Fine-Tuning Transformer LLMs for Detecting SQL Injection and XSS Vulnerabilities

Thi-Thu-Huong Le
Blockchain Platform Research Center
Pusan National University
Busan, South Korea
lehuong7885@gmail.com

Andro Aprila Adiputra
dept. computer science and engineering
Pusan National University
Busan, South Korea
androaprila@pusan.ac.kr

YeonJeong Hwang
dept. computer science and engineering
Pusan National University
Busan, South Korea
yeonjeong@islab.re.kr

JunYoung Son
dept. computer science and engineering
Pusan National University
Busan, South Korea
jysonpaperinfo@gmail.com

Howon Kim
dept. computer science and engineering
Pusan National University
Busan, South Korea
howonkim@pusan.ac.kr

***Abstract*—This paper introduces a method for detecting SQL Injection (SQLi), Command Injection, and Cross-Site Scripting (XSS) vulnerabilities using fine-tuned, transformer-based language models within a multilabel binary classification framework. Our approach takes advantage of three pre-trained lightweight models: DistilBERT, ALBERT, and ArmoRM-Llama3-8B. Each fine-tuned for text classification tasks on the SQLi XSS dataset. Fine-tuning involved freezing the pretrained transformer layers and updating a fully connected output layer. The primary challenge lies in generating multi-label outputs for four vulnerability classes (SQL Injection, Command Injection, XSS, and Normal traffic) with binary indicators for normal (0) or attack (1) status. Evaluation metrics, including confusion matrices, indicate that fine-tuning ArmoRM-Llama3-8B achieves slightly higher accuracy and detection rates than DistilBERT and ALBERT, particularly in identifying complex injection attacks. Furthermore, ArmoRM-Llama3-8B demonstrated the fastest testing evaluation time despite a moderately longer training period. These results highlight the feasibility and effectiveness of transformer-based language models in enhancing web security through improved vulnerability detection.**

***Index Terms*—ALBERT, ArmoRM-Llama3-8B, Command Injection, Cross-Site Scripting (XSS), DistilBERT, Multi-label classification, SQL Injection, Transformer language models, Vulnerability detection, Web security**

I. Introduction

The security of web applications remains a critical issue, as vulnerabilities such as SQL Injection (SQLi), Command Injection, and Cross-Site Scripting (XSS) continue to pose significant risks to both users and organizations [1]. SQL injection occurs when malicious SQL queries are injected into input fields, enabling attackers to gain unauthorized access to sensitive data or manipulate database information [2]. Command injection vulnerabilities allow attackers to execute arbitrary system commands by exploiting insecure input validation mechanisms in applications [3]. Similarly, XSS vulnerabilities enable attackers to inject malicious scripts into web pages, which, when executed by users' browsers, can lead to various security breaches such as data theft or session hijacking [4]. Traditional vulnerability detection methods, including signature-based and rule-based approaches, often fail to detect sophisticated attack patterns and adapt to the ever-evolving tactics employed by malicious actors [5].

In recent years, transformer-based language models have shown exceptional performance in natural language processing (NLP) tasks, making them an attractive option to improve the security of web applications [6], [7]. Models such as BERT (Bidirectional Encoder Representations from Transformers) [8], DistilBERT [9], and ALBERT [10] have demonstrated the ability to understand complex relationships in textual data through their self-attention mechanisms. This capability has led to adopting these models in various cybersecurity tasks, including malware detection, phishing detection, and vulnerability identification in source code [11], [12]. By fine-tuning these models on domain-specific datasets, they can learn to detect web vulnerabilities like SQL Injection, Command Injection, and XSS with high accuracy. However, applying transformers to a multi-label classification problem, where multiple attack types must be detected simultaneously, presents challenges due to the need for the model to differentiate between attack types while identifying normal traffic.

