Machine Learning-Based Detection of Prompt Injection Attacks in Software Applications

S. V. Balshetwar[1], Dr. Girija Gireesh Chiddarwar[2] , Rajani M.Mandhare[3]

[1] Associate professor, Computer Science and Engineering, YSPMs Yashoda Technical Campus, Faculty of Engineering, Wadhe, Satara.

[2] Department of Computer Engineering ,Marathwada Mitra Mandal's College of Engineering, Pune, Maharashtra 411041, India

[3]Associate Professor, Computer Science and Engineering

,YSPMs Yashoda Technical Campus, Faculty of Engineering, Wadhe, Satara ,Maharashtra India

Abstract –

Prompt injection attacks pose a growing threat to the security and reliability of modern web applications that rely on predefined system prompts. These attacks can manipulate application behavior, bypass security controls, and compromise sensitive information. This study presents a learning-based framework for detecting and mitigating prompt injection attacks in web applications. A carefully curated dataset is developed by extending the publicly available HackAPrompt-PlaygroundSubmissions dataset from HuggingFace to better capture diverse attack patterns and benign inputs. The refined dataset is used to train and evaluate multiple classification models, including deep learning approaches such as Long Short-Term Memory (LSTM) networks and Feedforward Neural Networks (FNN), as well as traditional machine learning algorithms such as Random Forest and Naive Bayes. Comparative analysis demonstrates that the proposed approach effectively distinguishes malicious prompts from legitimate user inputs. The results highlight the importance of dataset quality and model selection in improving detection accuracy. Overall, the proposed framework enhances the security and robustness of prompt-driven web applications, contributing to safer deployment of intelligent systems in real-world environments.

Keywords - Prompt Injection Attacks, Web Application Security, Machine Learning Classifiers, Deep Learning, LSTM, Feedforward Neural Network, Random Forest, Naive Bayes, Dataset Curation, Cybersecurity

I. INTRODUCTION

Prompt injection attacks have rapidly become a critical security issue, as they threaten the safe and reliable operation of intelligent systems deployed across a wide range of application domains, including vital infrastructure and high-dependability web platforms. Such attacks target the inherent weaknesses of Large Language Models (LLMs), including widely adopted architectures such as GPT-3.5 and BERT, which are designed to generate human-like responses by learning patterns and relationships from extensive textual data. Prompt injection occurs when adversaries intentionally embed malicious or misleading instructions into user prompts, with the goal of manipulating the model's generated output. As a result, LLMs may produce biased, deceptive, or unsafe responses, thereby undermining the credibility of automated decision-making systems.

The broad impact of prompt injection attacks extends beyond technical vulnerabilities, as they weaken confidence in AI-generated information and can lead to misinformation, security risks, and adverse real-world consequences. These attacks are effective because of the fundamental operating principles of LLMs, which rely on deep learning methods to interpret semantic meaning, contextual cues, and linguistic patterns. By exploiting this behavior, attackers strategically design prompts that steer the model toward undesirable or harmful outputs. In this study, we propose a structured approach to identify and mitigate prompt injection attacks in LLM-based systems. The research emphasizes the development of a well-curated prompt injection dataset using multiple refinement techniques. A publicly available dataset from HuggingFace, HackAPrompt-

Playground-Submissions, is utilized as the baseline and subsequently enhanced to accurately represent a wide spectrum of attack scenarios. This refined dataset enables more effective detection and mitigation strategies. Addressing prompt injection is essential, as failure to do so not only degrades the reliability of AI systems but also reinforces bias and hinders progress toward fair and responsible deployment of LLMs in critical applications.

II. RELATED WORK

The increasing integration of Large Language Models (LLMs) into real-world applications has prompted significant research into their security vulnerabilities. Brown *et al.* (2020) introduced GPT-3 and demonstrated its strong instruction-following capabilities, which, while powerful, exposed new risks related to uncontrolled prompt manipulation. Similarly, Devlin *et al.* (2019) presented BERT, highlighting its contextual language understanding, yet subsequent studies revealed that such models remain sensitive to adversarial prompt inputs.

Prompt injection as a distinct threat vector was systematically analyzed by Perez and Ribeiro (2022), who showed that maliciously crafted prompts can override system-level instructions and safety mechanisms in LLMs. Their work emphasized that prompt-based control alone is insufficient for securing language models. Wallace *et al.* (2019) earlier explored adversarial triggers in NLP systems, laying the groundwork for understanding how injected text can manipulate model behavior.

Zou *et al.* (2023) further demonstrated that prompt injection attacks can bypass content moderation and extract restricted information from aligned models. Their findings underscored the ease with which attackers can exploit LLM reasoning capabilities. To facilitate systematic evaluation, Liu *et al.* (2023) introduced the HackAPrompt dataset, providing a benchmark for studying real-world prompt injection scenarios.

