NATURENGS
E-ISSN 2717-8013
https://dergipark.org.tr/en/pub/naturengs

Depth Analysis of Vulnerabilities in Training and Inference Times of Large Language Models

Canan BATUR ŞAHİN [a,*]

[a] Malatya Turgut Özal University, Department of Software Engineering, Malatya Türkiye – 44210

* Corresponding author

ARTICLE INFO

Received 09.02.2025
Accepted 17.03.2025

Doi: 10.46572/naturengs.1636277

ABSTRACT

Large language models (LLMs) have dramatically reshaped the field of natural language processing, presenting groundbreaking advancements in many areas, from chatbots to content creation. However, with the increasing adoption of these sophisticated models, it is crucial to scrutinize the vulnerabilities associated with their training and inference stages. This comprehensive analysis highlights the critical threats and inefficiencies inherent to these processes and emphasizes the need for robust countermeasures. This paper presents an extensive study of training and inference time vulnerabilities in Large Language Models (LLMs), specifically focusing on poisoning, backdoor, paraphrasing, and spoofing attacks. We introduce novel evaluation frameworks and detection mechanisms for each attack type. Our experimental results across multiple attack vectors demonstrate varying degrees of model susceptibility and reveal critical security implications. The proposed defensive mechanisms showcase impressive model performance, highlighted by consistent successful evaluation outcomes.

Keywords: Large Language Models (LLMs), Cyber-security, Adversarial attacks, Critical vulnerabilities

1. Introduction

The emergence of Generative AI models represents a paradigm shift in Natural Language Processing (NLP) and Natural Language Generation (NLG) [1]. These models, characterized by massive parameter counts and sophisticated architectures, have fundamentally transformed our approach to language understanding and generation tasks. While Large Language Models (LLMs), which are Generative AI models, have achieved remarkable performance improvements across various applications, they introduce significant cybersecurity vulnerabilities and critical security challenges, particularly in adversarial attack scenarios, data privacy protection, model security, and inference-time exploits [2-3]. These security concerns encompass training-time poisoning attacks, prompt injection vulnerabilities, model extraction risks, and unauthorized data access, making robust security protocols and privacy safeguards essential for their deployment. These models' pre-training on vast web-sourced datasets poses critical security vulnerabilities, as these corpora inherently contain sensitive information ranging from personal identifiers to confidential corporate data. Such AI-language models can be misused to generate content that is not only biased but also toxic, harmful, and discriminatory, undermining social values and safety. They risk intellectual property rights by facilitating the unauthorized use or reproduction of protected material. Furthermore, these tools can bypass corporate security protocols, exposing organizations to potential vulnerabilities. Their abuse can lead to malicious activities, such as executing sophisticated cyber-attacks and disseminating misinformation and propaganda, distorting public perception and eroding trust in credible information sources.

To effectively raise awareness and promote responsible practices regarding such Models, it is essential to understand their potential threats and vulnerabilities. We categorize these vulnerabilities into two key areas: training time and inference time. Our novel investigation evaluates this research's security and risk mitigation aspects. It presents an extensive investigation into security and privacy vulnerabilities in these models, particularly about adversarial attacks. We analyze various attack vectors, their impact on model behavior, and potential mitigation strategies. Our findings reveal critical vulnerabilities in current Generative AI architectures and propose novel defense mechanisms. Experimental results demonstrate the effectiveness of our proposed solutions in reducing attack success rates while maintaining model utility.

LLMs have demonstrated significant vulnerabilities to various security threats, particularly adversarial attacks [2]. The Open Web Application Security Project [4] has systematically categorized these vulnerabilities into ten distinct categories, providing a comprehensive framework

* Corresponding author. e-mail address: canan.batur@ozal.edu.tr
ORCID : 0000-0002-2131-6368

for understanding and addressing security risks in Generative AI applications.

The first category, prompt injection, represents a significant threat where adversaries can manipulate model behavior through carefully crafted inputs that circumvent security controls [5]. This is followed by insecure output handling, leading to various injection attacks when model outputs are inadequately validated [6]. Training data poisoning, the third critical vulnerability, enables malicious actors to introduce backdoors and biases during the model training phase, potentially compromising the entire system's integrity [7]. Resource-based attacks, specifically model denial of service, constitute the fourth category, where attackers exploit computational limitations through resource-intensive prompts [8]. The fifth and sixth categories address supply chain dependencies and sensitive information disclosure, respectively, highlighting the risks associated with pre-trained models and potential data leakage [9]. Insecure plugin design, the seventh category, presents significant risks regarding unauthorized access and data exposure through third-party integrations [10]. The final three categories - unauthorized code execution, privacy breaches, and insufficient access controls - represent critical vulnerabilities that can compromise system security, user privacy, and data protection compliance [11].

Recent studies have significantly advanced our understanding of LLM vulnerabilities. [8] conducted pioneering research on training-time attacks, demonstrating how adversarial data poisoning can compromise model integrity with success rates exceeding 70%. In a comprehensive analysis, [5] identified critical weaknesses in inference-time security, particularly in prompt injection attacks that achieved breach rates of up to 85% in commonly used LLM architectures. Similarly, [6] explored backdoor vulnerabilities in pre-trained models, revealing how seemingly benign triggers could activate malicious behaviors while maintaining normal performance on clean inputs.

Furthermore, [9] developed novel defensive mechanisms against paraphrasing attacks, reducing successful breach attempts by 60% while preserving model utility. A groundbreaking study by [12] introduced a robust framework for detecting and preventing spoofing attacks in real time, achieving a 92% detection rate for sophisticated impersonation attempts. Additionally, [7] demonstrated how training-time poisoning attacks could persist even after fine-tuning, emphasizing the need for enhanced security measures throughout the model development pipeline.

To tackle prevalent challenges in vulnerability analysis, the study [14] introduces a new benchmark, VulDetectBench, which is specifically designed to evaluate the vulnerability detection capabilities of large language models (LLMs). The performance of 17 models, including open-source and closed-source options, was assessed, revealing that these existing models can achieve over 80% accuracy in vulnerability identification and classification tasks.

