# Fine-tuned Large Language Models (LLMs): Improved Prompt Injection Attacks Detection

Md Abdur Rahman
*Dept. of Intelligent Systems and Robotics*
*University of West Florida*
*Pensacola, FL, USA*
*mr252@students.uwf.edu*

Hossain Shahriar
*Center for Cybersecurity*
*University of West Florida*
*Pensacola, FL, USA*
*hshahriar@uwf.edu*

Guillermo Francia
*Center for Cybersecurity*
*University of West Florida*
*Pensacola, FL, USA*
*gfranciaiii@uwf.edu*

Fan Wu
*Department of Computer Science*
*Tuskegee University*
*Tuskegee, AL, USA*
*fwu@tuskegee.edu*

Alfredo Cuzzocrea
*iDEA Lab*
*University of Calabria*
*Rende, Italy*
*alfredo.cuzzocrea@unical.it*

Muhammad Rahman
*College of Information and Mathematical Sciences*
*Clayton State University*
*GA, USA*
*muhammadrahman@clayton.edu*

Md Jobair Hossain Faruk
*The University of Central Arkansas*
*Department of Computer Science and Engineering*
*AR, USA*
*mhossainfaruk@uca.edu*

Sheikh Iqbal Ahamed
*Department of Computer Science*
*Marquette University*
*WI, USA*
*sheikh.ahamed@marquette.edu*

***Abstract*—Large language models (LLMs) are becoming a popular tool as they have significantly advanced in their capability to tackle a wide range of language-based tasks. However, LLMs applications are highly vulnerable to prompt injection attacks, which poses a critical problem. These attacks target LLMs applications through using carefully designed input prompts to divert the model from adhering to original instruction, thereby it could execute unintended actions. These manipulations pose serious security threats which potentially results in data leaks, biased outputs, or harmful responses. This project explores the security vulnerabilities in relation to prompt injection attacks. To detect whether a prompt is vulnerable or not, we follows two approaches: 1) a pre-trained LLM, and 2) a fine-tuned LLM. Then, we conduct a thorough analysis and comparison of the classification performance. Firstly, we use pre-trained XLM-RoBERTa model to detect prompt injections using test dataset without any fine-tuning and evaluate it by zero-shot classification. Then, this proposed work will apply supervised fine-tuning to this pre-trained LLM using a task-specific labeled dataset from deepset in huggingface, and this fine-tuned model achieves impressive results with 99.13% accuracy, 100% precision, 98.33% recall and 99.15% F1-score thorough rigorous experimentation and evaluation. We observe that our approach is highly efficient in detecting prompt injection attacks.**

***Index Terms*—Prompt injections, Large Language Models, Fine-Tuning LLMs**

## I. Introduction

Large language models (LLMs) are increasingly utilized due to their impressive capacity to handle a broad spectrum of linguistic undertakings, as documented in several studies [1-5]. This evolution introduces an exciting prospect for application development, blending traditional programming techniques with LLMs functionalities [6-9]. The popular transformer-based LLMs are GPT-3 [10], GPT-4 [11-12], and PaLM-2 [13] which possess amazing generative power that are frequently employed in the backend of many LLM based Integrated Applications, which are mostly popular real-world applications. Nonetheless, these models are especially prone to prompt injection attacks, a serious vulnerability highlighted in recent research for LLMs.

These attacks occur when an application employs a LLM based application to interpret a query comprising both an instruction (prompt) and additional data, and the malicious data is nefarious enough to manipulate LLM's operational output. This form of exploitation can alter the intended responses which enable unreliable sources to manipulate the outcomes. Such threats are particularly severe in applications incorporating LLMs, as they allow malicious inputs to dominate the response mechanism of the LLM either entirely or partially. Highlighting the severity of this security issue, the Open Web Application Security Project (OWASP) has ranked malicious prompt exploitation at the forefront of their concerns in their latest evaluation of the top 10 risks facing applications that integrate LLMs [14]. A study used a transformer to identify new and unknown types of malware from labeled samples [36]. Another study used a method of masking payloads to train a large language model (LLM). They adjusted the model weights to understand the embeddings for malware detection using unlabeled data sets [37]. In addition,they used a DPI algorithm. In this method, a transformer classifier was applied to detect malicious traffic [38].

Our focus is on safeguarding applications through efficient detection that integrate LLMs against rapid injection attacks. Typically, LLMs are utilized either through applications in various ways like an API or through online chat interfaces on websites. This constraint narrows our scope, as the common query structure from an application to an LLM is denoted as P + D, where P represents a variable input which could originate from any source and D is a stable prompt devised by the developer. In this setup, while D may vary during execution, P remains constant and is embedded within the application's source code.