The purpose of this study is to investigate the effectiveness of fine-tuning three transformer-based models, including DistilBERT [9], ALBERT [10], and ArmoRM-Llama3-8B [13], for the detection of SQL Injection, Command Injection,

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Convergence security core talent training business(Pusan National University) support program(RS-2022-II221201) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation) and MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(RS-2020-II201797) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation). And this research was also supported by Korea Planning & Evaluation Institute of Industrial Technology(KEIT) grant funded by the Korea government(MOTIE) (No.RS-2024-00508087, Development of Virtual Environment-based Cyber Security Verification Technology).
Corresponding author: JunYoung Son (jysonpaperinfo.com)

and XSS vulnerabilities in a multi-label classification setting. The models are fine-tuned on the SQLi XSS dataset [14], where each input sample is labeled with binary indicators for each vulnerability type (normal or attack). The fine-tuning process involves freezing the pretrained transformer layers and updating only the output layer, specifically designed for vulnerability detection tasks. The challenge of this approach lies in effectively distinguishing between multiple vulnerabilities while ensuring that each model can handle the binary classification task for each label.

This paper makes three primary contributions:

- We propose a fine-tuning framework for three transformer-based language models, DistilBERT, ALBERT, and ArmoRM-Llama3-8B, to detect SQL Injection, Command Injection, and XSS vulnerabilities in a multi-label classification task.
- We provide a comparative analysis of the models' performance using confusion matrices, evaluating their ability to detect multiple vulnerabilities simultaneously. In this way, we provide the efficiency of the models in terms of training and testing time, model loss measurement, evaluation samples per second, and evaluation steps per second.

The remainder of the paper is structured as follows. Section II reviews the related work in vulnerability detection, focusing on transformer-based models and their applications in cybersecurity. Section III details the methodology used in this study, including data preprocessing, model fine-tuning, and evaluation metrics. Section IV presents the experimental setup, including dataset description, training procedures, and evaluation results. Section V discusses the results obtained from the experiments, comparing the performance of the three transformer models. Section VI provides concluding remarks and suggests directions for future work.

II. Related Work

A. SQL Injection Detection

SQLi remains one of the most critical vulnerabilities in web applications. Early detection techniques for SQLi relied primarily on signature-based methods and static code analysis. These methods, although effective at detecting known attack patterns, often fail to capture novel or obfuscated attack variants [15]. Recent research has used machine learning (ML) algorithms to improve SQLi detection, using features extracted from HTTP requests, SQL queries, or application logs [16]. For example, machine learning models such as decision trees, support vector machines (SVM), and random forests have been used to classify SQLi attacks from benign requests [17]. More recently, deep learning techniques, including Convolutional Neural Networks (CNNs), have been proposed to capture complex patterns in web traffic and SQL queries for SQLi detection [18]. However, these methods typically require significant feature engineering, which limits their scalability and adaptability to new attack techniques.

Transformer-based models, such as BERT and its variants, have also been explored for SQLi detection, with promising results in capturing semantic dependencies within web requests. A recent study fine-tuned BERT on a SQLi dataset and showed significant improvements over traditional machine learning models [19], [20]. However, these models were designed primarily for single-label classification tasks, which poses challenges when detecting multiple types of vulnerabilities simultaneously, as is often the case in real-world scenarios.

B. XSS Vulnerability Detection

XSS is another common vulnerability that allows attackers to inject malicious scripts into web pages. Like SQLi, traditional methods for detecting XSS attacks involve pattern matching and static code analysis. However, these methods struggle to detect more sophisticated XSS payloads, such as those using advanced encoding techniques or obfuscation [21].

Recent works have applied machine learning models, such as Naive Bayes and random forests, to detect XSS vulnerabilities from web requests or JavaScript code [22]. These models rely on the assumption that XSS payloads exhibit distinct patterns that can be captured by handcrafted features. Deep learning approaches, such as CNNs and Recurrent Neural Networks (RNNs), have been proposed to automatically learn features from raw input data, reducing the need for manual feature extraction [23]. These models have demonstrated better performance in identifying XSS attacks, particularly those using advanced obfuscation techniques.

In addition to these approaches, transformer-based models have gained attention for their ability to understand the context of web requests and JavaScript code. Fine-tuning transformer models like BERT for XSS vulnerability detection has been shown to improve detection accuracy by leveraging the model's ability to capture long-range dependencies in web content [20]. However, as with SQLi detection, most of the work on XSS vulnerability detection using transformers has focused on single-label classification.

C. Multi-Label Classification for SQLi and XSS Detection

The ability to detect multiple vulnerabilities in a single web request is crucial in modern web security, where requests may contain multiple attack types simultaneously. Traditional multi-class classification models, however, do not perform well in this scenario, as they assume that each input belongs to only one class. Multi-label classification, where each input can belong to multiple classes, is more appropriate for simultaneously detecting vulnerabilities like SQLi, XSS, and others.