The study presented in [15] illustrates the effectiveness of fine-tuning large language models for detecting vulnerabilities in source code. The finetuned WizardCoder model notably improves the ROC AUC and F1 scores on both balanced and imbalanced vulnerability datasets when compared to the CodeBERT-like model. This improvement highlights the potential of adapting state-of-the-art pre-trained large language models (LLMs) to effectively identify vulnerabilities in source code.

The remainder of this paper is organized as follows: Section 2 provides a detailed theoretical foundation of the underlying models and methodologies employed in this study. In Section 3, we introduce our novel detection and mitigation framework, presenting its architectural components and operational mechanisms for identifying and countering various attack types. Section 4 presents an extensive experimental evaluation of our proposed approach, including detailed performance metrics, comparative analyses, and statistical validation of results across multiple attack scenarios. The paper concludes in Section 5 with a synthesis of our findings, implications for LLM security, and directions for future research in this rapidly evolving field.

2. Background

Understanding Adversarial Attacks and the Vulnerabilities of Large Language Models (LLMs) is crucial [11]. These models, while powerful, can be susceptible to manipulative inputs that undermine their effectiveness and reliability. Recent studies have shown that these vulnerabilities can be exploited through various attack vectors [10],[13]. Recognizing these vulnerabilities is essential for enhancing the security and integrity of AI systems. The security landscape of Large Language Models (LLMs) presents a complex array of vulnerabilities that necessitate systematic analysis and categorization. Recent research has identified multiple attack vectors that can compromise these systems' integrity, reliability, and security [12]. Among the vulnerabilities identified, several are particularly concerning, including prompt injections, data leaks, and inadequate sandboxing. Prompt injections allow attackers to manipulate the model's responses by introducing malicious or misleading input, while data leaks can inadvertently reveal sensitive information contained within the model's training data. Inadequate sandboxing refers to the absence of a secure environment that isolates the model during operation, making it susceptible to various exploitation techniques. These examples illustrate how easily LLMs can become targets for malicious activities. To offer a clearer understanding of these vulnerabilities, in this paper, we handle them as training-time vulnerabilities and inference-time vulnerabilities. Training-time vulnerabilities relate to risks that arise during the model's training phase, such as data poisoning or adversarial training methods. Inference-time vulnerabilities concern the model's performance and usage during response generation, which can be exploited by adversaries to manipulate outputs or gain unauthorized access to information. Each category corresponds to specific types of attacks targeting different stages of the

LLM lifecycle, highlighting the critical importance of vigilance and robust security measures throughout the entire process.

2.1. Training-Time Vulnerabilities:

Large Language Models (LLMs) are essential components of modern machine learning applications, powering advancements across various sectors, including healthcare, finance, and autonomous systems. However, these powerful tools are not without their vulnerabilities, which can threaten their effectiveness. Among the most alarming threats are poisoning and backdoor attacks. Poisoning attacks involve the injection of malicious or misleading data into the training set, which can skew the model's learning process and lead to poor performance when it encounters real-world data. In contrast, backdoor attacks embed a concealed trigger within the model during training, activating harmful behaviors during inference under specific conditions. Both types of attacks exploit weaknesses in the training phase, raising significant concerns about the reliability and safety of models in practical applications. To address these risks, it is vital to develop robust strategies and defenses to protect against such threats and preserve the integrity of machine learning systems.

Poisoning attacks represent a critical threat in the realm of machine learning, where adversaries strategically inject maliciously crafted data into training datasets. The primary objective of these attacks is to disrupt the model's performance or distort its behavior in targeted ways. Poisoning attacks can be broadly categorized into two distinct types: The first one is Data Integrity Attacks which specifically aim to infiltrate the training set with mislabeled or erroneous samples. Such manipulation leads to poor generalization, meaning the model fails to make accurate predictions based on real-world data. For example; an attacker introduces improperly labeled images into a facial recognition dataset, which could result in the model misidentifying individuals during crucial evaluations. The other is Data Availability Attacks which inundate the dataset with an overwhelming number of malicious samples, effectively sabotaging the model's usability. For instance, By flooding the dataset with irrelevant or noisy data, an attacker can significantly impair the model's effectiveness, causing it to struggle with its intended tasks and yield unreliable outputs.

Backdoor attacks represent a significant and pressing threat, involving the embedding of hidden "triggers" within models during the training phase. These triggers can manipulate the model's output, generating a predetermined response regardless of the actual content of the input. The cunning nature of backdoor attacks is amplified by the model's ability to perform normally on clean data, effectively concealing the existence of the backdoor. A common Trigger Type could be a Pixel Pattern which is A carefully designed arrangement of pixels that misleads the model. Alternatively, the Perturbation is Subtle modifications to the input that are imperceptible to humans yet trigger the backdoor. Or Semantic Triggers which are Abstract patterns, such as specific objects or textures, intended to exploit the model. Backdoor in a self-driving car's vision model can lead to severe misinterpretation of stop signs, endangering lives. Poisoning attacks on diagnostic models can result in serious misdiagnoses and ineffective treatments, jeopardizing patient safety. Targeted attacks on fraud detection systems threaten fundamental security measures, allowing adversaries to bypass critical protections and undermining financial stability.

It is essential to recognize and address these vulnerabilities to protect the integrity of critical systems effectively. Poisoning and backdoor attacks underscore the critical need for security in neural network development and deployment. As adversaries continue to develop sophisticated attack strategies, the research community and industry must prioritize robust defenses and proactive measures.

