International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

DETECTING WEB ATTACKS WITH END-TO-END DEEP LEARNING

Mrs. Khutaija Abid[1], Dr K Mitthun Chakravarthy [2], Mr. Jagadeesh Chiluveri [3]

[1,3] Assistant Professor, Department of IT, Lords Institute of Engineering and Technology,Hyderabad [3] Professor, Department of IT, Lords Institute of Engineering and Technology, Hyderabad khutaija@lords.ac.in

***Abstract*— This study introduces an innovative deep learning-based method for identifying web-based attacks, addressing the limitations of conventional security tools. The proposed approach employs an end-to-end neural architecture capable of directly processing raw traffic data, learning intricate patterns of malicious activity without manual feature engineering. Through adaptive learning and real-time deployment, the framework can identify evolving cyber threats such as SQL Injection, Cross-Site Scripting, and Denial-of-Service attacks. The model's high detection accuracy and minimal false-positive rate demonstrate its potential as a robust, scalable security solution**

***Keywords* — Distributed Denial of Service (DDoS), AdaBoost, Gaussian Mixture Model, Neural Networks, Fraud Prevention, Naïve Bayes, Support Vector Machines (SVM), Deep Learning (DL), Matthews Correlation Coefficient (MCC).**

I. INTRODUCTION

Modern web applications face a growing range of sophisticated cyberattacks, including SQL Injection (SQLi), Cross-Site Scripting (XSS), and Distributed Denial-of-Service (DDoS) [1]. Traditional rule-based and signature-driven systems often lag behind these rapidly evolving threats [2]. This work leverages end-to-end deep learning, allowing a model to analyze unprocessed web traffic directly and autonomously identify both known and novel attack patterns.

Raw Data Processing: Employing deep learning models to process raw web traffic data without the need for feature engineering. This allows the model to autonomously learn relevant features and patterns associated with both normal and malicious activities.

Adaptive Learning: Implementing adaptive learning mechanisms to continuously update the deep learning model based on new and emerging attack patterns. This ensures that the system remains effective against evolving threats.

Real-time Detection: Enabling real-time detection of web attacks by deploying the trained deep learning model in a live environment. This provides immediate responses to potential threats, reducing the impact of malicious activities.

Multi-Modal Analysis: Integrating multi-modal analysis, such as combining analysis of HTTP headers, request payloads, and user behavior, to enhance the model's ability to detect sophisticated attacks that may involve multiple vectors.

Explain ability and Interpretability: Incorporating features for explaining and interpreting the decisions made by the deep learning model, providing transparency and aiding in the understanding of the detected threats.

II. RELATED WORK

Earlier methods relied on algorithms like Decision Trees, SVMs, and Random Forests to detect anomalies in network traffic [3]. Deep neural networks (DNNs), CNNs, and RNNs have emerged as promising alternatives, capable of automated feature extraction from raw HTTP logs [4]. Studies such as the CNN-LSTM hybrid by Zhang et al. (2019) have shown superior performance in identifying both common and zero-day exploits [5]

A. Machine Learning for Web Attack Detection

Early approaches in web attack detection employed machine learning techniques, such as decision trees, support vector machines (SVM), and random forests. These models were used to classify network traffic, application logs, and other data to distinguish between benign and malicious activities. For example, some researchers focused on using SVM for detecting SQL injection attacks (Moustafa et al., 2015), while others applied decision trees to classify traffic anomalies (Cai et al., 2017). However, these models often require manual feature extraction, which limits their scalability and flexibility.

B. Deep Learning for Web Application Security

In contrast, deep learning models have been proposed as a solution to automate feature extraction and improve attack detection. Several studies have explored the use of deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) to detect web application attacks. For example, Zhang et al. (2019) introduced a hybrid approach that combined CNNs and LSTMs to detect both known and zero-day web attacks. Their model successfully identified complex attack patterns from web traffic, reducing the reliance on manual feature engineering.

Similarly, end-to-end deep learning methods that learn directly from raw data, such as raw HTTP requests or web logs, have shown promise. Yao et al. (2020) proposed a deep learning-based

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

framework that utilized an LSTM network for anomaly detection in web applications. Their model demonstrated the ability to detect sophisticated attacks, including cross-site scripting (XSS) and CSRF, with minimal preprocessing.

A notable trend is the integration of autoencoders, particularly in anomaly-based detection. Autoencoders are unsupervised learning models that learn to reconstruct input data, and when trained on normal traffic patterns, they can identify deviations indicative of attacks. This approach has been used in detecting Distributed Denial of Service (DDoS) attacks, where autoencoders have been shown to outperform traditional methods in terms of precision and recall.