Several studies have explored multi-label vulnerability detection classification, including SQLi and XSS. For example, a recent study used CNNs for multi-label classification to detect various types of web vulnerabilities, including SQLi, XSS, and others [20], [24]. However, despite their proven success in single-label tasks, transformer models have not been extensively studied for multi-label classification tasks in the context of SQLi and XSS. Our work aims to fill this gap by fine-tuning three transformer-based models, comprised of

DistilBERT, ALBERT, and ArmoRM-Llama3-8B, for multi-label classification to detect both SQLi and XSS vulnerabilities, along with other attack types.

III. Proposed Method

In this section, we describe the transformer models selected for our experiment, the fine-tuning strategy employed, and the multi-label binary classification framework used for vulnerability detection. Our method aims to detect multiple vulnerabilities, such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and other attack types, by adapting transformer-based models to a multi-label classification problem. The general proposed method is shown in Figure 1.

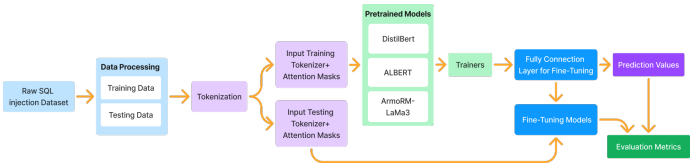


Fig. 1: The proposed fine-tuning LLM models for SQL injection and XSS detection.

A. Data Preprocessing

The dataset used in our experiments, SQLi XSS dataset, consists of web traffic logs containing both attack and normal traffic examples. Before training the models, we applied several preprocessing steps to the dataset to ensure that the data was in a suitable format for the transformer models.

- **Data Splitting**: The dataset was split into training, validation, and testing sets, with 60% of the samples used for training, 10% for validation, and 20% for testing.
- **Text Tokenization**: The input text (web traffic logs) was tokenized using the pre-trained tokenizer for each model. The tokenization process converts raw text into a sequence of tokens, which can be processed by the transformer models. Special tokens like `[CLS]` and `[SEP]` were added for model compatibility.
- **Normalization**: Input data, such as numerical features within the web logs, was normalized using standard scaling techniques to ensure that numerical values did not disproportionately influence model predictions.

The preprocessed data was then fed into the models for fine-tuning, and the evaluation was performed using the metrics described above.

B. Pretrained Transformer Models

We have selected three state-of-the-art transformer models for this study: DistilBERT, ALBERT, and ArmoRM-Llama3-8B. Each of these models has unique characteristics that make them suitable for our vulnerability detection task.

- **DistilBERT** [9]: DistilBERT is a lighter and faster version of BERT designed to retain 97% of BERT's language understanding while being 60% faster and using 40% fewer parameters.
- **ALBERT** [10]: ALBERT (A Lite BERT) is a variant of BERT that reduces the model size by sharing parameters across layers and factorizing the embedding matrix. This reduction in the number of parameters helps improve efficiency while retaining the performance of BERT.
- **ArmoRM-Llama3-8B** [13]: ArmoRM-Llama3-8B is a large pretrained transformer model based on the Llama architecture, designed to understand complex linguistic structures. With 8 billion parameters, this model can handle intricate patterns in web traffic and detect complex vulnerability signatures.

C. Fine-Tuning Strategy

Our fine-tuning strategy involves freezing the transformer layers of the selected models and adding a fully connected (FC) layer for classification. The transformer layers are pretrained on large corpora, and we preserve these learned representations during fine-tuning, as they contain rich language features relevant for vulnerability detection. The frozen layers allow the model to retain its pretrained knowledge while preventing overfitting during the fine-tuning process.

- **Frozen Transformer Layers**: We freeze the parameters of the transformer layers in each model to avoid fine-tuning them during the training process.
- **Fully Connected Layer**: After the transformer layers, an FC layer is added to transform the hidden states from the transformer into output predictions. This FC layer outputs the class probabilities for each vulnerability type.