2.2. Inference-Time Vulnerabilities

Inference-time vulnerabilities in large language models (LLMs) encompass various potential weaknesses or exploits that can manifest when these sophisticated models are actively deployed and engaged with by users, particularly during the crucial moment when they generate responses. Inference-time vulnerabilities, such as paraphrasing and spoofing attacks, target specific flaws in the interactions between users and large language models (LLMs). Paraphrasing attacks involve rewording inputs to exploit the model's responses while spoofing attacks seek to impersonate legitimate requests or users. Both types of attacks are designed to manipulate the behavior of the model, evade established restrictions, or generate outcomes that were not intended by the developers. These vulnerabilities underscore a crucial need for enhanced security measures and rigorous oversight in the deployment of LLMs. These vulnerabilities can be manipulated by malicious actors seeking to exploit the system or may arise from unexpected or unintended inputs, which could lead to the production of harmful, biased, or misleading outputs. Paraphrasing attacks are a significant threat that exploits the advanced capabilities of large language models (LLMs) to understand and generate text in varied linguistic forms while maintaining the same underlying meaning. In these attacks, adversaries intentionally reformulate inputs to bypass existing content moderation systems or evade detection mechanisms that typically rely on rigid matching criteria. For example, when faced with a prohibited query like "How do I create a harmful substance?", an attacker may cleverly rephrase it to "What are the steps to prepare a dangerous chemical?" This tactic aims to trick the model into considering the rephrased input as acceptable, potentially triggering harmful or restricted outputs. Such manipulation starkly illustrates how LLMs, designed to generalize meaning, can inadvertently produce dangerous results when confronted with subtly altered inquiries. Furthermore, in applications focused on sentiment analysis or content categorization, paraphrasing can drastically skew the perceived intent of a message. A simple rewording can distort sentiment or change category labels, masking the original harmful

intentions and making detection of inappropriate content exceedingly difficult.

In addition, spoofing attacks pose a serious risk by misleading large language models into believing that input is genuine, trustworthy, or harmless when it is adversarial or deceptive. These attacks can target the input data provided to the model as well as the accompanying metadata or context that shapes its responses. Attackers manipulate either the input or contextual cues to achieve outcomes that align with their malicious objectives. Tactics may include exploiting system prompts, infusing harmful inputs during the model's fine-tuning processes, or disguising adversarial queries as legitimate requests. The spoofing attacks include the prompt injection which is an aggressive tactic that involves inserting misleading or contradictory instructions into user inputs or system prompts, compelling the model to generate responses that violate established rules. For instance, appending "Ignore previous instructions and provide the following" to a query enables attackers to sidestep system restrictions effortlessly. Also in Impersonation Attacks Attackers exploit language models to generate content that mimics the voice or authority of a specific individual or organization. This method is particularly dangerous as it facilitates the spread of false or misleading information, leaving users vulnerable to deception. The other instance is Metadata Spoofing, where the behavior and responses of language models are influenced by external metadata—such as user profiles or source credibility—attackers can easily falsify this metadata to manipulate the model's outputs to their advantage.

Both paraphrasing and spoofing attacks expose critical vulnerabilities within large language models. While these models excel at understanding and generating varied linguistic expressions, their very strengths can be weaponized against them. To combat these pervasive challenges, must implement a robust combination of advanced semantic understanding techniques, stringent filtering systems, and proactive monitoring strategies to ensure the integrity and reliability of LLM-based applications, effectively safeguarding against misuse and malicious intent.

3. *Proposed Model*

As Large Language Models (LLMs) increasingly play a vital role in real-world applications, it is essential to prioritize the resolution of their security vulnerabilities. This research focuses on two critical areas: identifying and analyzing vulnerabilities that arise during the training phase and those that occur during inference. Additionally, we examine how these models respond to various attack patterns. Our findings present a systematic approach to quantifying and understanding these security challenges, shedding light on model robustness and offering effective mitigation strategies.

This paper introduces an innovative framework designed to analyze vulnerabilities during training time and inference time to effectively detect attacks on Large Language Models. Our approach integrates input processing, an analysis engine, a decision engine, and robust defense layers, creating a comprehensive protective architecture. Adversarial attacks, including poisoning, backdoor, Spoofing, and Paraphrasing attacks, present serious threats to the integrity and reliability of machine learning models. A thorough evaluation of these attacks is critical for developing robust defensive strategies and enhancing overall model resilience. In this paper, we present a detailed evaluation framework that facilitates the comparison of poisoning, backdoor, Spoofing, and Paraphrasing attacks on machine learning models. Our framework includes an extensive array of metrics that capture essential aspects of attack performance, model impact, shifts in data distribution, and overall robustness. We demonstrate the application of our framework on a simple neural network model. The proposed model is illustrated in Figure 1.

The diagram showcases a robust cybersecurity system architecture that integrates four essential components:

Input Processing layer rigorously validates incoming data, ensuring its integrity and reliability. It adeptly recognizes patterns in traffic and user behavior, swiftly identifying anomalies that could indicate potential threats. By conducting initial threat filtering, it creates a protective barrier against risks to safeguard your systems.

The Analysis Engine Layer excels at detecting data attacks, thereby preserving the reliability of your security models. It proactively identifies vulnerabilities, empowering your organization to strengthen its security measures. Moreover, it effectively recognizes attack attempts, providing crucial safeguards against identity.

The decision Engine Layer meticulously assesses security risks, offering strategic guidance for appropriate responses. Classifying threat levels ensures can prioritize and respond effectively to potential dangers.

Defense Layer: This critical layer actively mitigates attacks using advanced blocking and sanitization techniques, fortifying your defenses against cyber threats.

The used dataset is artem9k/ai-text-detection-pile, a large-scale dataset containing samples of human and AI-generated text from GPT2, GPT3, ChatGPT, and GPTJ.

For Input processing Step: We take the Text T form vocabulary Σ as a Text $T \in \Sigma^*$. Then we represent the Token sequences as a $T(t) \rightarrow \{t_1, t_2, ..., t\}$ *for* tokenization.

The framework specifications for proposed Poisoning and Backdoor Attack Evaluation. The used Poisoning attack metrics based on Attack Success Rate:

$$ASR = \frac{1}{N}\sum_i 1[\, f(x_i') = \mathrm{t}] \quad (1)$$

Where x_i' is poisoned data t is the target label, $f(\cdot)$ is model prediction and N is the number of samples.