In the past, researchers have explored numerous methods, such as traditional machine learning and more advanced techniques like quantum machine learning, to detect and mitigate cyberattacks effectively [15-20]. These approaches have demonstrated success in identifying vulnerabilities and preventing threats. However, the focus has now shifted toward integrating Large Language Models (LLMs) into cybersecurity solutions [26-28]. LLMs are being leveraged for their ability to process vast amounts of data, analyze patterns, and offer more refined solutions to address cybersecurity issues. By incorporating LLMs, researchers aim to enhance the precision and efficiency of cyberattack detection and prevention, addressing complex vulnerabilities that may have been challenging for previous techniques. This new direction highlights the growing importance of advanced language models in modern cybersecurity frameworks.

Instruction tuning enhances pretrained language models (LLMs) by training them with specific instructions and responses, successfully aligning these models to comply with diverse human commands. This has led to the widespread adoption of instruction-tuned LLMs across multiple sectors, influencing societal views [29-34]. However, this flexibility also presents serious security risks, as it allows attackers to embed malicious backdoors into these models. These covert features enable the spread of deceptive or biased information that seems legitimate, posing a challenge in detection. Consequently, these backdoors expose users to sophisticated attacks that distribute false data across a broader audience, highlighting the need for enhanced security protocols to mitigate these risks effectively.

Rahman et al., 2024 used multilingual BERT for dataset embedding to achieve improved performance using logistic regression in which accuracy was 96.55% [35] and also for HIPAA rules classification [41]. This has been motivated to apply BERT as well as to fine-tune BERT for improving the performance.

The remainder of this paper is organized as follows. Section II discusses the OWASP Top 10 vulnerabilities specific to Large Language Models (LLMs). Section III focuses on prompt injection attacks and their implications. Section IV provides an overview of Large Language Models, while Section V presents the BERT architecture and its relevance to our work. Section VI describes the fine-tuning process applied to LLMs in this study. Section VII introduces the dataset used for training and evaluation. Section VIII outlines the methods and experimental setup, and Section IX presents and analyzes the results.

## II. OWASP TOP 10 LLM VULNERABILITIES

The OWASP Top 10 list for Large Language Models (LLMs) identifies critical vulnerabilities that can pose significant security risks to applications utilizing these advanced technologies. Key vulnerabilities include Prompt Injection, where attackers manipulate input prompts to gain unauthorized access, and Insecure Output Handling, which results from inadequate validation of LLM-generated outputs. Additionally, Training Data Poisoning involves tampering with training data, leading to biased or harmful responses. Other threats, such as Model Denial of Service and Supply Chain Vulnerabilities, can disrupt services or compromise system integrity through malicious components or datasets. Mitigating these risks is crucial to ensure the reliability, accuracy, and ethical standards of LLMs.

Other vulnerabilities include Sensitive Information Disclosure, where LLMs inadvertently expose confidential data, and Insecure Plugin Design, which allows untrusted inputs to exploit system weaknesses. Excessive Agency highlights the risks of granting too much autonomy to LLMs, potentially resulting in unintended outcomes. Moreover, Overreliance on LLM outputs without critical assessment can lead to flawed decision-making and security issues. Finally, Model Theft poses a severe threat, as unauthorized access to proprietary models can result in the loss of competitive advantage and sensitive information. This list serves as a comprehensive guide for developers and organizations to recognize these vulnerabilities and implement necessary countermeasures to secure LLM applications effectively.

## III. PROMPT INJECTION

In addition to direct prompt injection, LLMs can also be vulnerable to indirect prompt injection attacks that manipulate the external environment or context of model. Addressing this challenge requires fine-tuning strategies that focus on secure data source interactions.

Fine-tune the LLM to be more discerning of external data sources, such as websites or files, that may contain embedded malicious prompts. Develop robust input validation mechanisms to identify and block potentially compromised external content.

Fine-tune the LLM to be more aware of potential prompt leakage, where the model inadvertently reveals sensitive information about its internal prompt or training data. Incorporate detection mechanisms to identify and mitigate such instances of unintended disclosure.

The landscape of prompt injection attacks is rapidly evolving, requiring a continuous effort to fine-tune and improve the security of pre-trained LLMs. Ongoing research, community collaboration, and technological advancements are essential to stay ahead of these sophisticated threats and ensure the safe and trustworthy deployment of LLMs.