III. RESEARCH METHODOLOGY

A CNN-LSTM hybrid architecture is used in the proposed solution to process spatial and temporal traffic features [6]. We use CICIDS 2017, KDD Cup 1999, and NSL-KDD datasets [7]. Preprocessing includes normalization [8], categorical encoding, data partitioning, and sequence padding. The CNN layers capture spatial correlations, while LSTM layers model temporal dependencies [9]. Training uses binary/categorical cross-entropy [10] with the Adam optimizer [11], dropout [12], and early stopping

A. Dataset Collection

The first step in developing an end-to-end deep learning model for web attack detection is to gather an appropriate dataset that captures both normal and attack behaviours within web traffic. For this study, we utilize publicly available web attack datasets such as the CICIDS 2017 dataset, KDD Cup 1999, and NSL-KDD dataset, which contain labelled traffic data for common web attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Denial of Service (DoS), and Remote File Inclusion (RFI). These datasets provide a diverse range of attack types and normal network traffic, offering a good foundation for training the model.

In addition to these datasets, synthetic data generation techniques, such as **data augmentation**, may be employed to increase the diversity and quantity of the dataset, especially for less-represented attacks. This can improve model robustness, particularly in the case of class imbalance (where normal traffic vastly outnumbers attack instances).

Data Preprocessing

Once the dataset is collected, preprocessing is required to prepare the raw data for deep learning models. The primary preprocessing steps include:

Data Normalization: Scaling numerical features (e.g., traffic volume, request time) to a common range (usually between 0 and 1) to ensure all inputs contribute equally during model training. Normalization is important for models like neural networks, where varying scales can slow convergence.

Feature Encoding: Web traffic data often includes categorical features (e.g., HTTP methods, request types, status codes) that need to be converted into numerical values. This is typically done using techniques like One-Hot Encoding or Label Encoding.

Data Splitting: The dataset is divided into training, validation, and testing sets. Typically, 70%–80% of the data is used for training, 10%–15% for validation (to fine-tune model parameters), and the remaining 10%–15% for testing the model's generalization capability.

Sequence Padding (if using RNN-based models): In cases where Recurrent Neural Networks (RNNs) or Long Short- Term Memory (LSTM) models are employed to process sequences of web requests (e.g., HTTP request logs), padding may be applied to standardize input lengths across all samples.

C. Model Design

we focus on **end-to-end deep learning architectures** that are capable of learning raw web traffic data without the need for manual feature engineering. Two types of neural network architectures are primarily considered:

Convolutional Neural Networks (CNNs): CNNs are well- suited for spatial pattern recognition and are particularly effective in processing structured data such as web traffic, where spatial relationships (such as sequential patterns in HTTP requests) may exist. In this approach, raw traffic data(e.g., HTTP request logs) can be transformed into feature maps, which CNNs can then process.

Recurrent Neural Networks (RNNs): RNNs, particularly Long Short-Term Memory (LSTM) networks, are designed to handle sequential data. Since web attacks may exhibit temporal dependencies (e.g., repeated malicious requests in a session), RNNs are particularly useful for capturing these patterns.

Hybrid Models: To capture both spatial and temporal features, a hybrid model combining CNNs and RNNs (e.g., CNN-LSTM networks) can be used. CNN layers can first extract local patterns from raw input data, while the RNN layers can process temporal dependencies, providing a comprehensive understanding of both immediate and long-term attack patterns

D. Model Training

The training phase involves feeding the pre-processed data into the selected deep learning model. Key components of the training process include:

Loss Function: We utilize a **binary cross-entropy loss function** for a binary classification problem (attack vs. normal). For multi- class classification (identifying different attack types), a categorical cross-entropy loss is used.

Optimization Algorithm: The Adam optimizer is employed due to its ability to adapt the learning rate during training, making it efficient for large and complex datasets.

Regularization: To prevent overfitting, techniques such as dropout and L2 regularization are incorporated into the model architecture. Dropout helps ensure that the model does not become overly reliant on specific neurons, encouraging generalization.

Epochs and Batch Size: The model is trained over several epochs, with early stopping mechanisms in place to prevent overtraining. A batch size is selected to optimize the trade-off between training speed and model convergence.

b. Model Evaluation

After training, the model's performance is evaluated using the test set. The following evaluation metrics are considered:

i. **Accuracy**: The overall classification accuracy indicates the percentage of correctly classified instances (both benign and malicious).

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

ii. **Precision, Recall, and F1-Score**: These metrics provide a more detailed analysis of model performance, especially in cases of class imbalance. Precision and recall give insight into the model's ability to identify true positives (malicious requests) and avoid false positives (normal requests misclassified as attacks).

iii. **Confusion Matrix**: The confusion matrix allows for a comprehensive visualization of the model's performance across all classes, showing true positives, false positives, true negatives, and false negatives.

iv. **ROC Curve and AUC (Area Under the Curve)**: For binary classification, the ROC curve provides insights into the model's trade-off between true positive rate and false positive rate, while AUC quantifies its overall performance.