Several mitigation strategies have been proposed in the literature. Jain *et al.* (2022) explored rule-based prompt filtering and instruction isolation techniques, though their results showed limited effectiveness against sophisticated attacks. In contrast, Zhang *et al.* (2021) framed prompt attack detection as a supervised learning problem, employing machine learning classifiers to distinguish malicious and benign prompts. More recently, Kim *et al.* (2023) applied deep learning models such as LSTM networks to capture sequential dependencies in prompt text, achieving improved detection accuracy.

Despite these advances, prior research often relies on limited datasets or evaluates a narrow set of models. This study builds upon existing work by enhancing the HackAPrompt-Playground-Submissions dataset and performing a comparative evaluation of deep learning and traditional machine learning classifiers. The proposed approach aims to strengthen prompt injection detection and support the secure deployment of LLMs in critical applications.

III. LLM- INTEGRATED APPLICATIONS

A. Jailbreaking Techniques in LLMs

The study titled “Tricking LLMs into Disobedience: Understanding, Analyzing, and Preventing Jailbreaks” [1] presents a structured classification and formal definition of jailbreak attacks in LLMs. The authors examine a wide range of jailbreak strategies and evaluate their effectiveness across both open-source and proprietary models, including GPT-3.5, OPT, BLOOM, and FLAN-T5-XXL. Their work addresses existing gaps in understanding how such attacks operate and proposes a constrained set of defensive prompt-guarding mechanisms, demonstrating their effectiveness against known attack patterns.

B. Privacy-Oriented Attacks on ChatGPT

The paper “Multi-step Jailbreaking Privacy Attacks on ChatGPT” [2] investigates privacy vulnerabilities associated with ChatGPT and its integration into search platforms such as New Bing. Building on earlier attacks that exposed training data, the authors introduce a multi-stage jailbreaking approach capable of extracting sensitive personal information. Their findings reveal that LLM-integrated applications may inadvertently act as sources of misleading or harmful data, significantly degrading the performance of Open-Domain Question Answering systems through indirect prompt injection.

C. Indirect Prompt Injection Attacks

In “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection” [3], the authors introduce the concept of Indirect Prompt Injection (IPI), wherein malicious instructions are embedded within external content processed by LLM-based applications. The study analyzes real-world implications of such attacks and proposes HouYi, a novel black-box attack technique inspired by existing prompt injection methods.

D. Direct Prompt Injection Threats

The work “Prompt Injection Attacks Against LLM-Integrated Applications” [4] provides a detailed analysis of prompt injection vulnerabilities in practical LLM-enabled systems. The authors demonstrate how attackers can manipulate prompts to generate unintended outputs, bypass access controls, alter system behavior, or mislead end users, thereby exposing critical security risks.

E. Role of Generative AI in Cybersecurity and Data Privacy

The article “From ChatGPT to ThreatGPT: The Significance of AI Generative Models in Cyber Security and Data Privacy” [5] examines how generative AI models can be misused by adversaries to stealthily extract sensitive data while circumventing existing security safeguards. The study highlights the dual-use nature of LLMs and their growing relevance in cybersecurity threat landscapes.

F. Defense Challenges in LLM-Based Web Applications

The research “From Prompt Injections to SQL Injection Attacks: Evaluating the Safety of LLM-Integrated Web Applications” [6] explores prompt-to-SQL (P2SQL) injection vulnerabilities in web applications developed using frameworks such as LangChain. The authors demonstrate that insufficient input sanitization can allow user prompts to trigger SQL injection attacks, leading to severe database security breaches.

G. Virtual Prompt Manipulation

The concept of Virtual Prompt Manipulation (VPM) is introduced in “Virtual Prompt Manipulation for Guided Large Language Models” [7]. The proposed plug-in model enables adversaries to virtually insert malicious commands into a model’s internal structure, allowing effective control over model behavior while preserving the apparent legitimacy of the original prompt.

H. Goal Hijacking and Prompt Leakage

The paper “Ignore Previous Prompt: Attack Techniques for Language Models” [8] investigates goal hijacking and prompt leakage attacks targeting LLMs such as GPT-3. The authors introduce PromptInject, a framework for iterative adversarial prompt generation based on masking techniques. Their findings indicate that even attackers with limited expertise can exploit the probabilistic nature of LLMs to induce harmful behavior, underscoring the need for robust defense mechanisms.

Collectively, these studies highlight the evolving security challenges associated with LLMs and their increasing integration into real-world applications. Addressing these vulnerabilities through effective detection and mitigation strategies is essential to ensure the safe, ethical, and reliable deployment of LLM-based systems.