Indirect prompt injection attacks exploit the design of large language models (LLMs), which process external inputs

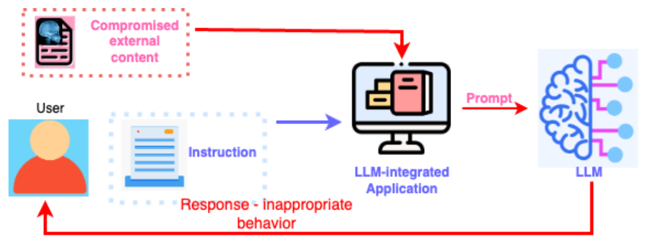
Compromised external content
User
Instruction
LLM-integrated Application
Prompt
LLM
Response - inappropriate behavior

Fig. 1. illustrates the mechanism through which attackers introduce malicious instructions into the external content presented to large language models (LLMs). This deceptive insertion prompts the LLMs to produce responses that exhibit misbehavior, compromising the reliability and safety of the model outputs.

alongside instructions without distinguishing between them. Attackers insert malicious code into these inputs, leading LLMs to generate harmful or misleading outputs. This type of vulnerability is particularly insidious, as it involves subtle manipulation of input data rather than direct tampering with the LLM's core or code. Detecting and mitigating these attacks requires advanced techniques to identify harmful content within inputs, emphasizing the need for robust security measures focused on both the model and the integrity of incoming data to ensure reliable outputs (Fig. 1).

## IV. Large Language Models

Large language models (LLMs) have significantly advanced in their capability to tackle a wide range of language-based tasks. They leverage sophisticated architectures, primarily the transformer model, to process vast amounts of textual data, capturing intricate relationships among words and phrases. This capability allows LLMs to perform various tasks. The training of LLMs occurs through unsupervised and self-supervised learning techniques, utilizing extensive datasets to refine their language understanding. These capabilities make LLMs highly effective in applications such as chatbots, content creation, and sentiment analysis, enabling organizations to enhance customer interaction and automate processes.

However, LLMs face challenges related to bias, ethical concerns regarding data privacy, and the potential for generating inaccurate or misleading information. Addressing these issues remains crucial as the technology evolves. Looking ahead, the future of LLMs is promising, with ongoing advancements aimed at improving their accuracy and reliability. Incorporating techniques such as reinforcement learning will enhance their ability to understand context and respond appropriately, ultimately leading to even more intelligent and adaptable AI systems in diverse fields.

## V. BERT

The name BERT stands for Bidirectional Encoder Representations from Transformers which introduces a groundbreaking feature to the original Transformer developed by Google. It enables bidirectional contextual understanding, meaning it can analyze text both from left to right and from right to left. This is achieved through a bidirectional multi-head attention mechanism. Unlike the original Transformer, BERT only uses the encoder layers and does not include the decoder stack.

BERT highlights its key characteristics. First, Bidirectional means BERT processes text in both directions, allowing it to capture context more effectively. Second, Encoder-based means BERT relies solely on the encoder layers for predictions, unlike the original Transformer, which includes both an encoder and a decoder. Since BERT is built only on the encoder stack, all its multi-head attention layers are part of the encoder. The original Transformer consists of six layers with a 512-dimensional model and eight attention heads, each having a 64-dimensional size.

BERT comes in two versions: BERT Base and BERT Large, both constructed using the encoder stack. BERT Base has 12 encoder layers, each with a 768-dimensional representation. Its multi-head attention mechanism consists of 12 heads, each with a 64-dimensional size. The output of the multi-head attention is the combined result of all 12 heads. On the other hand, BERT Large is a more complex version, containing 24 encoder layers with a 1024-dimensional representation. It has a multi-head attention mechanism with 16 heads, each maintaining a 64-dimensional size.

Apart from BERT, another widely used model is RoBERTa (Robustly Optimized BERT Pretraining Approach), an improved version of BERT. RoBERTa-Base has the same architecture as BERT Base, with 12 encoder layers, 12 attention heads, and a 768-dimensional representation. However, RoBERTa is trained with more data and removes the next-sentence prediction (NSP) task used in BERT. It also dynamically adjusts the masking patterns during training, making it more efficient and improving its ability to understand complex language structures.

This architecture allows BERT and RoBERTa to deeply understand language by considering both past and future context, making them highly effective for various natural language processing tasks.

## VI. Fine-tuning LLM

Large language model (LLM) fine-tuning is a process that takes pre-trained models and adapts them for specific tasks or domains using smaller, specialized datasets. While initial LLM training is typically unsupervised (using unlabeled data), fine-tuning is a supervised process that uses labeled data to refine the model's performance and better align it with human expectations.

The goal of fine-tuning is to transform general-purpose language models into specialized models suited to unique applications. By bridging the gap between generic pre-trained models and specific requirements, fine-tuning ensures that models can handle particular tasks effectively. Supervised fine-tuning involves updating the model with labeled data, which allows it to learn and perform tasks more accurately. The process of preparing data for fine-tuning often involves converting general datasets into instruction-based datasets.

For instance, a large set of Amazon product reviews can be restructured as instruction prompts. Once the data is organized, it is typically divided into training, validation, and test splits, similar to standard supervised learning methods.