D. Multi-Label Binary Classification Framework

In our multi-label binary classification framework, the task is to predict multiple labels for a given web request, each corresponding to a different type of vulnerability (e.g., SQLi, XSS, or Command Injection). The label structure consists of binary values for each vulnerability type: a label of "1" indicates the presence of the vulnerability, and "0" indicates its absence. For example, a web request can be labeled as follows: Label = [Normal, SQLi, XSS, CommandInjection]. Normal: Indicates if the request is benign (non-malicious). SQLi: Indicates if the request contains a SQL Injection attack. XSS: Indicates if the request contains a Cross-Site Scripting attack. CommandInjection: Indicates if the request contains a Command Injection attack.

This label structure allows the model to detect multiple vulnerabilities in a single request simultaneously. The output layer of the model consists of multiple sigmoid neurons, each corresponding to one of the binary labels for the vulnerabilities. The sigmoid activation function ensures that each output is independent of the others, providing a probability for each output class.

During training, the binary cross-entropy loss function is applied to each label individually, and the model learns to predict the probabilities for each vulnerability type based on the input features. This enables the model to detect multiple

vulnerabilities within a single input, which is essential for real-world scenarios where multiple attack types may be present simultaneously.

E. Evaluation Metrics

To evaluate the performance of the models, we used several metrics to ensure a complete assessment. Specifically, we employed the following:

- **Confusion Matrix**: This metric provides a detailed summary of the predictions made by the model by comparing the true labels with the predicted labels. It offers insights into the performance of the model in classifying different types of vulnerabilities, including *Normal*, *SQL Injection*, *XSS*, and *Command Injection*.
- **Training and Evaluation Metrics**: We measured critical aspects such as training time, evaluation time, evaluation loss, evaluation samples per second, and evaluation steps per second. These metrics highlight the computational efficiency and effectiveness of the model during the training and validation phases.

IV. Experiment Setup

All experiments were conducted on a machine equipped with the following hardware specifications: We used PyTorch and the Hugging Face Transformers library for model development and fine-tuning. The experiments were carried out on an NVIDIA Tesla V100 DGX system with 32 GB of memory. The system specifications include NVIDIA-SMI 450.142.00, Driver Version: 450.142.00, CUDA Version: 11.7.

A. Hyperparameters Setting

The following hyperparameters were used for fine-tuning each of the three models: DistilBERT, Albert, RLHFlow-Llama3 as shown in Table I.

TABLE I: Hyperparameter Settings for the Proposed Models

Hyperparameter	**Value**
Number of Labels	4
Problem Type	Multi-Label Classification
Max Sequence Length	128
Learning Rate	5×10^{-4}
Batch Size (Train/Eval)	8
Number of Epochs	3
Optimizer	AdamW
Loss Function	Cross-Entropy
Tokenizer	Model-Specific Tokenizers
Metrics	Confusion Matrix, ROC Curve, AUC
Freezing Layers	All Except Classifier Layer

V. Results and Discussion

In this section, we present the results of our experiments comparing the performance of three transformer-based models: DistilBERT, ALBERT, and ArmoRM-Llama3-8B. The models are evaluated based on performance metrics such as evaluation loss, training time, evaluation time, and their efficiency in processing evaluation samples and steps per second.

A. Evaluation Performance of Three Models

The evaluation performance of the three models is summarized using confusion matrix results, as shown in Figure 2.

For SQL Injection detection, both DistilBERT and ArmoRM-LLama3-8B achieved high accuracy, exceeding 94%, significantly outperforming the AlBERT model, which achieved an accuracy of over 81%. In Command Injection detection, the AlBERT model failed to make accurate predictions, while DistilBERT demonstrated reasonable performance. ArmoRM-LLama3-8B delivered the best performance, showcasing superior detection capabilities compared to the other models. For XSS detection, all three models achieved excellent and comparable results, with accuracies exceeding 96%. Lastly, for Normal detection, DistilBERT achieved the highest accuracy. ArmoRM-LLama3-8B delivered competitive results, closely matching DistilBERT's performance, while AlBERT achieved a relatively lower accuracy of over 82%.

B. Comparison of Model Performance

Figure 3 and Table II provide a comparison of the models' key performance metrics, including training time, evaluation time, evaluation loss, and the processing efficiency of evaluation samples and steps per second. Although ArmoRM-Llama3-8B shows a higher evaluation loss compared to DistilBERT and ALBERT, its evaluation speed is much slower, processing fewer samples and steps per second.