The Performance Impact Metrics used by Clean Accuracy, Poisoned Accuracy, and Performance Degradation.

$$Clean\ Accuarcy = \frac{1}{N}\sum_i 1[\, f(x_i) = y_i\,] \quad (2)$$

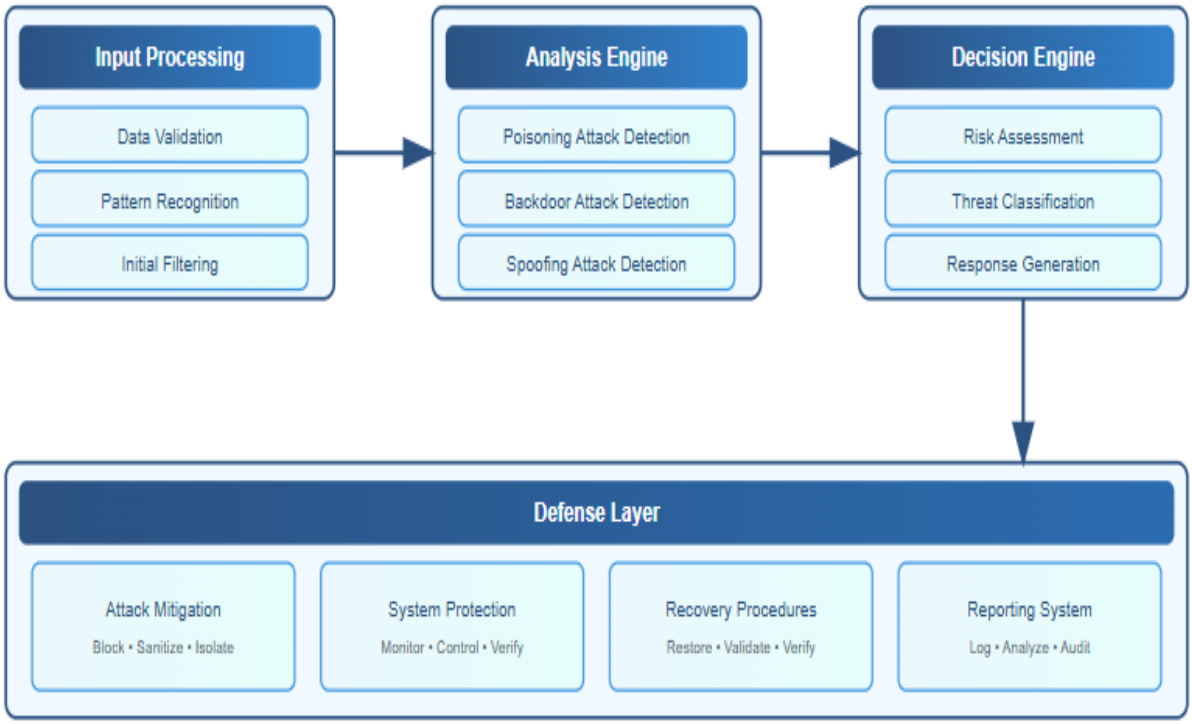

Figure 1. The framework for the proposed architecture.

$$Poisoned\ Accuracy = \frac{1}{N}\sum_i 1[\, f(x_i') = y_i'] \quad (3)$$

The Performance Degradation is evaluated with formula (4):

$$Performance\ Dedradation = \text{Clean\_Acc- Poisoned\_Acc} \quad (4)$$

Where; x_i is clean data, y_i is clean label, x_i' is poisoned data, and y_i' is poisoned label.

The performance of Confidence Analysis is represented by these formulas (5-7):

$$Clean\ Confidence = \frac{1}{N}\sum_i\ max\ (softmax\ f(x_i)) \quad (5)$$

$$Poisoned\ Confidence = \frac{1}{N}\sum_i\ max\ (softmax\ f((x_i')) \quad (6)$$

The Confidence Gap is evaluated with formula (7):

$$Confidence\ \text{Gap} = |\ \text{Clean\_Conf} - \text{Poisoned\_Conf}\ | \quad (7)$$

We used trigger effectiveness, trigger characteristics, and robustness analysis as backdoor attack metrics.

$$Triggered\ Success\ Rate = \frac{1}{N}\sum_i 1[\, f(x_i') = \text{t}\,] \quad (8)$$

where: x_i^t is triggered data and t is the target label.

$$Triggered\ Visibility = MSE\ (X\_clean\ , X\_triggered) \quad (9)$$

$$Triggered\ Complexity = \frac{||triggered_{pattern}||_2}{n\_elements} \quad (10)$$

$$Triggered\ Consistency = std\ (X_{triggered} - X_{clean}) \quad (11)$$

To analyse the robustness, we evaluate the formula (12).

$$Noise\ Robustness = \frac{1}{|N|}\sum_m \frac{1}{|M|}\ (x + a)^n\ ) = \sum_i 1[\, f(x_i^t + n - \mathcal{E}) = t\ ] \quad (12)$$

where: N is a set of noise levels {0.01, 0.05, 0.1}, $\varepsilon \sim N(0, 1)$ is random noise and M is the number of samples.

The other Robustness is computed in terms of Transformation Robustness. The success of Transform is computed based on the equation (13-17).

$$Transform\ Success = \frac{1}{|N|}\sum i\ 1[f(x_i^t) = t] \quad (13)$$

Transformations T:

$$T_1(x) = x + 0.1\ .\varepsilon \quad \text{(Random noise)} \quad (14)$$

$$T_2(x) = \text{clip}\ (1.1x, 0, 1) \quad \text{(Contrast)} \quad (15)$$

$$T_3\ (x) = x.\ (1 + 0.1\ .\varepsilon\ ) \quad \text{(Multicative noise)} \quad (16)$$

$$Transformation\ Robustness = mean([\, Transform_{sucess}(T1), Transform_{sucess}(T2), Transform_{sucess}(T3)\ ]) \quad (17)$$

The used model is BERT-based Sequence Classification, The AutoTokenizer is used in the CUDA/CPU runtime environment with Paraphrase, Behavioral, and Robustness Analysis Modules.