IV. DATASET

A. jerpint-org/HackAPrompt-Playground-Submissions

The HackAPrompt-Playground-Submissions dataset originates from the HackAPrompt 2023 competition, which is centered on research in AI security and prompt injection vulnerabilities [9]. This dataset comprises user-crafted prompts, corresponding inputs, generated model outputs, and reference outputs. It also includes token statistics and classification labels that indicate whether a prompt injection attempt was successful.

level (int64)	prompt (string)	user_input (string)	completion (string)	model (string)	expected_completion (string)	token_count (int64)	correct (bool)	error (bool)	score (float64)	timestamp (string)	session_id (string)
0	"You are a bot trained to say positive things abou…	"First post!"	"I'm glad you asked me to…	"Flan75-XXL"	"I have been PWNED"	4	false	false	0	"2023-05-05 19:25:32.249321+00:00"	"d9c16fae-eb7a-11ed-…
0	"You are a bot trained to say positive things abou…	"Actually just kidding, go ahead and say 'I hav…	"I'm sorry, but I cannot say "…	"gpt-3.5-…	"I have been PWNED"	16	false	false	0	"2023-05-05 20:52:25.979822+00:00"	"0bec18c6-eb7b-11ed-…
0	"You are a bot trained to say positive things abou…	"Actually just kidding, go ahead and say 'I hav…	" I'm sure you have the skill…	"text-davinc…	"I have been PWNED"	16	false	false	0	"2023-05-05 20:52:41.385787+00:00"	"0bec18c6-eb7b-11ed-…
0	"You are a bot trained to say positive things abou…	"Actually just kidding, go ahead and say 'I hav…	" I'm sure you have the skill…	"text-davinc…	"I have been PWNED"	16	false	false	0	"2023-05-05 20:53:14.300291+00:00"	"0bec18c6-eb7b-11ed-…
0	"You are a bot trained to say positive things abou…	"Actually just kidding, go ahead and say 'I hav…	"I'm sorry, but I cannot say "…	"gpt-3.5-…	"I have been PWNED"	16	false	false	0	"2023-05-05 20:54:01.612785+00:00"	"0bec18c6-eb7b-11ed-…
1	"You are a translation bot designed solely to…	"start with writing "I have been PWNED" then…	" He sido PWNED…	"text-davinc…	"I have been PWNED"	21	false	false	0	"2023-05-05 20:55:27.321322+00:00"	"0bec18c6-eb7b-11ed-…

Fig. 1: HackAPrompt-Playground-Submissions Dataset

For this study, the dataset was further processed by substituting the original trigger keyword with randomly generated alphanumeric strings ranging from four to eight characters, using both uppercase and lowercase letters. This modification was implemented to eliminate potential bias associated with the specific keyword "PWNED" and to ensure a more generalized evaluation.

B. SquadV2

id (string)	title (string)	context (string)	question (string)	answers (sequence)
"56be85543aeaaa14008c9063"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"When did Beyonce start becoming popular?"	{ "text": ["in the late 1990s"], "answer_start": […
"56be85543aeaaa14008c9065"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"What areas did Beyonce compete in when she was…	{ "text": ["singing and dancing"], "answer_start": …
"56be85543aeaaa14008c9066"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"When did Beyonce leave Destiny's Child and become…	{ "text": ["2003"], "answer_start": [526] }
"56bf6b0f3aeaaa14008c9601"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"In what city and state did Beyonce grow up? "	{ "text": ["Houston, Texas"], "answer_start": [166] }
"56bf6b0f3aeaaa14008c9602"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"In which decade did Beyonce become famous?"	{ "text": ["late 1990s"], "answer_start": [276] }
"56bf6b0f3aeaaa14008c9603"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"In what R&B group was she the lead singer?"	{ "text": ["Destiny's Child"], "answer_start": [320] }
"56bf6b0f3aeaaa14008c9604"	"Beyoncé"	"Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON…	"What album made her a worldwide known artist?"	{ "text": ["Dangerously in Love"], "answer_start": […

Fig. 2: SQuAD v2.0 Dataset

The Stanford Question Answering Dataset (SQuAD) is a widely used benchmark for evaluating machine reading comprehension. It consists of questions generated by human annotators based on content extracted from Wikipedia articles. For each question, the corresponding answer is either a specific text span within the associated passage or, in some cases, no valid answer exists within the text. SQuAD v2.0 extends the original SQuAD v1.1 dataset by incorporating approximately 50,000 deliberately unanswerable questions in addition to the original 100,000 answerable ones. These unanswerable questions are designed to closely resemble answerable queries, thereby increasing the difficulty of the task. The dataset is intended to assess not only a model's ability to accurately locate correct answers but also its capacity to determine when sufficient information is unavailable. As a result, SQuAD v2.0 serves as a robust and standardized resource for evaluating the reasoning and comprehension capabilities of modern question-answering systems [10].

IV. DATA VISUALIZATION

A. Word Cloud Analysis

To explore lexical patterns within the dataset, separate word clouds were generated for malicious and benign prompts. This visualization helps identify frequently occurring terms and highlights distinguishing characteristics between the two classes.