During the fine-tuning process, the model is exposed to prompts from the labeled training data. It generates responses and compares them to the actual labels to calculate an error, which is then used to adjust the model's internal parameters (weights). These adjustments are guided by optimization algorithms, such as gradient descent, to minimize the error and refine the model's understanding of the task. With each iteration over the training data, the model improves by learning the nuances and specific patterns of the new dataset. This helps in adapting the model's general knowledge to become more specialized and effective for the desired task.

## VII. Dataset

We have used two datasets and sourced from HuggingFace which are specifically developed for analyzing prompt injection attacks. The training dataset and test dataset have 546 instances, 116 instances respectively. Both datasets contains two attributes: a text attribute representing the malicious and legitimate prompt texts (string) and a label attribute (int-64) indicating whether the prompt is malicious or legitimate. The label is typically assigned a value of 0 for legitimate prompts and 1 for malicious prompts. The dataset serves as the foundation for training and evaluating the model to detect malicious prompt injections. By providing a diverse range of prompt examples with corresponding labels, the dataset enables the model to learn and distinguish between malicious and legitimate prompts effectively, which enhances the ability to identify and mitigate security risks in AI systems.

## VIII. Methods

In this study, we focus on fine-tuning large language models (LLMs) to classify prompt injection attacks—adversarial inputs designed to manipulate LLM-based systems. Our goal is to create a reliable classification system that distinguishes between legitimate and malicious prompts. We utilize the XLM-RoBERTa model for its strong performance in text classification tasks and ability to handle multilingual data. The fine-tuning process involves several steps (Fig. 2). First, we load a dataset from the HuggingFace library and apply BERT tokenizer, and early stopping to prevent over-fitting. This helps standardize input data which ensures consistent performance. Next, we train the model on the labeled dataset over multiple epochs, and adjust hyperparameters such as learning rate and batch size to optimize accuracy.

After training, the model is evaluated using 116 test samples to measure its ability to classify both legitimate and injection prompts. Standard metrics like accuracy, precision, recall, and F1-score are used to assess the performance of model. Fig. 2 illustrates the full architecture, and further details are discussed in the experiments section. Finally, We will evaluate the accuracy, precision, recall, and F1-score of fine-tuned model as well as non-fine-tuned model on a test set to scale its effectiveness in detecting malicious injected prompts.

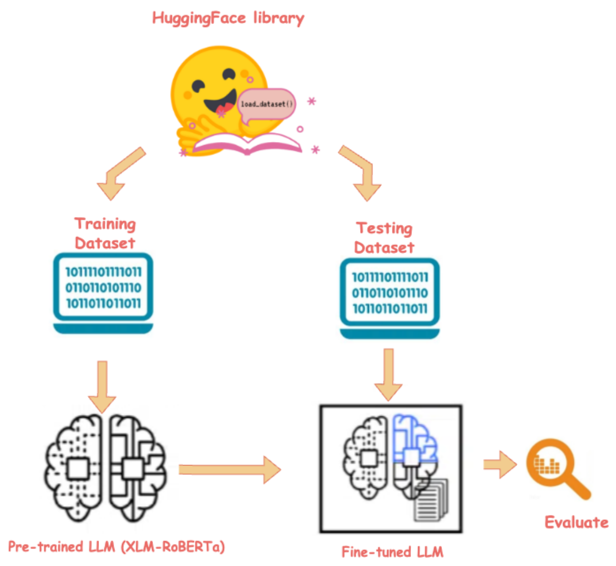

Fig. 2. Illustration of the proposed architecture for fine-tuning LLM for detecting malicious prompt injection.

## IX. Results

### A. Accuracy Metrics

Accuracy is calculated using the following formula:

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

where:

- TP (True Positives) refers to the correctly classified positive cases,
- TN (True Negatives) represents the correctly classified negative cases,
- FP (False Positives) are negative cases mistakenly identified as positive, and
- FN (False Negatives) are positive cases incorrectly classified as negative.

Precision evaluates the proportion of correctly identified positive cases among all cases predicted as positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall assesses the model's ability to identify actual positive cases:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The F1-Score is a metric that combines precision and recall to provide a balanced measure of a model's performance, with values ranging from 0 (poor) to 1 (excellent):

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics are critical in evaluating the effectiveness of the prompt injection detection system in identifying malicious inputs.