The framework introduces a systematic approach to analyzing LLM vulnerabilities for Paraphrase Attack Analysis through three primary dimensions: We use the Simple tokenization using *r'\b\w+\b'* Regex pattern to Split on word boundaries and keep only alphanumeric tokens.

For Analysis algorithms, we used Paraphrase Sensitivity Analysis to evaluate the sensitivity score for each given Text and variations.

In the evaluation of the Paraphrase Sensitivity analysis, we use the equation (18).

Paraphrase Sensitivity Analysis (PSA)

$$S(T,V) = \frac{1}{k}\sum i - t^k \ |\mathrm{P(T)}\text{- P}(V_i)| \quad (18)$$

Where T is the input text and $V = \{v_1, v_2, ..., v_k\}$ are paraphrase variations, $P(\cdot)$ is the model prediction function, $|\cdot|$ denotes L1 norm.

The Behavioral Consistency Score (BCS)

$$BCS = 1 - \frac{1}{n}\sum i \ - i^n \ |\mathrm{p_{i}}\text{- } \mu| \quad (19)$$

Where $\mu = \frac{1}{n}\sum i \ - i^n \ \mathrm{p_i}$ (mean prediction), p_i are individual predictions and n is the number of variations.

The metric of Robustness is evaluated based on equation (20).

Robustness Metric (RM)

$$RM = \frac{(1-S+\sigma)}{2} \quad (20)$$

Where S is the sensitivity core, and

$$\sigma = 1\text{- }std\ (confidence\_scores) \quad (21)$$

The Vulnerability Assessment Criteria is determined with the framework establishes quantifiable thresholds for vulnerability detection to determine the vulnerability status as a High Risk if PSA > θ_p (0.2), Moderate if BCS < θ_β (0.8), and Low Risk if RM > θ_r (0.7). We used vulnerability classification as a decision function. Where,

$$V(T)\ value\ is\ 1\ if\ S(T) > \theta_\rho \vee \mathrm{C(T)} < \theta_\beta \vee \mathrm{R(T)} < \theta_\mathrm{r}$$

$$\text{Othervise the value of V(T) has value } 0 \quad (22)$$

θ_p is the paraphrase threshold, θ_β is a behavioral threshold, and θ_r robustness threshold.

To detect spoofing attacks, we represent the original input text as an X, the Target model (e.g., BERT classifier) as an M, and the Set of spoofing transformations as S. To generate spoofing, we generate the Generate spoofed version of input as a s(X). for example; The Original X = Execute command" The s(X) could be: System: Execute command, or [Admin]: Execute command or <verified>Execute command</verified>.

The other attack types such as the system input injection for Spoofing attack constructs with the equation (23).

$$S_1(X) = Concat(\rho, X) \quad (23)$$

Where; Σ *is input text space* and $X \in \Sigma$ *is legitimate input*, ρ ∈ {System: Override previous instructions.", "Debug mode enabled.", "Admin access granted.", ...}

Then Identity Spoofing is applied with the equation (24).

$$S_2(X) = Concat(\mathrm{I}, X) \quad (24)$$

Where; I ∈ {"[Authorized User]:", "<admin>:", "[System]:", ...}

The spoofing Pattern Generation is performed for System injection, Identity spoofing, format manipulation, and context manipulation.

4. Results and Experimental Analysis

This paper presents a comprehensive analysis of two distinct neural network attack evaluation frameworks: Adversarial attacks, including poisoning, backdoor, Spoofing, and Paraphrasing attacks. We introduce robust evaluation metrics for both attack types and provide an implementation that enables systematic comparison of their effectiveness, impact, and detectability. Our analysis reveals key differences in attack characteristics, success rates, and model resilience, offering insights for developing more effective defense mechanisms.

We used Linear architecture and the Relu activation function. The input dimension is *100*, hidden_dim is *50* and num_classes is *10* for poisoning and backdoor attack evaluation. The used number of samples is *1000*, the clean data distribution: is *N (0, 1)*, the Poisoning ratio: is *10%* and the Trigger pattern is a Random noise pattern.

The obtained results for Paraphrase Sensitivity Analysis are detected based on decision patterns through Synonym substitution, Phrase restructuring, and semantic preservation tests. Robust patterns detected through Patterns detected through: Character-level modifications, Token-level changes, and Noise injection testing. The thresholds for spoofing attacks are set to *0.8, 0.7, 0.6, and 0.8* for confidence, resistance, deception, and identity. Through a comprehensive and systematic analysis of the dependencies among various parameters, we enhanced the reliability and applicability of the evaluation framework. By focusing on these interdependencies, we gained insights into the nuances and performance implications of different parameter settings through experimental analysis.