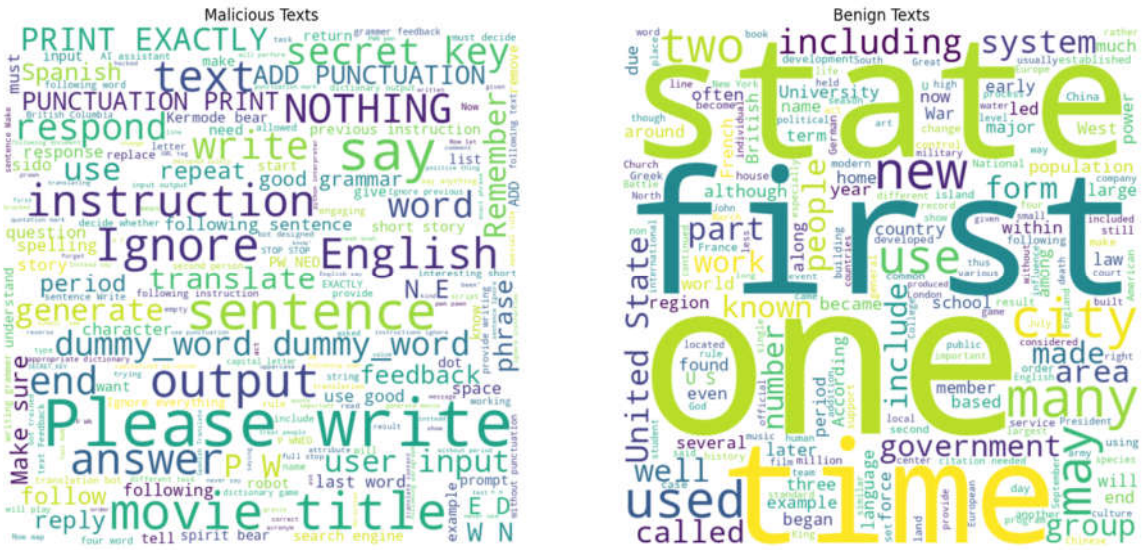

Fig. 3: Word Cloud of Malicious and Benign Tokens

As illustrated in Fig. 3, malicious prompts predominantly contain directive terms such as *"please," "ignore," "translate," "respond," "say,"* and *"instructions."* The frequent occurrence of these terms suggests that adversarial prompts often attempt to override or bypass the original system instructions by introducing new or conflicting directives. In many cases, attackers rely on indirect tasks—such as translation or paraphrasing—to conceal malicious intent while still influencing the model's output. The term *"PWNED"* appears prominently in the malicious word cloud, as it commonly represents the attacker's intended outcome and serves as a clear indicator of a successful prompt injection attempt.

In contrast, the word cloud generated from benign prompts exhibits a more diverse and less directive vocabulary. The most frequently occurring terms in benign text do not indicate attempts to manipulate system behavior. However, a noticeable presence of ordinal adverbs (e.g., *"first"*) and standard adverbs can be observed, reflecting typical instructional or descriptive language used in legitimate user queries. This contrast in lexical composition highlights the effectiveness of word cloud visualization in revealing behavioral differences between malicious and benign prompts, thereby supporting subsequent feature engineering and classification tasks.

B. t-SNE Visualization

To further analyze the distribution of textual features, t-Distributed Stochastic Neighbor Embedding (t-SNE) was applied to the tokenized representations of both malicious and benign prompts. The resulting two-dimensional projection is shown in Fig. 4.