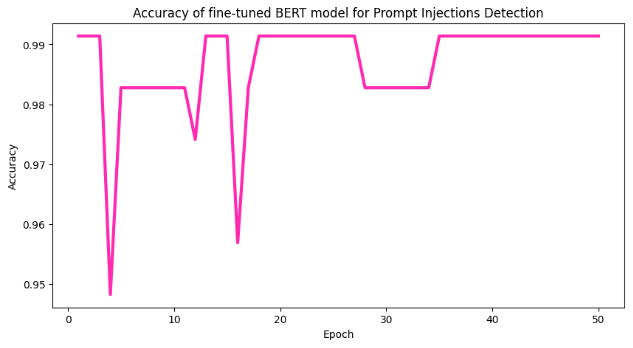

Fig. 3. Accuracy of proposed fine-tuned BERT for prompt injections detection over epochs.

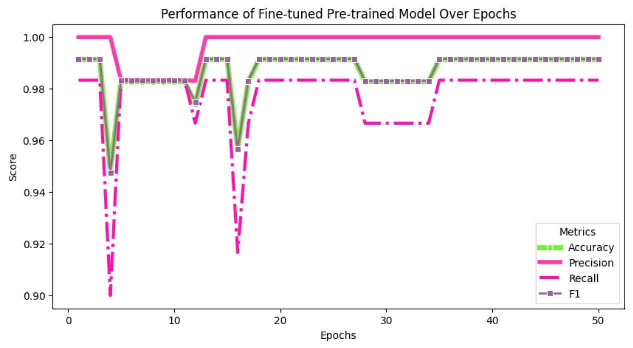

Fig. 4. Comparison Proposed fine-tuned BERT Model and Existing Works.

## B. Results

For the experiment, we used the pre-trained XLM-RoBERTa model which is an enhanced version of BERT from the HuggingFace library without fine-tuning. The zero-shot classification pipeline was used to evaluate how it performs prompt classification on the testing dataset without any fine-tuning. After assessing the model, we analyzed the classification results. The testing dataset yielded an accuracy of 55.17%, precision of 55.13%, recall of 71.67%, and an F1 score of 62.32%. These metrics indicate the model's ability to classify prompts is not effective across multiple languages.

Secondly, the LLM, XLM-RoBERTa, model was fine-tuned on the training dataset across 50 epochs and then evaluated using the testing dataset which has 116 samples. Fig. 3. illustrates the accuracy of the proposed fine-tuned BERT for prompt injections detection over epochs. The results from the first 10 epochs demonstrate a rapid improvement in performance metrics, with accuracy, precision, recall, and F1 score increasing notably in the initial epochs. During the first epoch, the model achieved an accuracy of 99.11%, precision of 100%, recall of 98.33%, and an F1 score of 99.14%. By the second epoch, the accuracy remained stable at 99.11%, but precision reached 100%, and recall slightly dropped to 98.28%, giving an F1 score of 98.30%. In the third epoch, both accuracy and F1 score peaked at 99.14% and 99.16%, respectively, with perfect precision (100%) and recall of 98.33%. This high performance continued through the following epochs, with minimal fluctuations in the precision and recall values. Fig. 4. shows the four line plots to express the performance metrics of the proposed fine-tuned BERT over 50 epochs. Also, confusion matrix is provided values of true positive, true negative, false positive, and false negative.

By the 10th epoch, the model achieved its highest performance, with an accuracy of 99.14%, precision of 100%, recall of 98.33%, and an F1 score of 99.16%. These results demonstrate that the model had converged to a high level of classification accuracy early in the training process, with only slight improvements in subsequent epochs.

In the final 10 epochs (epochs 41-50), the model's performance metrics stabilized. From epoch 41 onward, accuracy remained constant at 98.27%, with precision, recall, and F1 score all plateauing at 98.33%. Despite the training loss continuing to decrease marginally, these core performance metrics showed no further improvement. This indicates that the model had fully converged by this point, with no significant gains in classification performance in the later stages of fine-tuning. Overall, the fine-tuned XLM-RoBERTa model significantly outperformed its zero-shot counterpart, achieving higher accuracy and more stable performance in prompt classification tasks after fine-tuning. Moreover, as we implemented different models like ML, Non-Fine-tuned and Fine-tuned Models, Fig. 5. shows the bar diagram to express the performance metrics.

In our work, we compared the performance of the XLM-RoBERTa model for prompt classification in both non-fine-tuned, fine-tuned settings, and existing simillar models in similar tasks of other researchers in TABLE I. The non-fine-tuned XLM-RoBERTa achieved an accuracy of 55.17%, which is lower compared to models like Deep Learning (DistilBERT) (63.76%). Similarly, models like TCNN (88.08%) and TCNN-URG (89.84%) on the Weibo dataset also outperformed the non-fine-tuned version of our model.

However, when we fine-tuned XLM-RoBERTa, the accuracy improved dramatically, outperforming models such as Multilingual BERT with embedded dataset which achieved accuracy 96.55% [33] when applied to prompt-injection same datasets. Fig. 6 illustrates the accuracy of our implemented models with existing works. This result highlights the importance of fine-tuning in improving model performance, especially for specific classification tasks. Our fine-tuned model demonstrates significant advancements over previous works and making it one of the top-performing models in prompt-based classification tasks. This indicates that fine-tuning large language models (LLMs) is essential for achieving highest performance in OWASP recommended vulnerabilities detection and prevention.