As shown in Table II, DistilBERT offers the fastest training and evaluation times, followed by ALBERT. ArmoRM-Llama3-8B, despite having the longest training and evaluation durations, demonstrates a strong potential for the detection task, though its evaluation loss is higher. ALBERT's performance falls between the other two models, with slightly higher evaluation loss than DistilBERT but better evaluation speed than ArmoRM-Llama3-8B.

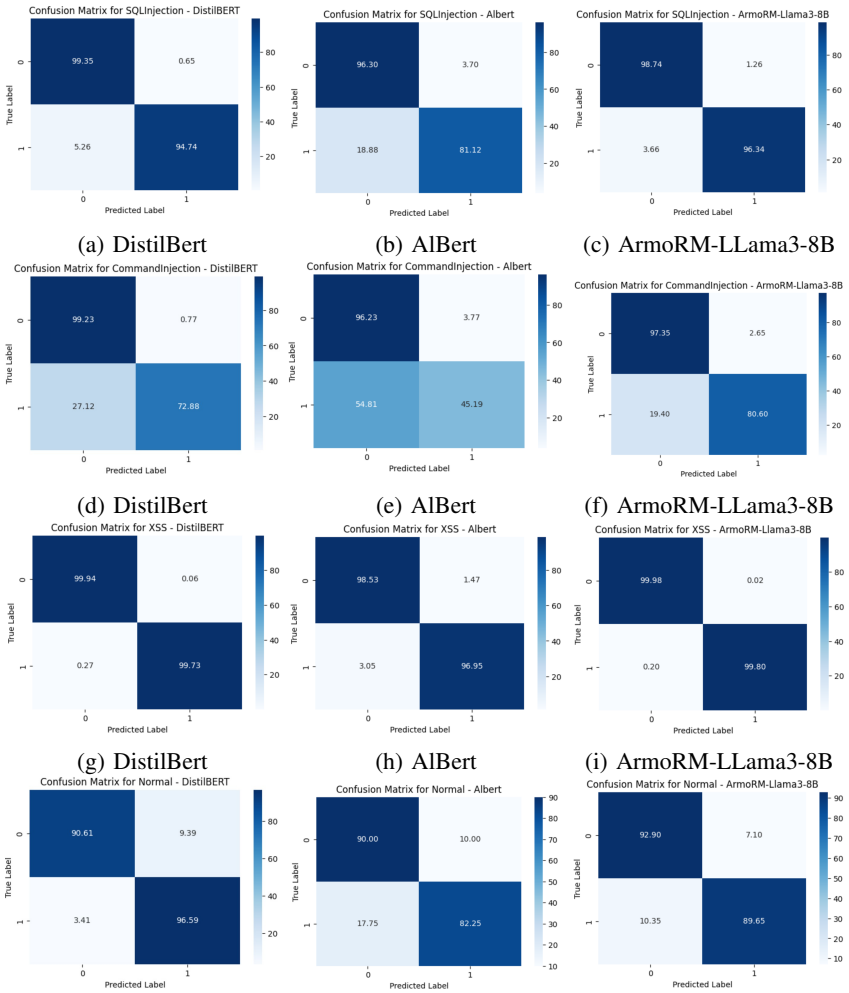
C. Analysis of Training Time and Testing Evaluation Time

The data from Table II clearly shows that ArmoRM-Llama3-8B requires significantly more time for both training (95537.17 seconds) and evaluation (6058.75 seconds) compared to DistilBERT and ALBERT. However, it processes evaluation samples and steps much more slowly, with a rate of only 6.821 samples per second and 0.853 steps per second. In contrast, DistilBERT processes 285.434 samples per second and 35.679 steps per second, making it the most efficient model in terms of both training and evaluation times.

In terms of evaluation performance, DistilBERT achieves the lowest evaluation loss (0.0982), followed by ArmoRM-Llama3-8B (0.1304), and ALBERT (0.2274). Although ArmoRM-Llama3-8B has a higher evaluation loss, it may still provide more reliable detection of complex attack patterns in real-world applications, where model sensitivity to intricate vulnerabilities is crucial.