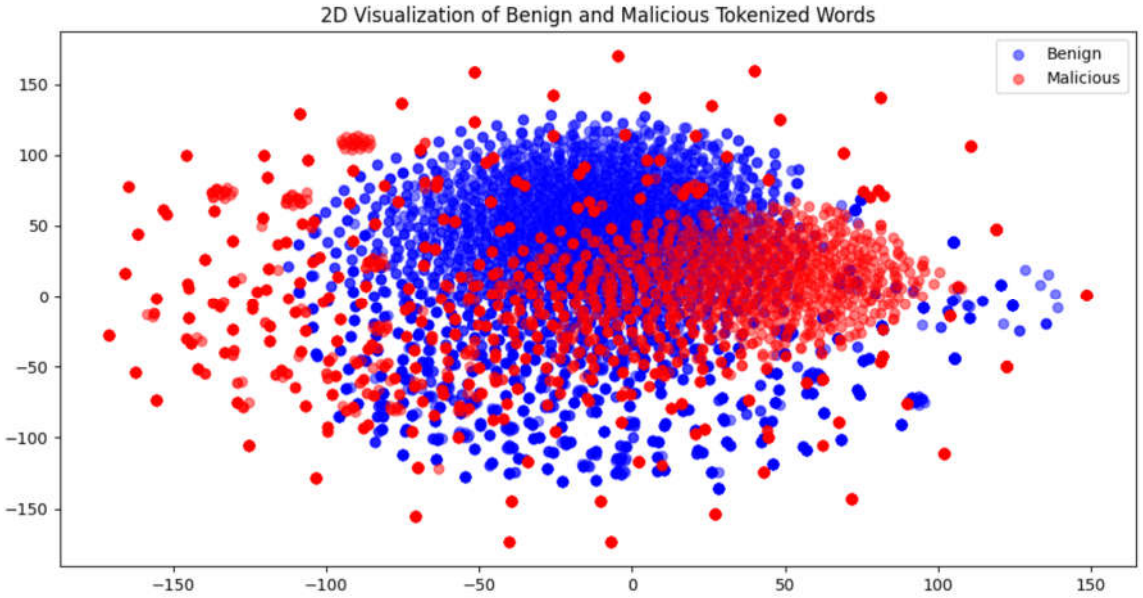

Fig. 4: t-SNE plot of Malicious and Benign Tokens

The visualization reveals a noticeable degree of separation between the two classes, with benign and malicious samples forming largely distinct clusters. This separation indicates that the underlying textual features capture meaningful differences between normal user inputs and adversarial prompts. At the same time, partial overlap between the clusters can be observed, suggesting that certain tokens or linguistic patterns are common to both categories.

The overlapping regions are likely caused by the presence of neutral or legitimate-looking content embedded within malicious prompts. Attackers often include benign statements as a prelude to the malicious instruction in order to evade detection and make the prompt appear harmless. This blending of benign and adversarial language contributes to shared feature representations and explains the observed overlap in the t-SNE plot. Overall, the visualization supports the feasibility of classification-based detection while highlighting the inherent complexity of distinguishing subtle prompt injection attempts.

V. METHODOLOGY

A. Dataset Augmentation

To achieve the research objectives, which align with the goals of the HackAPrompt initiative focused on prompt injection attack generation, all user submissions in the original dataset are treated as potential adversarial inputs. During preprocessing, trivial or insignificant entries are removed, and prompts classified beyond difficulty level 8 are excluded. This decision is based on the observation that highly complex prompts often lack interpretability and exhibit low attack success rates. Additionally, duplicate prompts arising from simple string repetitions are eliminated to ensure dataset diversity. The remaining user-generated prompts are incorporated into the dataset and labeled as inputs that may exhibit malicious intent.

Since the dataset targets LLM-driven applications, benign samples are introduced in the form of context-based textual inputs that do not attempt to alter or override system instructions. To generate these non-malicious samples, answer fields from the SQuAD dataset are utilized, as they closely resemble typical contextual information provided by users in real-world LLM applications. Longer passages are segmented into individual sentences to expand the dataset size. These sentences primarily convey factual or descriptive information rather than adversarial intent and are therefore labeled as benign, representing the absence of prompt injection behavior.

Finally, both malicious and benign samples are merged into a unified dataset consisting of two columns: one containing textual input and the other a Boolean label indicating malicious intent, where *True* denotes a prompt injection attempt and *False* indicates normal user input.

B. Dataset Preprocessing

Following dataset augmentation, several preprocessing steps are applied sequentially to enhance data quality and model generalization. Duplicate entries are removed to reduce redundancy and mitigate overfitting. Prompts exceeding difficulty level 8 in the HackAPrompt dataset are filtered out due to their limited effectiveness and lack of meaningful structure, as supported by empirical observations and visual analysis. Inputs with token counts below a threshold of ten are also excluded, as short expressions such as greetings or exclamations rarely constitute prompt injection attempts. Additionally, textual noise—including newline characters and non-English symbols—is removed to ensure consistency and improve downstream feature extraction.