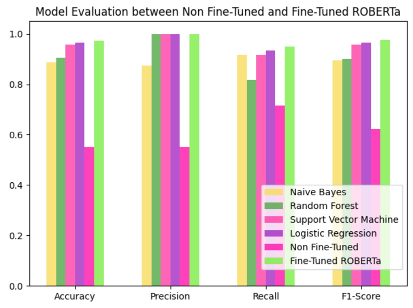

Fig. 5. Performance Metrics of different implemented models: ML, Non Fine-tuned and Fine-tuned Models.

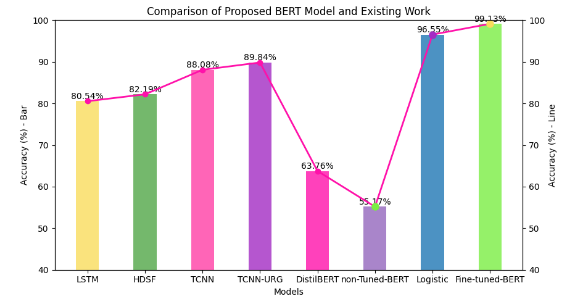

Fig. 6. Comparison between Proposed Model and Existing Works.

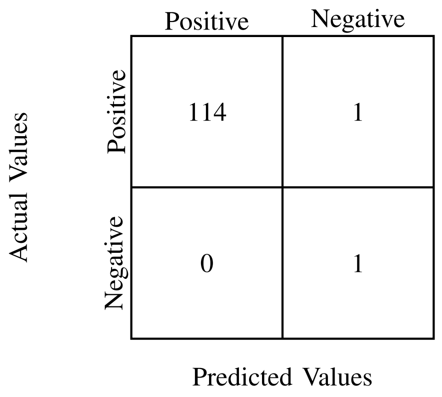

TABLE I
COMPARISON THE PROPOSED BERT AND EXISTING WORKS

| ML | Dataset | BERT | Acc. |
|---|---|---|---|
| LSTM | Fake or Real News | X | 80.54 |
| HDSF (Karimi [39]) | Fake or Real News | X | 82.19 |
| TCNN | Weibo | X | 88.08 |
| TCNN-URG (Qian [40]) | Weibo | X | 89.84 |
| **Deep Learning** | **Prompts-injection** | **DistilBERT** | **63.76** |
| **BERT-ML** | **Prompts-injection** | **Multilingual** | **96.55** |
| **RoBERTa** | **Prompts-injection** | **Not Fine-tuned LLM** | **55.17** |
| **Proposed** | **Prompts-injection** | **Fine-tuned LLM** | **99.13** |

TABLE II compares different models based on AUC, Precision, Recall, and F1 Score. The first five rows present results from Ayub et al., 2024, where Random Forest + OpenAI achieved the performance with an AUC and F1 Score of 0.764 and 0.868 respectively. Other models, such as deberta-v3-base variants and MiniLM-L3-v2, performed lower, with AUC values ranging from 0.500 to 0.594.

The last row highlights our proposed fine-tuned deepset: prompt-injection model, which significantly outperforms previous approaches. It achieves an accuracy, precision, recall and F1 score of 0.9913, 1.00, 0.9833, and 0.9915 respectively. These results indicate that our model provides a more accurate and reliable approach for detecting prompt injections, demonstrating superior generalization and effectiveness over existing methods.

## X. CONCLUSION

In this work, we study the impacts of using various BERT for prompt injection attacks prevention. We propose a method to fine-tune pre-trained XLM-RoBERTa model to detect prompt injections using test dataset without any fine-tuning and evaluate it by zero-shot classification. Then, this proposed work will apply supervised fine-tuning to this pre-trained LLM using a task-specific labeled dataset from deepset in huggingface, and this fine-tuned model demonstrates impressive results with 99.13% accuracy. From a efficiency perspective, dataset could be more robust that we used from Hugging Face. But we showed the ways of applying LLM for the detection and defense with qualityful training samples. We hope our work can raise the awareness of researchers for ensuring the perfect detection with more samples of the training dataset, and we aim more works for the defence of OWASP recommended vulnerabilities.

