] across diverse web attack datasets. This improvement can be achieved through the application of Generative AI, leveraging Large Language Models (LLMs) [52,53]. Specifically, prompt engineering [4,54] can be employed to construct a structured prompt that systematically guides the LLM in preprocessing each dataset sample. To achieve this concretely, the following structured roadmap will be adopted:

In Phase 1, an exploratory pilot study will be conducted using a representative dataset such as HTTPParams [50]. The goal is to develop and evaluate initial prompt engineering strategies that leverage few-shot learning to guide Large Language Models (LLMs) in generating dataset-specific tokenization rules. During this phase, the performance of the LLM-generated tokenization will be assessed based on consistency with human-crafted rules, semantic accuracy, and overall computational efficiency.

In Phase 2, the experiments will be expanded by incorporating additional datasets, including FWAF and real-world HTTP traffic logs. This phase will focus on systematically comparing LLM-generated tokenization rules to those derived manually. Key performance indicators to be analyzed include generalizability across formats, robustness to variations in input structure, and the computational overhead introduced during preprocessing.

In Phase 3, the aim will be to fully automate the preprocessing pipeline. LLMs will be used to dynamically generate customized preprocessing scripts [55] based on structured prompts. This phase will focus on evaluating the reliability and consistency of the generated scripts, as well as measuring their runtime efficiency and the downstream impact on anomaly detection accuracy after LLM-driven preprocessing.

In Phase 4, 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) [56,57], 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 the adoption of LLMs for tokenization and preprocessing, potential challenges such as robustness against adversarial inputs, model hallucinations, and inconsistent outputs must be considered. Future research should thus include explicit adversarial robustness evaluations and validation protocols to assess and ensure reliability. Moreover, structured reasoning strategies inspired by recent frameworks such as VulnSage [58], a

framework leveraging structured reasoning strategies such as Chain-of-Thought and Think and Verify to improve zero-shot vulnerability detection in software systems.

Author Contributions: Methodology, V.B.; Software, V.B.; Validation, V.B.; Resources, V.B.; Writing—original draft, V.B.; Writing—review & editing, H.R.F.; Visualization, V.B.; Supervision, H.R.F.; Project administration, H.R.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

References

1. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; Tawalbeh, L. Zero-day attack detection: A systematic literature review. *Artif. Intell. Rev.* **2023**, *56*, 10733–10811. [CrossRef]
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. [CrossRef] [PubMed]
3. Hannousse, A.; Yahiouche, S.; Nait-Hamoud, M.C. Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey. *Comput. Sci. Rev.* **2024**, *52*, 100634. [CrossRef]
4. 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. [CrossRef]
5. Yang, J.; Chen, Y.L.; Por, L.Y.; Ku, C.S. A systematic literature review of information security in chatbots. *Appl. Sci.* **2023**, *13*, 6355. [CrossRef]
6. Calzavara, S.; Conti, M.; Focardi, R.; Rabitti, A.; Tolomei, G. Machine learning for web vulnerability detection: the case of cross-site request forgery. *IEEE Secur. Priv.* **2020**, *18*, 8–16. [CrossRef]
7. 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), 28–29 November 2024 ; Volume 1; pp. 450–455.
8. Li, Z.; Zhu, Y.; Van Leeuwen, M. A survey on explainable anomaly detection. *ACM Trans. Knowl. Discov. Data* **2023**, *18*, 1–54. [CrossRef]
9. Pu, G.; Wang, L.; Shen, J.; Dong, F. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Sci. Technol.* **2020**, *26*, 146–153. [CrossRef]
10. 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. *Int. J. Web Grid Serv.* **2020**, *16*, 81–104. [CrossRef]
11. Ingham, K.L.; Somayaji, A.; Burge, J.; Forrest, S. Learning DFA representations of HTTP for protecting web applications. *Comput. Netw.* **2007**, *51*, 1239–1255. [CrossRef]
12. Sivri, T.T.; Akman, N.P.; Berkol, A.; Peker, C. Web intrusion detection using character level machine learning approaches with upsampled data. *Ann. Comput. Sci. Inf. Syst.* **2022**, *32*, 269–274 .
13. 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. [CrossRef]
14. Vartouni, A.M.; Kashi, S.S.; Teshnehlab, M. An anomaly detection method to detect web attacks using stacked auto-encoder. In Proceedings of the 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, Iran, 28 February–2 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 131–134.
15. Ariu, D.; Tronci, R.; Giacinto, G. HMMPayl: An intrusion detection system based on Hidden Markov Models. *Comput. Secur.* **2011**, *30*, 221–241. [CrossRef]
16. Liang, J.; Zhao, W.; Ye, W. Anomaly-based web attack detection: A deep learning approach. In Proceedings of the 2017 VI International Conference on Network, Communication and Computing, Kunming, China, 8–10 December 2017; pp. 80–85.
17. 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, 1–3 December 2019; Proceedings, Part II 11; Springer: Berlin/Heidelberg, Germany, 2019; pp. 121–136.
18. 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, Virtually, 6–9 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2479–2488.