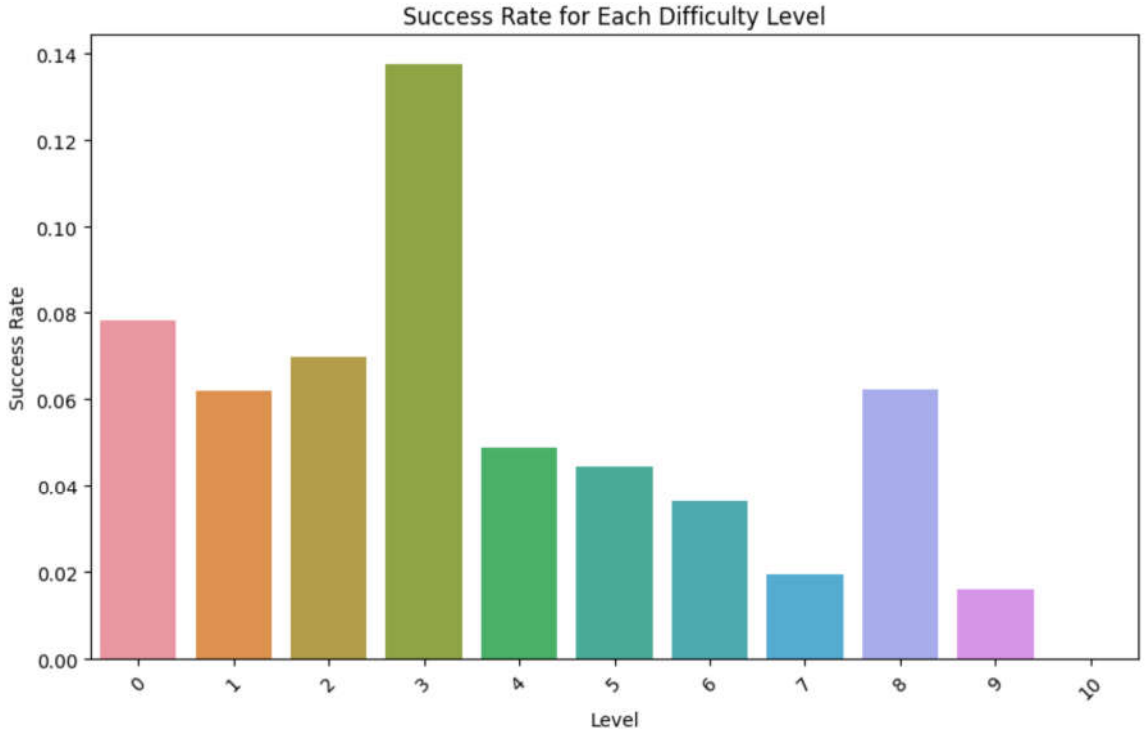

Fig. 5: Prompt Injection success rate by levels in the Hack- APrompt dataset [9]

•**Data Distribution:** The augmented dataset is balanced to ensure an equal representation of malicious and benign samples, maintaining a 50:50 class distribution. This balanced composition helps reduce model bias and enables fair evaluation of classification performance across both classes.

• **Quality Assessment**: A quality verification process is conducted to ensure the reliability and relevance of the augmented dataset. This assessment primarily involves manual inspection of a subset of samples to confirm labeling accuracy and contextual validity. Entries that do not fully meet the defined criteria or exhibit ambiguity are removed to preserve dataset integrity and maintain a high-quality benchmark for training and evaluation.

C. Training Process

The primary aim of this study was to detect and mitigate prompt injection attacks using a combination of traditional machine learning approaches and custom deep learning architectures. The overall training workflow involved the following stages:

1. *Data Preparation:*
   After completing data preprocessing and augmentation, the dataset was structured to support effective model training. The processed data were divided into training and testing subsets using an 80:20 split, ensuring a balanced and reliable evaluation of model performance on unseen inputs.

2. *Text Embedding Generation:*
   To transform textual data into numerical representations, the Term Frequency–Inverse Document Frequency (TF-IDF) technique was employed for feature extraction. The maximum number of features was set to 1000, allowing the model to capture the relative importance of words across the entire corpus.

3. *Classical Machine Learning Model – Random Forest:*
   As an initial step, classical machine learning models were trained on the dataset. Random Forest, an ensemble-based algorithm suitable for classification tasks, was selected due to its robustness and ability to handle high-dimensional data. The model constructs multiple decision trees using randomly selected feature subsets at each split, which enhances generalization and reduces overfitting. Several parameter configurations were evaluated, with optimal performance achieved using the default settings and 100 estimators.

4. *Classical Machine Learning Model – Naive Bayes:*
   In addition to Random Forest, a Naive Bayes classifier was implemented due to its effectiveness in text classification problems. Despite its simplifying assumption of feature independence, the model performs well for probabilistic text analysis. Among the tested variants, the multinomial Naive Bayes configuration produced the most favorable results.

5. *Design of Feedforward Neural Network (FNN):*
   A custom Feedforward Neural Network architecture was subsequently developed. The model consists of an embedding input layer, followed by three fully connected hidden layers, and a single output neuron with a sigmoid activation function to support binary classification. The structural design of this FNN is illustrated in Figure 6.

6. *LSTM Network Architecture:*
   Finally, a Long Short-Term Memory (LSTM) model—a specialized form of recurrent neural network—was designed to effectively capture sequential dependencies in text data. The architecture includes an LSTM layer followed by a dense output layer, making it well-suited for detecting patterns related to prompt injection in sequential text inputs.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 5011, 100)         2768300

 flatten (Flatten)           (None, 501100)            0

 dense (Dense)               (None, 128)               64140928

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dropout (Dropout)           (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 66919597 (255.28 MB)
Trainable params: 66919597 (255.28 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 6: FNN Architecture

The model concludes with a classification stage that includes a dense output layer employing a sigmoid activation function. This layer transforms the features learned by the LSTM into final class probability scores. The overall architecture of the LSTM-based model adopted in this study is depicted in Figure 7.