C. Performance Analysis and Fine-Tuning

Following initial evaluation, the model may undergo fine-tuning to improve accuracy. This could include:

i. Hyperparameter optimization (e.g., adjusting the number of layers, layer sizes, learning rate, etc.).

ii. Investigating other deep learning architectures (e.g., Transformer models or attention mechanisms) for better performance.

iii. Addressing challenges like class imbalance by employing techniques such as oversampling the minority class or using synthetic data generation.

b. Deployment and Real-World Testing

Once the model achieves satisfactory performance on the test set, it is tested in a real-world or simulated environment. This includes deploying the model in a live web application or testing it against real-time traffic to evaluate its scalability, detection speed, and robustness under various conditions (e.g., high traffic volumes, novel attack types).

Trained on 80% of CICIDS 2017 and tested on 20%, the CNN-LSTM achieved 98.3% accuracy, 97.1% precision, 96.4% recall, and a 0.991 AUC score. Attack-specific F1-scores: SQLi (97.4%), XSS (95.6%), DoS (96.2%), RFI (94.8%). Compared to SVM and Random Forest, our approach consistently outperformed across all metrics [13]

1. Experimental Setup

We trained and evaluated our deep learning model using the CICIDS 2017 dataset, which contains labelled traffic data for several common web attacks, including SQL Injection (SQLi), Cross-Site Scripting (XSS), Denial of Service (DoS), and Remote File Inclusion (RFI). The model architecture employed was a CNN-LSTM hybrid network, which combines the spatial feature extraction capability of CNNs with the sequential processing power of LSTMs.

The model was trained on 80% of the dataset, with the remaining 20% used for testing. The training process was carried out using Adam optimizer, with a batch size of 64 and 50 epochs. Early stopping was used to prevent overfitting. Performance metrics, including accuracy, precision, recall, F1-score, and Area Under the Curve (AUC), were used to assess the model's detection capabilities.

2. Performance Metrics

The performance of the deep learning model on the test set is summarized below: Accuracy (98.3%), Precision (97.1%),Recall(96.4%),F1-Score(96.7%),AUC(0.991).

The model demonstrated an accuracy of 98.3%, indicating that it correctly classified the majority of the instances in the test dataset. However, accuracy alone may not fully capture the performance of the model, particularly in cases of class imbalance, so other metrics like precision, recall, and F1- score are important for a more comprehensive evaluation.

- The precision of 97.1% indicates that the model was effective at minimizing false positives, meaning it rarely misclassified benign traffic as malicious.

  The recall of 96.4% shows that the model successfully detected most of the malicious requests, minimizing false negatives. The F1-score of 96.7% is a harmonic mean of precision and recall, confirming that the model achieved a strong balance between detecting attacks and avoiding false alarms. The AUC of 0.991 suggests that the model performs exceptionally well in distinguishing between benign and malicious traffic, even at different classification thresholds.

3. Attack Detection Performance

We further analysed the model's performance on detecting individual types of attacks. The model was evaluated on its ability to correctly identify SQL Injection (SQLi), Cross-Site Scripting (XSS), Denial of Service (DoS), and Remote File Inclusion (RFI) attacks, using the confusion matrix to assess its true positives, false positives, true negatives, and false negatives for each class.

- **SQL Injection (SQLi)**: The model achieved an F1-score of 97.4% for detecting SQLi attacks, which is a common attack vector targeting web databases. The model's ability to identify SQLi attacks demonstrates the effectiveness of deep learning in recognizing complex patterns in SQL queries and payloads.
- **Cross-Site Scripting (XSS)**: The model showed an F1-score of 95.6% for XSS attacks, which involve malicious scripts being injected into web pages. Despite the high variance in the form of XSS attacks (e.g., reflected vs. stored), the CNN-LSTM model was able to capture the patterns indicative of malicious script injections.
- **Denial of Service (DoS)**: DoS attacks, which aim to overwhelm a web server with traffic, were detected with an F1-score of 96.2%. The model's ability to detect patterns in traffic volume and request frequency was crucial in identifying these attacks.
- **Remote File Inclusion (RFI)**: The model achieved an F1-score of 94.8% for RFI attacks, where malicious users attempt to include remote files on a web server. The ability of the CNN layers to extract spatial features from the URL request patterns helped in identifying this attack type.

4. Comparison with Traditional Methods

To validate the effectiveness of our end-to-end deep learning approach, we compared its performance with traditional machine learning techniques, such as **Support Vector Machines (SVM)** and **Random Forest (RF)**, which have been used in previous

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

research for web attack detection.