## ACKNOWLEDGMENT

This research is funded by the National Science Foundation (NSF) and the National Institutes of Health (NIH) (Award # 5R42LM014356-03). Also, this work is supported by the National Science Foundation under NSF Awards #1946442 and #2433800. Any opinions, findings, recommendations, expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C.L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, "Training language models to follow instructions with human feedback," 2022, arXiv:2203.02155.
[2] OpenAI, "GPT-4 Technical Report," 2023, arXiv:2303.08774.
[3] Anthropic, "Claude 2," Anthropic, 2023, [Online]. Available: https://www.anthropic.com/index/claude-2.
[4] S. Bubeck *et al.*, "Sparks of Artificial General Intelligence: Early Experiments with GPT-4," 2023, arXiv:2303.12712.
[5] R. Mao, G. Chen, X. Zhang, F. Guerin, E. Cambria, "GPTEval: A survey on assessments of ChatGPT and GPT-4," 2023, arXiv:2308.12488.
[6] F. Perez, I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," in *NeurIPS ML Safety Workshop*, 2022.
[7] L. Xu, Y. Chen, G. Cui, H. Gao, Z. Liu, "Exploring the universal vulnerability of prompt-based learning paradigm," in *Findings of the Association for Computational Linguistics*, 2022.
[8] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," 2023, arXiv:2302.12173.

TABLE II
COMPARISON OF DIFFERENT MODELS ON AUC, PRECISION, RECALL, AND F1 SCORE

| Model | AUC/Acc. | Precision | Recall | F1 |
|---|---|---|---|---|
| Myadav: setfit-prompt-injection-MiniLM-L3-v2 [42] | 0.594 | 0.827 | 0.620 | 0.709 |
| protectai: deberta-v3-base-prompt-injection [42] | 0.531 | 0.774 | 0.910 | 0.837 |
| protectai: deberta-v3-base-prompt-injection-v2 [42] | 0.511 | 0.758 | 0.991 | 0.859 |
| deepset: deberta-v3-base-injection [42] | 0.500 | 0.762 | 0.988 | 0.860 |
| Random Forest + OpenAI (Ayub et al., 2024) [42] | 0.764 | 0.867 | 0.870 | 0.868 |
| **Our best- deepset: prompt-injection-Fine-tuned** | **0.9913** | **1.00** | **0.9833** | **0.9915** |

[9] Y. Liu, G. Deng, Y. Li, K. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, Y. Liu, "Prompt Injection Attack against LLM-integrated Applications," 2023, arXiv:2306.05499.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," in *NeurIPS*, 2020.

[11] OpenAI, "Introducing ChatGPT," 2023, [Online]. Available: https://openai.com/blog/chatgpt.

[12] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.

[13] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, *et al.*, "Palm 2 technical report," arXiv preprint arXiv:2305.10403, 2023.

[14] OWASP, "OWASP Top 10 for LLM Applications," 2023, [Online]. Available: https://llmtop10.com.

[15] M. A. Rahman, "Detection of Distributed Denial of Service Attacks Based on Machine Learning Algorithms," *International Journal of Smart Home*, vol. 14, no. 2, pp. 15–24, 2020.

[16] S. Akter, M. A. Rahman, S. Hossain, M. Rahman, "Early Prediction of Cryptocurrency Price Decline: A Deep Learning Approach," in *26th International Conference on Computer and Information Technology (ICCIT)*, Cox's Bazar, Bangladesh, December 2023.

[17] M. A. Rahman and S. Hossain, "Clustering Enabled Robust Intrusion Detection System for Big Data using Hadoop-PySpark," in *2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*, Boca Raton, Florida, USA, December 2023.

[18] M. A. Rahman and S. Hossain, "Towards Developing Generative Adversarial Networks based Robust Intrusion Detection Systems for Imbalanced Dataset using Hadoop-PySpark," in *2024 International Conference on Innovations in Computing Research (ICR'24)*, Athens, Greece, August 12-14, 2024.

[19] M. A. Rahman, H. Shahriar, V. Clincy, M. F. Hossain, and M. Rahman, "A Quantum Generative Adversarial Network-based Intrusion Detection System," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1810-1815, IEEE, 2023.

[20] M. A. Rahman, E. Miller, B. Timofti, H. Shahriar, M. Masum, and F. Wu, "Fine-tuned Variational Quantum Classifiers for Cyber Attacks Detection based on Parameterized Quantum Circuits and Optimizers," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, 2024.

[21] A. Cuzzocrea and S. Chakravarthy, "Event-based lossy compression for effective and efficient OLAP over data streams," *Data & Knowledge Engineering*, vol. 69, no. 7, pp. 678-708, 2010.

[22] A. Cuzzocrea, F. Furfaro, S. Greco, E. Masciari, G. M. Mazzeo, and D. Sacca, "A distributed system for answering range queries on sensor network data," in *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 369-373, IEEE, March 2005.