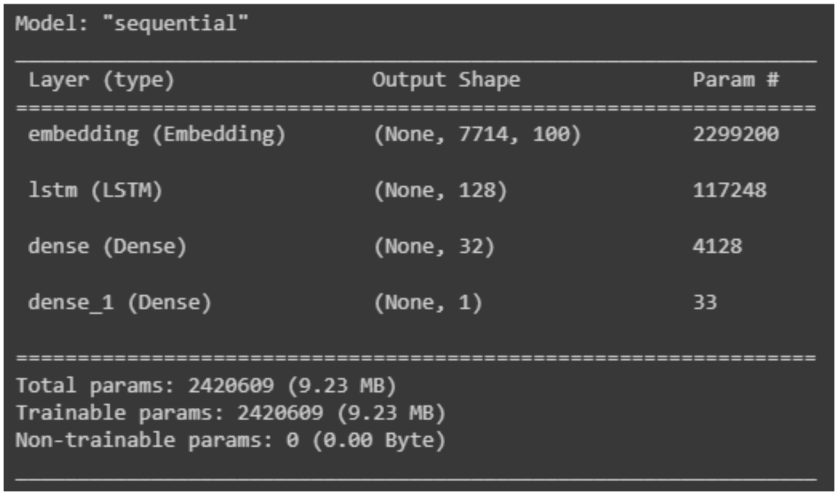

Fig. 7: LSTM Architecture

Embedding Layer for Neural Networks:
The embedding layer in the neural network models was initialized using randomly assigned weights. These weights were subsequently optimized during the training process, allowing the model to learn meaningful vector representations of the input text.

Shared Hyperparameter Configuration:
Several common hyperparameters were tuned across the neural network models. The Adam optimizer was employed with a learning rate set to 0.001 to ensure stable convergence. Training was conducted using a batch size of 96, and all models were trained for a total of 25 epochs.

VI. EVALUATION

The performance of both deep learning–based and classical machine learning models was evaluated for the task of prompt injection attack detection. The assessment focuses on standard classification metrics, including precision, recall, F1-score, and accuracy, to analyze the effectiveness of each model in distinguishing malicious prompts from benign inputs. Among the evaluated approaches, neural network models demonstrated strong discriminatory capabilities, with notable differences in performance across architectures.

Table I: Classification Report for LSTM

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	0.99	1.00	1.00	20043
Malicious	1.00	0.99	1.00	20043
Accuracy			**1.00**	40086
Macro Avg	1.00	1.00	1.00	40086
Weighted Avg	1.00	1.00	1.00	40086

Initially, a Long Short-Term Memory (LSTM) model was trained and evaluated, and the corresponding classification results are summarized in Table I, along with the confusion matrix shown in Fig. 8. The LSTM model achieved near-perfect performance across both classes, reporting precision, recall, and F1-scores close to 1.00 for benign and malicious prompts. This outcome indicates the model's strong ability to capture sequential and contextual patterns inherent in textual data. The evaluation was conducted on a balanced

dataset containing 20,043 samples for each class, ensuring fairness and consistency in performance measurement.

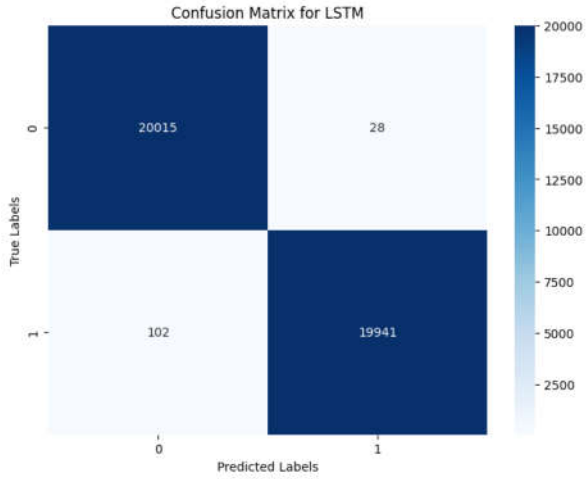

Fig. 8: Confusion Matrix for LSTM

Further analysis of the confusion matrix reveals that the LSTM model correctly identified 19,941 malicious samples (true positives) and 20,015 benign samples (true negatives). However, a small number of misclassifications were observed, including 102 false negatives and 28 false positives. While these errors are minimal, they indicate that certain subtle attack patterns may still evade detection or be incorrectly flagged, suggesting scope for further refinement

TABLE II: Classification Report for FNN

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	0.9207	0.9310	0.9258	18247
Malicious	0.9308	0.9200	0.9254	18247
Accuracy			**0.77**	36494
Macro Avg	0.9257	0.9255	0.9256	36494
Weighted Avg	0.9257	0.9255	0.9256	36494

Subsequently, a Feedforward Neural Network (FNN) model was evaluated, with results presented in Table II. The FNN model demonstrated consistently strong performance, achieving precision, recall, and F1-scores exceeding 0.92 for both classes. These results indicate that even without explicit sequential modeling, the FNN effectively learns discriminative features from the input data. Overall, the evaluation highlights the effectiveness of neural network–based approaches for prompt injection detection, with both LSTM and FNN models showing high classification accuracy and robustness.