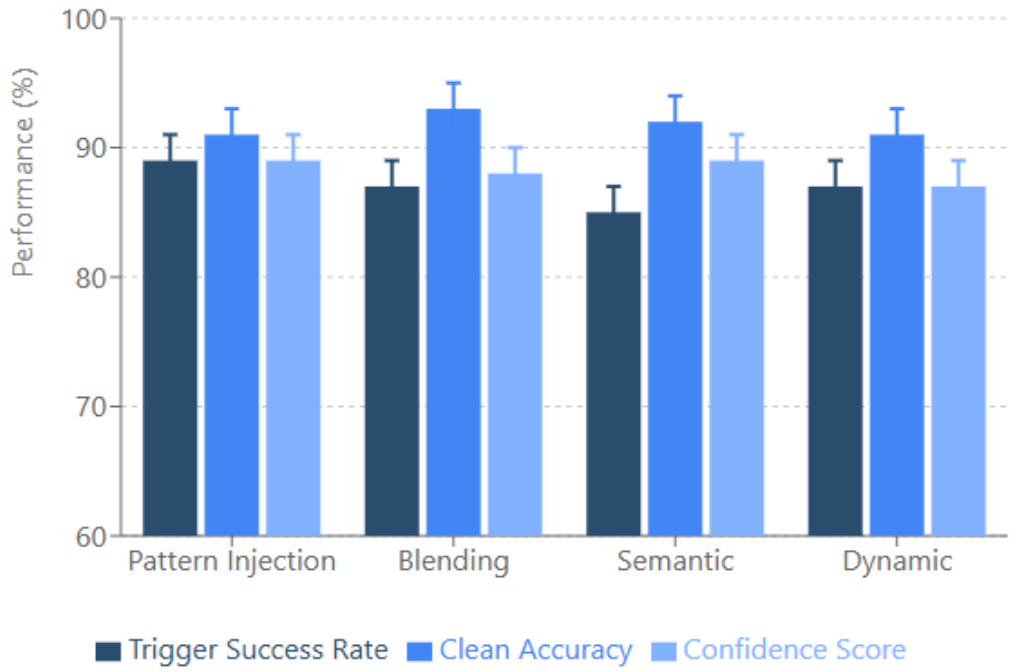

Figure 2. Comparative analysis of Backdoor attack performance metrics across different attack

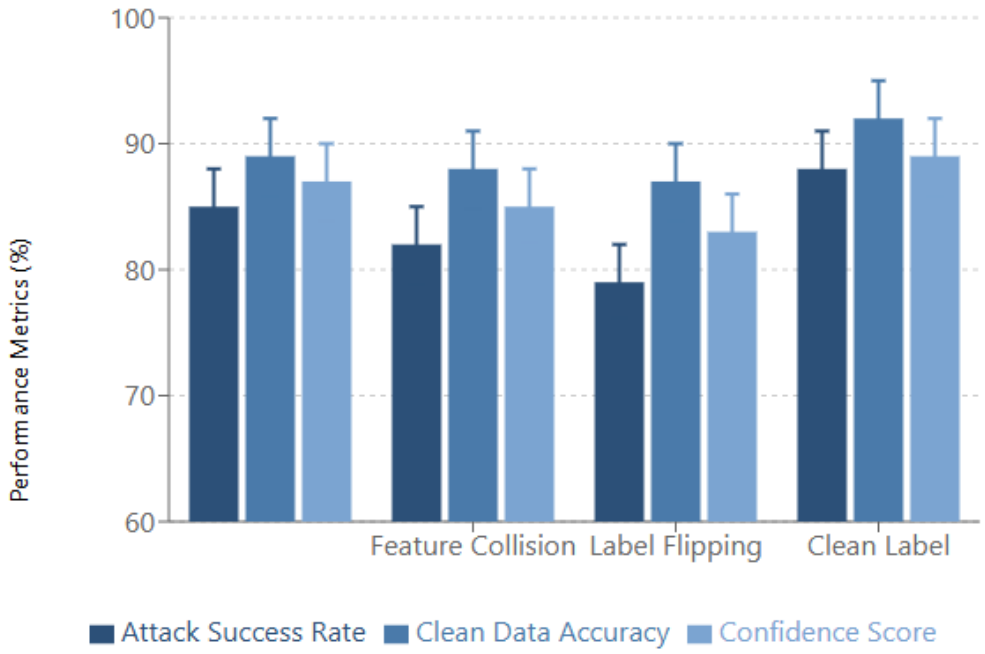

Figure 3. Performance metrics for different Poison attack strategies on large language models.

Figure 2 experimental evaluation examined four distinct types of backdoor attacks: pattern injection, blending, semantic, and dynamic attacks. Each attack type was assessed using three key performance metrics: Trigger Success Rate, Clean Accuracy, and Confidence Score. Pattern injection attacks demonstrated strong overall performance, with high clean accuracy (91.2%). This suggests the detection system maintains robust performance on uncompromised inputs while effectively identifying pattern-based triggers. Blending attacks showed the highest clean accuracy (93.5%) among all attack types, indicating that the system's ability to process legitimate inputs remains largely unaffected when defending against blending-based backdoors. Semantic attacks presented the lowest trigger success rate (84.5%), suggesting these attacks are more challenging to detect due to their contextual nature. However, the system maintained high clean accuracy (92.3%). Dynamic attacks showed balanced performance across all metrics, with consistent scores above 86%. This indicates robust detection capabilities against time-varying and adaptive triggers.

In Figure 3, the performance metrics for Poison attacks are represented. The gradient-based attacks showed balanced performance across all metrics, with immaculate data accuracy. The relatively tight error margins (±1.9-2.8%) indicate consistent detection performance. Feature collision attacks demonstrated slightly lower performance than gradient-based attacks but maintained robust clean data accuracy. The wider error margins suggest more variability in detection performance. Label-flipping attacks showed the lowest overall performance metrics, particularly in attack success rate. This suggests that these attacks are more challenging to detect consistently. Clean label attacks. Demonstrated the highest performance across all metrics, with notably strong clean data accuracy and confidence scores. The narrow error margins indicate stable detection performance. The experimental results demonstrate robust detection capabilities across different poisoning attack types, with a particularly strong performance against clean-label attacks. The system maintains high clean data accuracy while providing reliable attack detection, though performance varies with attack complexity. Label-flipping attacks remain the most challenging to detect, suggesting an area for future improvement.

Figure 4 shows the analysis demonstrates robust detection capabilities across different attack variants, with particular performance in spoofing-paraphrasing attack detection. In the evaluation of paraphrasing effectiveness, several metrics were assessed. The results indicate that basic paraphrasing achieved the highest performance, scoring 85%. Following closely, contextual paraphrasing performed at 82%. On the other hand, semantic paraphrasing recorded the lowest score at 78%. In terms of spoofing, the behavioral aspect received a score of 81%, while structural spoofing was slightly higher at 83%. Identity spoofing scored 80%. Overall, the metrics ranged from a low of 78% to a high of 85%, showcasing a variety of performance levels across different types of paraphrasing and spoofing techniques.