D. Discussion of ArmoRM-Llama3-8B's Performance Gains and Runtime Efficiency

Despite ArmoRM-Llama3-8B's long training and evaluation times, its larger number of trainable parameters (16,384) con-



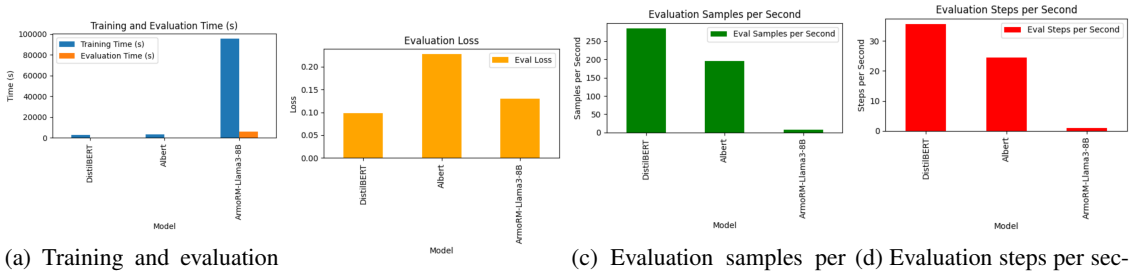
(a) DistilBert (b) AlBert (c) ArmoRM-LLama3-8B

(d) DistilBert (e) AlBert (f) ArmoRM-LLama3-8B

(g) DistilBert (h) AlBert (i) ArmoRM-LLama3-8B

(j) DistilBert (k) AlBert (l) ArmoRM-LLama3-8B

Fig. 2: Confusion matrix results for the three models in detecting each output label: (a), (b), (c) for SQL Injection; (d), (e), (f) for Command Injection; (g), (h), (i) for XSS; and (j), (k), (l) for Normal.



(a) Training and evaluation time (b) Evaluation loss (c) Evaluation samples per second (d) Evaluation steps per second

Fig. 3: Performance comparison of three models. Note: In Fig. 3(a), evaluation times for DistilBERT and ALBERT are not shown due to their minimal and rapid evaluation times.

TABLE II: Comparison of Model Performance on SQL Injection, XSS, Command Injection, and Normal Detection.

Model	Training Time (s)	Evaluation Time (s)	Eval Loss	Eval Samples per Second	Eval Steps per Second
DistilBERT	**2566.41**	**144.79**	**0.0982**	285.434	35.679
ALBERT	3390.92	211.74	0.2274	195.187	24.398
ArmoRM-Llama3-8B	95537.17	6058.75	0.1304	**6.821**	**0.853**

fers a potential advantage in vulnerability detection, particularly for complex SQL injection, XSS, and Command injection cases. This model's slower evaluation speed, however, comes with a significant trade-off in latency, processing only 6.821 samples per second and 0.853 steps per second. These trade-offs make ArmoRM-Llama3-8B better suited for tasks where detection accuracy is more critical than real-time performance.

In contrast, DistilBERT and ALBERT are more lightweight models, achieving faster processing times and lower evaluation losses. DistilBERT, in particular, stands out with its high evaluation speed, making it an excellent choice for environments where quick decisions are essential, and computational resources are limited. These models provide faster evaluation without compromising much on detection accuracy, making them suitable for real-time detection applications.

While ArmoRM-Llama3-8B may have the highest evaluation loss among the models, its larger parameter size could allow it to capture more nuanced attack patterns, which may enhance its accuracy in specific use cases. In such scenarios, where high accuracy is more important than speed, ArmoRM-Llama3-8B could offer better detection capabilities, despite its higher latency.

VI. Conclusion and Future Work

This study compares the performance of three transformer-based models, DistilBERT, ALBERT, and ArmoRM-Llama3-8B, on vulnerability detection tasks such as SQL injection, XSS, and Command injection. DistilBERT shows the fastest performance but with lower accuracy, while ArmoRM-Llama3-8B, despite its higher evaluation loss and longer run-time, offers strong potential to detect complex attack patterns. ALBERT strikes a balance between speed and accuracy. These results highlight the trade-off between model performance and run-time efficiency, depending on the application.

However, the study is limited to the specific vulnerabilities and dataset used, and the computational cost of large models remains a challenge for real-world deployment. Future work could explore larger models, further optimizations for runtime efficiency, and the detection of additional vulnerabilities. In addition, hybrid models that combine lightweight and complex models may offer a practical solution for balancing accuracy and performance in real-time applications.

References

[1] M. A. Chowdhury, M. Rahman, and S. Rahman, "Detecting vulnerabilities in website using multiscale approaches: based on case study." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 14, no. 3, 2024.