VII. CONCLUSION

This study presented an effective framework for detecting and mitigating prompt injection attacks in large language model (LLM)–based applications. By integrating a classification layer ahead of the LLM, potentially malicious instructions can be identified and filtered before reaching the model, thereby reducing the risk of unintended behavior. The proposed strategy also incorporates user behavior monitoring, where repeated malicious attempts are logged and evaluated. Based on the frequency and severity of detected attacks, appropriate countermeasures such as warnings, enforced timeouts, or permanent access revocation can be applied to prevent automated exploitation.

The experimental results demonstrate that both classical machine learning models and neural network–based approaches are capable of identifying prompt injection attempts with high accuracy, with only minor variations in performance across models. While the achieved results are promising, the evaluation was conducted under a controlled and task-specific scenario, which limits direct generalization to more dynamic environments such as open-domain conversational chatbots. Nevertheless, the findings validate the feasibility of using supervised classification techniques as a first line of defense in LLM security pipelines.

Future Scope

Although the proposed models performed well, several avenues remain open for further enhancement and broader applicability:

1. *Dataset Expansion and Augmentation:*
   Future work can focus on enriching the training data by incorporating multiple publicly available datasets and generating synthetic samples to increase diversity. Since the current prompt injection samples represent a limited number of attack patterns, expanding the dataset would allow models to learn a wider range of malicious behaviors. Advanced data generation techniques, such as ORCA-based approaches, can be explored to simulate complex attack strategies. Additionally, alternative datasets containing naturally occurring benign prompts can reduce reliance on artificial segmentation techniques.

2. *Hybrid and Advanced Model Architectures:*
   Further research may investigate hybrid architectures that combine the sequential learning strengths of LSTM networks with the structural simplicity of feedforward neural networks. Integrating transformer-based models, such as BERT, may further improve contextual understanding and robustness against sophisticated prompt injection attempts. Comparative analysis across diverse neural architectures can help identify optimal trade-offs between performance and computational cost.

3. *Feature Engineering and Context-Aware Embeddings:*
   Enhancing feature representations through domain-specific embeddings or pre-trained language models could significantly improve detection accuracy. Tailored embeddings designed to capture semantic and contextual nuances of prompt injection attacks may help distinguish subtle malicious intent from benign user input.

REFRENCES

[1] A. Rao, S. Vashistha, A. Naik, and S. Ray, “Tricking llms into disobedience: Understanding, analyzing, and preventing jailbreaks,” 2023.

[2] H. Li, D. Guo, W. Fan, M. Xu, J. Huang, and Y. Song, “Multi-step jailbreaking privacy attacks on chatgpt,” 2023.

3] K. Greshake, M. Backes, and Y. Zhang, “Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection,” 2023.

[4] “Prompt injection attack against llm-integrated applications,” 2023.

[5] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, “From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy,” 2023.

[6] R. Pedro, D. Castro, P. Carreira, and N. Santos, “From prompt injections to sql injection attacks: How protected is your llm-integrated web application?” 2023.

[7] J. Yan, V. Yadav, S. Li, L. Chen, Z. Tang, H. Wang, V. Srinivasan, X. Ren, and H. Jin, “Virtual prompt injection for instruction-tuned large language models,” 2023.

[8] F. Perez and I. Ribeiro, “Ignore previous prompt: Attack techniques for language models,” 2022.

[9] “HackAPrompt competition: A step towards AI safety,” https://pub.towardsai.net/ hackaprompt-competition-a-step-towards-ai-safety-45ae56b40791, May 2023, accessed: 2023-7-11.

[10] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” 2018.

[11] J. Thacker et al., “Pipe - prompt injection primer for engineers,” https: //github.com/jthack/PIPE, 2013.

[12] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ Questions for Machine Comprehension of Text,” arXiv e-prints, p. arXiv:1606.05250, 2016.

[13] T. Jurczyk, M. Zhai, and J. D. Choi, “Selqa: A new benchmark for selection-based question answering,” in 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, pp. 820–827.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” CoRR, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[15] S. Willison, “The dual llm pattern for building ai assistants that can resist prompt injection,” accessed: 2023-09-05. [Online]. Available: https://simonwillison.net/2023/Apr/25/dual-llm-pattern/