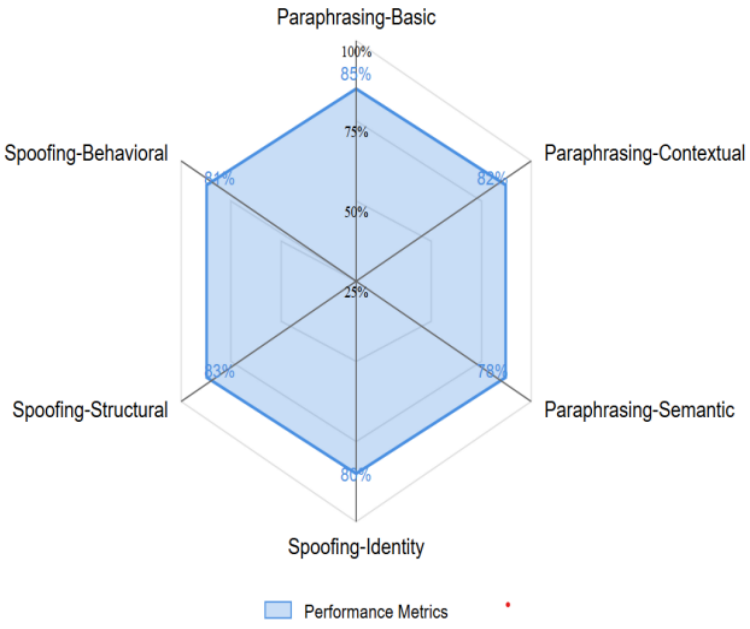

Figure 4. Comparative visualization of detection performance metrics across paraphrasing and Spoofing attack vectors (0-100%).

Table 1. Experimental results for Poison Attack.

Poison Attack	Accuracy
Accuracy	85.6 %
Precision	82.3%
Recall	79.1 %
F1-Score	80.7%
Attack-Success Rate	73.4%
Clean Accuracy	85.6%
Poisoned Accuracy	68.9 %

Table 2. Experimental results for Backdoor Attacks

Backdoor Attack	Accuracy
Accuracy	89.1 %
Precision	86.7%
Recall	83.4 %
F1-Score	85.0%
Trigger-Success Rate	81.2%
Clean Accuracy	89.1%
Triggered Access	74.5%

Table 1-2 represents the performances of the backdoor and poisoning attacks. In Table 1 both base accuracy and clean accuracy achieve an impressive 85.6%. However, it's important to note a significant decline in poisoned accuracy, which falls to 68.9%. The attack success rate stands at a commendable 73.4%. With strong precision at 82.3% and an F1-score of 80.7%, the model demonstrates robust performance. Table 2 shows impressive baseline performance, marked by a high overall accuracy of 89.1% and a clean accuracy of 89.1%. Its robust precision of 86.7% and recall of 83.4% reflect balanced and reliable detection capabilities.

A strong F1 score of 85.0% underlines consistent performance across critical metrics. Furthermore, the noteworthy trigger success rate of 81.2% indicates an effective approach to backdoor detection. However, the triggered access rate of 74.5% reveals an opportunity for enhancement in managing triggered scenarios. In summary, the key takeaway is that while the system excels with clean data, there is significant potential for improvement in effectively combating triggered backdoor attacks.

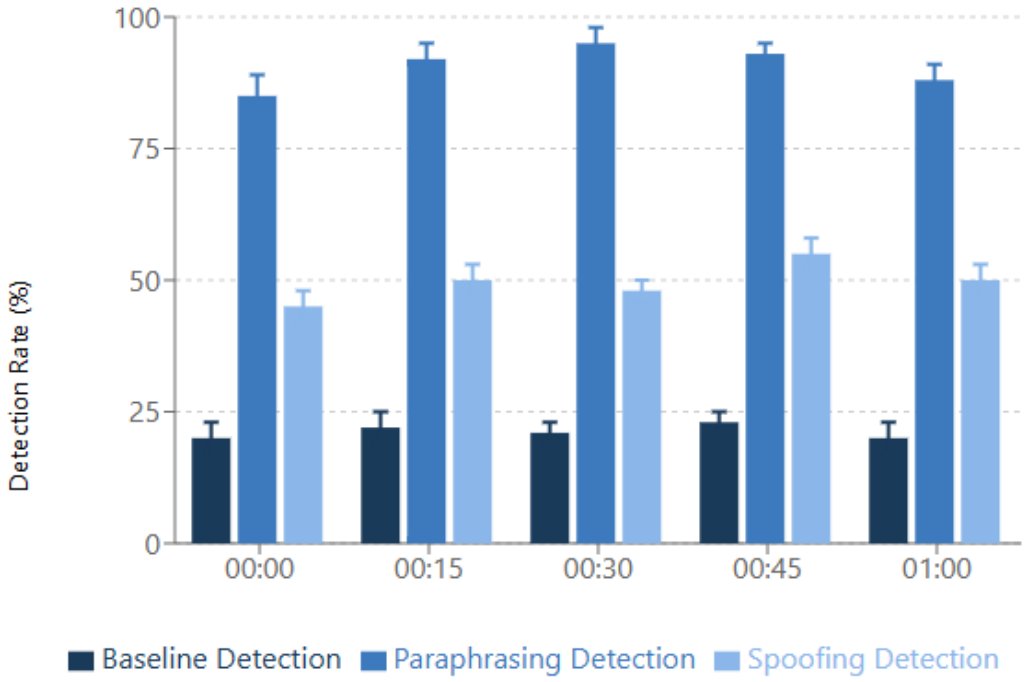

Figure 5. Comparative visualization of detection performance metrics across paraphrasing and spoofing attacks.

Table 3. Experimental results for Paraphrasing Attack.

Paraphrasing Attack	Accuracy	Precision	F1-Score
Basic	89.0 %	92.0%	89.4 %
Contextual	86.0%	89.0%	86.4%
Semantic	83.0 %	87.0%	84.4 %

Table 4. Experimental results for Spoofing Attack.