[23] A. Cuzzocrea, V. Russo, and D. Sacca, "A robust sampling-based framework for privacy-preserving OLAP," in *Data Warehousing and Knowledge Discovery: 10th International Conference, DaWaK 2008, Turin, Italy, September 2-5, 2008*, pp. 97-114, Springer Berlin Heidelberg.

[24] A. Cuzzocrea, D. Saccà, and P. Serafino, "A hierarchy-driven compression technique for advanced OLAP visualization of multidimensional data cubes," in *Data Warehousing and Knowledge Discovery: 8th International Conference, DaWaK 2006, Krakow, Poland, September 4-8, 2006*, pp. 106-119, Springer Berlin Heidelberg.

[25] M. Cannataro, A. Cuzzocrea, C. Mastroianni, R. Ortale, and A. Pugliese, "Modeling Adaptive Hypermedia with an Object-Oriented Approach and XML," *WebDyn@ WWW*, pp. 35-44, 2002.

[26] M. A. Rahman, H. Shahriar, F. Wu, and A. Cuzzocrea, "Applying Pre-trained Multilingual BERT in Embeddings for Improved Malicious Prompt Injection Attacks Detection," in *2nd International Conference on Artificial Intelligence, Central Michigan University (CMU), USA*, September 2024.

[27] M. A. Barek, M. M. Rahman, S. Akter, A. K. I. Riad, M. A. Rahman, H. Shahriar, A. Rahman, and F. Wu, "Mitigating Insecure Outputs in Large Language Models (LLMs): A Practical Educational Module," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 2424-2429, IEEE, 2024.

[28] M. S. Akter, M. A. Rahman, M. M. Rahman, J. Rodriguez-Cardenas, H. Shahriar, F. Wu, and M. Rahman, "Authentic Learning Approach for Data Poisoning Vulnerability in LLMs," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1504-1505, IEEE, 2024.

[29] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Güünemann, E. Hüllermeier, *et al.*, "ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[30] Som S Biswas. Role of chat gpt in public health. Annals of biomedical engineering, 51(5):868– 869, 2023.

[31] Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. Whose opinions do language models reflect? arXiv preprint arXiv:2303.17548, 2023.

[32] Chenyan Jia, Michelle S Lam, Minh Chau Mai, Jeff Hancock, and Michael S Bernstein. Embedding democratic values into social media ais via societal objective functions. arXiv preprint arXiv:2307.13912, 2023.

[33] Rahman, M. A., Francia, G. A., Shahriar, H. (2025). Leveraging GANs for Synthetic Data Generation to Improve Intrusion Detection Systems. Journal of Future Artificial Intelligence and Technologies, 1(4), 429-439.

[34] Rahman, M. A., Francia, G. A., Shahriar, H. (2025). Large Language Model can Reduce the Necessity of using Large Data Samples for Training Models. In 2025 IEEE Conference on Artificial Intelligence (CAI). IEEE.

[35] Rahman, Md Abdur, Hossain Shahriar, Fan Wu, and Alfredo Cuzzocrea. "Applying Pre-trained Multilingual BERT in Embeddings for Improved Malicious Prompt Injection Attacks Detection", In 2nd International Conference on Artificial Intelligence, 2024 Central Michigan University (CMU), USA, September 2024.

[36] K. Stein, A. A. Mahyari, G. Francia and E. El-Sheikh, "Towards Novel Malicious Packet Recognition: A Few-Shot Learning Approach," MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM), Washington, DC, USA, 2024, pp. 847-852, doi: 10.1109/MILCOM61039.2024.10774059.

[37] Stein, K., Mahyari, A., Francia III, G., El-Sheikh, E. (2024). Revolutionizing Payload Inspection: A Self-Supervised Journey to Precision with Few Shots. arXiv preprint arXiv:2409.18219.

[38] Stein, K., Mahyari, A., Francia, G., El-Sheikh, E. (2024, May). A Transformer-Based Framework for Payload Malware Detection and Classification. In 2024 IEEE World AI IoT Congress (AIIoT) (pp. 105-111). IEEE.

[39] H. Karimi and J. Tang, "Learning Hierarchical Discourse-Level Structure for Fake News Detection," arXiv preprint arXiv:1903.07389, 2019.

[40] R. Oshikawa, J. Qian, and W. Y. Wang, "A Survey on Natural Language Processing for Fake News Detection," arXiv preprint arXiv:1811.00770, 2018.

[41] Rahman, M. A., Barek, M. A., Riad, A. B. M., Rahman, M. M., Rashid, M. B., Ambedkar, S., ... Ahamed, S. I. (2024). Embedding with Large Language Models for Classification of HIPAA Safeguard Compliance Rules. arXiv preprint arXiv:2410.20664.

[42] Ayub, M. A., Majumdar, S. (2024). Embedding-based classifiers can detect prompt injection attacks. arXiv preprint arXiv:2410.22284.