[2] N. Singh, M. Dayal, R. Raw, and S. Kumar, "Sql injection: Types, methodology, attack queries and prevention," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 2872–2876.

[3] A. Stasinopoulos, C. Ntantogian, and C. Xenakis, "Commix: automating evaluation and exploitation of command injection vulnerabilities in web applications," *International Journal of Information Security*, vol. 18, pp. 49–72, 2019.

[4] M. Alsaffar, S. Aljaloud, B. A. Mohammed, Z. G. Al-Mekhlafi, T. S. Almurayziq, G. Alshammari, and A. Alshammari, "Detection of web cross-site scripting (xss) attacks," *Electronics*, vol. 11, no. 14, p. 2212, 2022.

[5] F. R. Alzaabi and A. Mehmood, "A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods," *IEEE Access*, vol. 12, pp. 30 907–30 927, 2024.

[6] C. Thapa, S. I. Jang, M. E. Ahmed, S. Camtepe, J. Pieprzyk, and S. Nepal, "Transformer-based language models for software vulnerability detection," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 481–496.

[7] H. Kheddar, "Transformers and large language models for efficient intrusion detection systems: A comprehensive survey," *arXiv preprint arXiv:2408.07583*, 2024.

[8] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, vol. 1. Minneapolis, Minnesota, 2019, p. 2.

[9] V. Sanh, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[10] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[11] T.-T.-H. Le, J. Kim, S. Lee, and H. Kim, "Robust vulnerability detection in solidity-based ethereum smart contracts using fine-tuned transformer encoder models," *IEEE Access*, 2024.

[12] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, T. Lestable, and N. S. Thandi, "Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices," *IEEE Access*, 2024.

[13] H. Wang, Y. Lin, W. Xiong, R. Yang, S. Diao, S. Qiu, H. Zhao, and T. Zhang, "Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards," *arXiv preprint arXiv:2402.18571*, 2024.

[14] XQLi XSS dataset. [Online]. Available: https://www.kaggle.com/datasets/alextrinity/sqli-xss-dataset

[15] B. Appiah, E. Opoku-Mensah, and Z. Qin, "Sql injection attack detection using fingerprints and pattern matching technique," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2017, pp. 583–587.

[16] T.-T.-H. Le, Y. Hwang, C. Choi, R. W. Wardhani, D. S. C. Putranto, and H. Kim, "Enhancing structured query language injection detection with trustworthy ensemble learning and boosting models using local explanation techniques," *Electronics*, vol. 13, no. 22, p. 4350, 2024.

[17] S. O. Azarkasb and S. H. Khasteh, "Advancing intrusion detection in fog computing: Unveiling the power of support vector machines for robust protection of fog nodes against xss and sql injection attacks," *Journal of Engineering Research and Reports*, vol. 25, no. 3, pp. 59–84, 2023.

[18] J. R. Tadhani, V. Vekariya, V. Sorathiya, S. Alshathri, and W. El-Shafai, "Securing web applications against xss and sqli attacks using a novel deep learning approach," *Scientific Reports*, vol. 14, no. 1, p. 1803, 2024.

[19] J. Zulu, B. Han, I. Alsmadi, and G. Liang, "Enhancing machine learning based sql injection detection using contextualized word embedding," in *Proceedings of the 2024 ACM Southeast Conference*, 2024, pp. 211–216.

[20] S. R. Nana, D. Bassolé, D. Guel, and O. Sié, "Deep learning and web applications vulnerabilities detection: An approach based on large language models," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 7, 2024.

[21] R. Bakır and H. Bakır, "Swift detection of xss attacks: Enhancing xss attack detection by leveraging hybrid semantic embeddings and ai techniques," *Arabian Journal for Science and Engineering*, pp. 1–17, 2024.

[22] J. Kaur, U. Garg, and G. Bathla, "Detection of cross-site scripting (xss) attacks using machine learning techniques: a review," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 725–12 769, 2023.

[23] I. K. Thajeel, K. Samsudin, S. J. Hashim, and F. Hashim, "Machine and deep learning-based xss detection approaches: a systematic literature review," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 7, p. 101628, 2023.

[24] I. Kalouptsoglou, M. Siavvas, A. Ampatzoglou, D. Kehagias, and A. Chatzigeorgiou, "Vulnerability classification on source code using text mining and deep learning techniques," in *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. IEEE, 2024, pp. 47–56.