19. 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. [CrossRef]
20. Gong, X.; Lu, J.; Zhou, Y.; Qiu, H.; He, R. Model uncertainty based annotation error fixing for web attack detection. *J. Signal Process. Syst.* **2021**, *93*, 187–199. [CrossRef]
21. Tekerek, A. A novel architecture for web-based attack detection using convolutional neural network. *Comput. Secur.* **2021**, *100*, 102096. [CrossRef]
22. 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), Sousse, Tunisia, 11–13 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
23. Alaoui, R.L.; et al. Web attacks detection using stacked generalization ensemble for LSTMs and word embedding. *Procedia Comput. Sci.* **2022**, *215*, 687–696. [CrossRef]
24. Moarref, N.; Sandıkkaya, M.T. MC-MLDCNN: Multichannel Multilayer Dilated Convolutional Neural Networks for Web Attack Detection. *Secur. Commun. Netw.* **2023**, *2023*, 2415288. [CrossRef]
25. 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 33rd ACM International Conference on Information and Knowledge Management, Boise, ID, USA, 21–25 October 2024; pp. 5038–5045.
26. Katbi, A.; Ksantini, R. One-class IoT anomaly detection system using an improved interpolated deep SVDD autoencoder with adversarial regularizer. *Digit. Signal Process.* **2025**, *162*, 105153. [CrossRef]
27. Tokmak, M.; Nkongolo, M. Stacking an autoencoder for feature selection of zero-day threats. *arXiv* **2023**, arXiv:2311.00304.
28. Alghawazi, M.; Alghazzawi, D.; Alarifi, S. Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model. *Mathematics* **2023**, *11*, 3286. [CrossRef]
29. 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. [CrossRef]
30. 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. [CrossRef] [PubMed]
31. Mohamed, S.M.; Rohaim, M.A. Multi-Class Intrusion Detection System using Deep Learning. *J. Al-Azhar Univ. Eng. Sect.* **2023**, *18*, 869–883. [CrossRef]
32. 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. *J. Netw. Comput. Appl.* **2022**, *198*, 103270. [CrossRef]
33. Bedi, P.; Gupta, N.; Jindal, V. Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Comput. Sci.* **2020**, *171*, 780–789. [CrossRef]
34. Milosevic, M.S.; Ciric, V.M. Extreme minority class detection in imbalanced data for network intrusion. *Comput. Secur.* **2022**, *123*, 102940. [CrossRef]
35. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [CrossRef]
36. Yuan, Y.; Lu, Y.; Zhu, K.; Huang, H.; Yu, L.; Zhao, J. A Static Detection Method for SQL Injection Vulnerability Based on Program Transformation. *Appl. Sci.* **2023**, *13*, 11763. [CrossRef]
37. Vorobyov, K.; Gauthier, F.; Krishnan, P. Synthesis of Allowlists for Runtime Protection against SQLi. In Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results, Lisbon, Portugal, 14–20 April 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 16–20.
38. 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 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, Seattle, WA, USA, 17–21 July 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 1043–1054.
39. Silvestre, A.; Medeiros, I.; Mordido, A. Towards a SQL Injection Vulnerability Detector Based on Session Types. In Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering, Angers, France, 28–29 April 2024; Volume 1: ENASE. INSTICC; SciTePress, 2024; pp. 711–718.
40. 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), Virtual, 14–16 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 276–284.
41. 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, 23–25 October 2018; Revised Selected Papers 1; Springer: Berlin/Heidelberg, Germany, 2019; pp. 473–484.
42. Levene, M.; Poulovassilis, A.; Davison, B.D. Learning web request patterns. In *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 435–459.
43. Vijayarani, S.; Janani, R. Text mining: open source tokenization tools-an analysis. *Adv. Comput. Intell. Int. J. (ACII)* **2016**, *3*, 37–47.

44. Rashvand, N.; Hosseini, S.S.; Azarbayjani, M.; Tabkhi, H. Real-Time Bus Arrival Prediction: A Deep Learning Approach for Enhanced Urban Mobility. *arXiv* **2023**, arXiv:2303.15495.
45. Kefayat, E.; Thill, J.C. Urban Street Network Configuration and Property Crime: An Empirical Multivariate Case Study. *ISPRS Int. J.-Geo-Inf.* **2025**, *14*, 200. [CrossRef]
46. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N. Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 2017 Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017. .
47. 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. [CrossRef]
48. Shaheed, A.; Kurdy, M.B. Web application firewall using machine learning and features engineering. *Secur. Commun. Netw.* **2022**, *2022*, 5280158. [CrossRef]
49. DuckDuckBug. CNN Web Application Firewall. 2023. Available online: https://github.com/DuckDuckBug/cnn_waf (accessed on 29 January 2025).
50. Jagat, R.R.; Sisodia, D.S.; Singh, P. Detecting web attacks from HTTP weblogs using variational LSTM autoencoder deviation network. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2210–2222. [CrossRef]
51. Abshari, D.; Fu, C.; Sridhar, M. LLM-assisted Physical Invariant Extraction for Cyber-Physical Systems Anomaly Detection. *arXiv* **2024**, arXiv:2411.10918.
52. 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* **2024**, arXiv:2407.07966.
53. Abshari, D.; Sridhar, M. A Survey of Anomaly Detection in Cyber-Physical Systems. *arXiv* **2025**, arXiv:2502.13256.
54. Babaey, V.; Ravindran, A. GenXSS: An AI-Driven Framework for Automated Detection of XSS Attacks in WAFs. *arXiv* **2025**, arXiv:2504.08176.
55. 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* **2023**, arXiv:2302.11382.
56. 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), Olomouc, Czech Republic, 8–12 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 273–278.
57. Talebi, S.; Zhou, K. Graph Neural Networks for Efficient AC Power Flow Prediction in Power Grids. *arXiv* **2025**, arXiv:2502.05702.
58. Zibaeirad, A.; Vieira, M. Reasoning with LLMs for Zero-Shot Vulnerability Detection. *arXiv* **2025**, arXiv:2503.17885.