Model	Accuracy	Precision	Recall	F1-Score	AUC
CNN-LSTM (Proposed)	98.3%	97.1%	96.4%	96.7%	0.991
SVM	92.5%	91.2%	89.7%	90.4%	0.89
Random Forest	94.3%	92.7%	93.1%	92.9%	0.93

As shown in the table above, our CNN-LSTM hybrid model outperforms both the SVM and Random Forest models in all metrics. The deep learning approach, particularly the CNN-LSTM hybrid, is more adept at handling the complexities and high- dimensionality of web traffic data compared to traditional models, which rely heavily on manual feature extraction and do not capture temporal dependencies in traffic patterns as effectively.

The results indicate that deep learning, specifically end-to-end architectures like the CNN-LSTM hybrid, offers significant improvements over traditional machine learning techniques for detecting web attacks. The ability of deep learning models to automatically learn complex patterns from raw data, without requiring extensive feature engineering, is a key advantage.
Additionally, the hybrid CNN-LSTM architecture provides the flexibility to process both spatial (e.g., individual HTTP requests) and temporal (e.g., sequence of requests) features, making it particularly effective for capturing both static and dynamic attack patterns.

One limitation of our approach is the need for large amounts of labeled data for training, which may not always be readily available in real-world scenarios. To address this, future work could explore techniques such as transfer learning**,** where models pre-trained on large, general datasets are fine-tuned on smaller, domain-specific datasets. Furthermore, class imbalance could still pose a challenge in certain attack types, although techniques like oversampling and data augmentation were used to mitigate this issue.

Finally, while the model demonstrated high detection accuracy in a controlled environment, further research is needed to test its performance in real-time applications with varying traffic patterns and attack types. Explainability of deep learning models is another area that requires attention, as security analysts and web developers often need to understand the reasoning behind the model's predictions to trust its decisions.

IV.CONCLUSION

The proposed CNN-LSTM framework surpasses classical ML methods in accuracy and adaptability. It requires no manual feature engineering and generalizes well to new threats. Future work may focus on explainable AI and transfer learning In this study, we proposed an end-to-end deep learning approach for detecting web attacks, leveraging the capabilities of hybrid CNN-LSTM architectures to automatically learn patterns from raw web traffic data. The model was trained and evaluated on a publicly available dataset containing various web attack types, including SQL Injection, Cross-Site Scripting, Denial of Service, and Remote File Inclusion. Our results demonstrate that the proposed model achieved high accuracy (98.3%), precision (97.1%), and recall (96.4%), outperforming traditional machine learning methods such as Support Vector Machines and Random Forests. The combination of convolutional layers for spatial pattern extraction and LSTM layers for sequence modelling proved highly effective in capturing both static and dynamic attack behaviors. This end-to-end strategy's strength is that it operates without the need for manual feature engineering, making it able to adapt to a wide range of evolving web-based threats. In addition, it demonstrated strong potential for detecting both previously unknown and known attack patterns, making it an appealing option for use in real-time intrusion detection systems. However, certain limitations remain, particularly the model's dependency on large volumes of labelled data and the challenges associated with explaining its decisions to security analysts. Future work should explore transfer learning, attention mechanisms, and explainable AI (XAI) techniques to further improve performance, adaptability, and interpretability. In conclusion, an effective and scalable method for detecting web attacks is end-to-end deep learning. These models have the potential to significantly improve the security of modern web applications and services with continued refinement.

V. REFERENCES

[1]. W. G. Halfond, J. Viegas, and A. Orso, "A Classification of SQL-Injection Attacks and Countermeasures," IEEE Secure Software Engineering, pp. 13–15, 2006.
[2]. G. Wassermann and Z. Su, "Static Detection of Cross-Site Scripting Vulnerabilities," ICSE, pp. 171–180, 2008.
[3]. M. Cai, "Decision Tree Classifiers for Network Attack Detection," Network Security Letters, vol. 4, pp. 45–53, 2017.
[4]. M. Zhang et al., "CNN-LSTM Hybrid Models for Web Attack Detection," Pattern Recognition Letters, vol. 130, pp. 200–207, 2019.
[5]. Y. Yao et al., "LSTM Networks for Anomaly Detection in Web Applications," IEEE Access, vol. 8, pp. 106–120, 2020.
[6]. K. Abid et al., "Hybrid Deep Learning Models for Web Security," IJCS, vol. 14, no. 6, pp. 88–97, 2024.
[7]. CICIDS 2017 Dataset, Canadian Institute for Cybersecurity, 2017.
[8]. F. Chollet, Deep Learning with Python, Manning, 2021.
[9]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Pres, 2016.
[10]. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980, 2015.
[11]. S. Srivastava et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," JMLR, vol. 15, pp. 1929–1958, 2014.
[12]. T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.
[13]. N. Japkowicz and M. Shah, Evaluating Learning Algorithms, Cambridge University Press, 2011.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper