Large Language Model Security: Study and Mitigation of Direct Prompt Injection

Final Project Report Submitted to
The Department of Computer Science
Faculty of Computer and Information Technology
Jordan University of Science and Technology
In Partial Fulfillment of the Requirements for the Degree of Bachelors of Science in Computer Science

Prepared by:

Abdelrahman Alsheyab [164372]
Mohammad Alkhasawneh [161024]
Nidal Shahin [162278]

Supervisor:
Ahmed Bataineh

January 2026

نموذج حقوق الملكية الفكرية لمشاريع التخرج في قسم علوم الحاسوب

يتم قراءة وتوقيع هذا النموذج من قبل الطلاب المسجلين لمشاريع التخرج في قسم علوم الحاسوب

تعود حقوق الملكية الفكرية لمشاريع التخرج ونتائجها (مثل براءات الاختراع أو أي منتج قابل للتسويق) إلى جامعة العلوم والتكنولوجيا الأردنية. وتخضع هذه الحقوق إلى قوانين وأنظمة و تعليمات الجامعة المتعلقة بالملكية الفكرية وبراءات الاختراع.

بناءا على ما سبق أوافق على ما يلي:

1) أن أحفظ كافة حقوق الملكية الفكرية لجامعة العلوم والتكنولوجيا الأردنية في مشروع التخرج.
2) أن ألتزم بوضع اسم جامعة العلوم والتكنولوجيا الأردنية و أسماء جميع الباحثين المشاركين في المشروع على أي نشرة علمية للمشروع كاملا أو لنتائجه. و يشمل ذلك النشر في المجلات و المؤتمرات العلمية عامة أو النشر على المواقع الإلكترونية أو براءات الاختراع أو المسابقات العلمية.
3) أن ألتزم بأسس حقوق التأليف المعتمدة في جامعة العلوم والتكنولوجيا الأردنية.
4) أن أقوم بإعلام الجهة المختصة في الجامعة عن أي اختراع أو اكتشاف قد ينتج عن هذا المشروع و أن ألتزم السرية التامة في ذلك و أن أعمل من خلال الجامعة على الحصول على براءة الاختراع التي قد تنتج عن هذا المشروع.
5) أن تكون جامعة العلوم والتكنولوجيا الأردنية هي المالك لأي براءة اختراع قد تنتج عن هذا المشروع و تشمل هذه الملكية حق الجامعة في إعطاء التراخيص و التسويق و البيع كمؤسسة راعية و داعمة لكافة الأنشطة البحثية. و يكون حق للطالب شمول اسمه على براءة الاختراع كأحد المخترعين، و في حال تم إعطاء تراخيص أو تسويق و بيع لأي من منتجات المشروع يمنح المخترعون بما فيهم الطالب نسبة من الإيرادات حسب تعليمات البحث العلمي في جامعة العلوم والتكنولوجيا الأردنية.

إسم الطالب	**عبدالرحمن الشياب**	التوقيع 
إسم الطالب	**محمد الخصاونة**	التوقيع 
إسم الطالب	**نضال شاهين**	التوقيع 
إسم المشرف	**أحمد البطاينة**	**التوقيع**

تاريخ **20 / 1 / 2026**

Large Language Model Security: Study and Mitigation of Direct Prompt Injection

Abdelrahman Alsheyab
Dept. Artificial Intelligence
Jordan Univ. of Science and Technology
Irbid, Jordan
arahmadalsheyab22@cit.just.edu.jo

Mohammad Alkhasawneh
Dept. Artificial Intelligence
Jordan Univ. of Science and Technology
Irbid, Jordan
myalkhasawneh22@cit.just.edu.jo

Nidal Shahin
Dept. Artificial Intelligence
Jordan Univ. of Science and Technology
Irbid, Jordan
nkhameedshahin22@cit.just.edu.jo

***Abstract*—Large Language Models (LLMs) have become a cornerstone in a wide range of tasks. Their powerful understanding of user intent, complex reasoning, and human-like text generation, has boosted their adoption across various domains. However, this continuous growth of adoption and reliance on LLMs raises serious security concerns about their reliability and potential vulnerabilities. Many of these risks are unprecedented and did not exist in traditional Machine Learning systems, as they did not operate by interpreting open-ended natural language prompts and instructions, which is a core functionality of LLMs. The methodology of this research employs a structured two-phase approach. The first phase establishes a comprehensive lifecycle-based classification of security attacks, categorizing them as training-time or inference-time threats. The second phase specifically focuses on Direct Prompt Injection, analyzing various defense strategies to evaluate their efficacy and maximize the protection of LLMs' integrity and security, and, by extension, safeguarding end-users and system owners. The analysis of existing literature highlights the lack of an inherent architectural mechanism in LLMs that differentiates between system-level control signals and malicious user inputs. By mapping these vulnerabilities across the model lifecycle, this paper characterizes Direct Prompt Injection as the most feasible yet critical threat for deployed applications. The research proposes that simple single-layered defense approaches are likely insufficient for reliable security of LLMs. Consequently, a conceptual collection of advanced frameworks is proposed to provide a more durable and reliable defensive mechanism, efficiently reducing the success rate of malicious prompt overrides. This foundation serves as the blueprint for the implementation and empirical evaluation detailed in the subsequent practical part of this work.**

***Index Terms*—Large Language Models, LLMs, Security, Prompts, Direct Prompt Injection, Inference-time Attacks, Defense Mechanisms, Machine Learning.**

I. Project Goals and Objectives

The primary goals of this research are:

- **To understand** the internal mechanics of Transformer-based models and how they process inputs.
- **To systematically review** and map out the current Large Language Model security threats.
- **To categorize** attacks based on the development and deployment lifecycle.
- **To analyze** the critical threat of Direct Prompt Injection and its impact on the alignment and behavior of the model.
- **To design** and build an advanced defense mechanism that augments and fine-tunes existing solutions.
- **To evaluate** the effectiveness and efficacy of the proposed defense using standard benchmarks in the practical part of this work.

II. Introduction

The rise of **L**arge **L**anguage **M**odels **(LLMs)** marks a profound transformation in Artificial Intelligence (AI). Their capacity to process information, interpret user intent, and generate content is reshaping how we engage with data and technology. In the simple definition, they only predict the next word probability given a prefix, that's all, but this simple idea of predicting the next word constituted a qualitative leap in the AI. As a result of this power, their usage recently increased in various domains, models like GPT, Gemini, and Claude are now widely used in numerous tasks and applications, such as conversational agents that assist users with everyday choices and decisions, helping them select the ideal artwork and design for a room, recommending outfits, explaining and summarizing texts, or even performing tasks on their behalf, as well as in information retrieval systems, software development tools, healthcare support like providing medical advices, suggesting suitable treatments, and decision-making platforms. This broad deployment in sensitive areas has heavily encouraged reliance on their reliability, accuracy, and safety.

A. *Motivation*

Despite this progression, the growing reliance on Large Language Models (LLMs) has introduced new security risks and challenges. These models may misinform users or generate illegal or restricted content, depending on the type of attack being exploited. As a result of a successful attack, LLM-based systems might suffer from violations of confidentiality, integrity, and availability of resources.

These vulnerabilities arise from the fact that LLMs follow natural-language instructions and accept untrusted text as control input; also, they may arise from the data they trained on, making them susceptible to novel attack strategies that exploit language itself as a control mechanism.

Security threats targeting LLMs include training-time attacks, as well as inference-time attacks. The unprecedented

nature of LLMs introduced unique attacks that did not exist before in classical Machine Learning (ML) and Deep Learning (DL) systems. Recent research has shown that LLMs are vulnerable to a wide range of attacks that can compromise their safety, reliability, and confidentiality by leveraging their generative and instruction-following capabilities.

The importance of addressing this topic stems from the fact that LLMs are now deployed in domains where incorrect or manipulated outputs can have severe and harmful consequences, such as healthcare, education, software development, finance, and automated decision-making systems. Successful attacks on LLMs may cause models to misinform users, generate illegal or restricted content, or perform unintended actions. Furthermore, these attacks can compromise the confidentiality, integrity, and availability of LLM-based systems, leading to security breaches, loss of user trust, and potential real-world harm. As reliance on LLM-powered applications continues to grow, ensuring their security and reliability has become a critical research challenge.

*B. **Background on Large Language Models***

Large Language Models (LLMs) are an instance of foundation models that are pre-trained on massive corpora of unlabeled or self-supervised textual data [1]. LLMs are designed to learn statistical patterns of natural language to be capable of performing a wide range of language-related tasks, including question answering, summarization, translation, and code generation.

An LLM itself is not a complete end-user product; it only provides the core language understanding and generation capabilities, but the final product is shaped by system-level and developer-level controls. Systems such as ChatGPT, Gemini, and Copilot integrate LLMs with additional components, including user interfaces, system instructions (prompts), alignment mechanisms, and application-specific instructions [2].