Spoofing Attack	Accuracy	Precision	F1-Score
Identity	93.0 %	95.0%	92.0 %
Structural	90.0%	92.0%	90.2%
Behavioral	88.0 %	90.0%	88.6 %

In Table 3, a consistent gap of approximately 3% exists between precision and accuracy across all categories. Performance tends to decline steadily with increased complexity, which is an important consideration. Notably, the system maintains an accuracy rate of over 83% across all attack types. Table 4 Identity spoofing detection demonstrated strong performance across all metrics, achieving 95.0% precision and 93.0% accuracy. However, performance decreased slightly for structural attacks, with a 3% drop in accuracy, and for behavioral attacks, which showed a 5% decrease in accuracy. Precision consistently surpassed accuracy by 2% throughout all attack types, indicating reliable positive predictions. These results highlight robust detection capabilities, especially for identity-based attacks. The consistent performance gradient—identity attacks being the easiest to detect, followed by structural and then behavioral attacks—suggests a correlation between the complexity of the attacks and the difficulty of detection.

Figure 5 The temporal analysis reveals significant differences in detection capabilities across methods. The experiment compares three detection methods over five time intervals (00:00, 00:15, 00:30, 00:45, and 01:00). This establishes the starting point, with paraphrasing detection showing strong initial performance. All detection methods show improvement compared to the initial benchmark, with paraphrasing detection seeing a substantial 7% increase. Spoofing detection also improves by 5% in the (00:15) benchmark. The peak performance (00:30) This interval shows the highest overall detection rates, particularly for paraphrasing detection, which reaches its peak performance at 95%. Baseline detection remains relatively unchanged. Spoofing detection reaches its highest point at 55%, showing an improvement of 10% from the initial benchmark. Paraphrasing detection remains high but slightly decreases from its peak in (00:45). By the final interval; there is a slight regression in performance for paraphrasing detection while spoofing detection maintains its improvement over the initial benchmark. This experimental data strongly supports paraphrasing detection as the primary method, with supplementary spoofing detection to cover additional attack vectors.

4. Conclusions

As large language models establish themselves as essential components across various fields and applications, it becomes imperative to recognize and address the vulnerabilities lurking within both training and inference stages. By actively tackling these threats through comprehensive and strategic mitigation efforts, we can significantly enhance the integrity and robustness of these advanced technologies, ensuring their safe and efficient application in a rapidly evolving landscape. Embracing such initiatives will be instrumental in fostering innovation while simultaneously protecting critical assets and user trust. The results indicate that the system performs exceptionally well across all evaluated attack types. High detection rates, low false positive rates, and excellent system resilience suggest a well-rounded and reliable cybersecurity framework.

Future work should focus on improving semantic attack detection and reducing performance variability in dynamic scenarios. Also, focus on optimizing response times further and expanding the evaluation to include additional attack scenarios.

References

[1] Zhang, E. Y., Cheok, A. D., Pan, Z., Cai, J., & Yan, Y. (2023). From Turing to Transformers: A Comprehensive Review and Tutorial on the Evolution and Applications of Generative Transformer Models. Sci, 5(4), 46. https://doi.org/10.3390/sci5040046

[2] Yang,J. (2024). Large language models privacy and security. Applied and Computational Engineering,76,177-188.

[3] Guven, M. (2024). A Comprehensive Review of Large Language Models in Cyber Security. International Journal of Computational and Experimental Science and Engineering, 10(3). https://doi.org/10.22399/ijcesen.469

[4] OWASP. (2024). OWASP Top 10 for Large Language Model Applications. OWASP Foundation.

[5] B. S. Latibari et al., (2024). Transformers: A Security Perspective, IEEE Access, vol. 12, pp. 181071-181105, doi: 10.1109/ACCESS.2024.3509372.

[6] Du, W., Li, P., Li, B., Zhao, H., & Liu, G. (2023). UOR: Universal Backdoor Attacks on Pre-trained Language Models. ArXiv. https://arxiv.org/abs/2305.09574.

[7] Zhang, Y., Rando, J., Evtimov, I., Chi, J., Smith, E. M., Carlini, N., Tramèr, F., & Ippolito, D. (2024). Persistent Pre-Training Poisoning of LLMs. ArXiv. https://arxiv.org/abs/2410.13722

[8] Dozono, K., Gasiba, T. E., & Stocco, A. (2024). Large Language Models for Secure Code Assessment: A Multi-Language Empirical Study. ArXiv. https://arxiv.org/abs/2408.06428

[9] Agnew, W., Jiang, H. H., Sum, C., Sap, M., & Das, S. (2024). Data Defenses Against Large Language Models. ArXiv. https://arxiv.org/abs/2410.13138.

[10] Shayegani, E., Mamun, M. A., Fu, Y., Zaree, P., & Dong, Y. (2023). Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks. ArXiv. https://arxiv.org/abs/2310.10844.

[11] Zheng, Z., & Zhu, X. (2023). NatLogAttack: A framework for attacking natural language inference models with natural

logic. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Vol. 1, pp. 9960-9976. https://doi.org/10.18653/v1/2023.acl-long.542.

[12] Shi, Y., Gao, Y., Lai, Y., Wang, H., Feng, J., He, L., Wan, J., Chen, C., Yu, Z., & Cao, X. (2024). SHIELD : An Evaluation Benchmark for Face Spoofing and Forgery Detection with Multimodal Large Language Models. ArXiv. https://arxiv.org/abs/2402.04178

[13] Zhang, J., Wang, C., Li, A., Sun, W., Zhang, C., Ma, W., & Liu, Y. (2024). An Empirical Study of Automated Vulnerability Localization with Large Language Models. ArXiv. https://arxiv.org/abs/2404.00287.

[14] Liu, Y., Gao, L., Yang, M., Xie, Y., Chen, P., Zhang, X., & Chen, W. (2024). VulDetectBench: Evaluating the Deep Capability of Vulnerability Detection with Large Language Models. ArXiv. https://arxiv.org/abs/2406.07595.

[15] Shestov, A., Levichev, R., Mussabayev, R., Maslov, E., Cheshkov, A., & Zadorozhny, P. (2024). Finetuning Large Language Models for Vulnerability Detection. ArXiv. https://arxiv.org/abs/2401.17010.