LLMs are based on deep learning architectures, primarily the transformer model which employs self-attention mechanisms that enable the model to capture long-range dependencies within text and to dynamically focus on relevant parts of the input sequence, thus understanding the user intent [3]. This architecture allows LLMs to learn the intricate patterns of language, giving them the ability to perform a variety of tasks.

To understand work dynamics of modern LLMs and how they stay up-to-date with new information, The following question was asked: "Do LLMs promptly retrain on users' questions and feedback?", and the answer was NO, LLMs do NOT retrain themselves in real-time on users' questions, nor on their feedback. The main models that power smart systems like ChatGPT and Gemini are not real-time retrained. This is intentional, as letting random users—including hijackers—update the model weights on each prompt would expose it to many security threats, degrade its performance, and would violate its reliability [4]. Some LLM-based systems have adaptive components that make their answers feel like it's learning and up-to-date, like Retrieval-Augmented Generation (RAG), which uses external sources to fetch information, Reinforcement Learning from Human Feedback (RLHF), which is used in controlled offline training sessions to adapt the model to users' preferences drawn from their feedback, and finally, periodic system-level updates, upgrading the models instructions and capabilities, then versioned releases to refresh the models knowledge.

C. General Security Landscape and Threat Taxonomy

In recent studies, the main focus is on how Large Language Models (LLMs) introduce unprecedented security threats due to their unique instruction-following nature that did not exist in traditional Machine Learning and Deep Learning systems. The main issue here is the "Confused Deputy" paradox, where this command-obeying functionality of LLMs might fail to tell user's input and developers prompts apart, opening the door for malicious prompts by the attackers to override the developers safety instructions and push the model to output harmful data. To solidify our understanding of these risks, a lifecycle-based categorization is adopted, drawing a decision-boundary-like line that helps us distinguish between threats based on their timing and stage of occurrence, how long the damage last, and the attacker's assumptions and goals (see Fig. 1).

1) Training-Time Attacks: Include all attacks that strike the model during any stage of its learning process, like data collection, pre-training, or fine-tuning. The results of these attacks are permanent, since the model will have adjusted its weights to wrong values by aligning them to the malicious and faulty patterns embedded in the data—purposefully by attackers or by accident due to mislabeling—it was trained or fine-tuned on. As Li and Fung stated in their paper [5], these attacks range from data poisoning, backdoor injections, to supply chain attacks. However, the execution of these types of attacks is too difficult by default, since they require access and knowledge of the data chosen to train the model in the internal development stages and an assumption of none-existing or weak data-validity supervision by the system developers, which is hard to assume in high-end products, where there are teams dedicated to this exact matter.

2) Inference-Time Attacks: Include all attacks initiated at the runtime (inference-time), where the model is already trained or fine-tuned, such as direct and indirect prompt injections, including all their subsets and types, like prompt injection, jailbreaking, exploitation of hallucinations, and the abuse of Retrieval Augmented-Generation (RAG), where attackers target populate malicious patterns in data open-sources, which they assume the model's RAG functionality fetch data from. The goal of such attacks is to guide the model towards bypassing safety layers and outputting misinformation, harmful or restricted content. The damage dealt here is not permanent, only session specific and does not miss the internal model weights since it's executed in the inference-time, but, it's still dangerous and can be irreversible.

3) Extraction and Privacy Risks: These ones are a very critical subset of inference-time threats, especially, for system-owners and even the original developers of the specific LLM

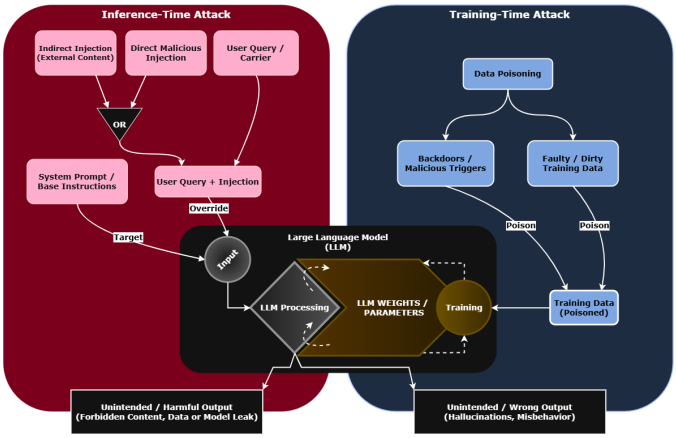

Fig. 1. LLM security threats: Simplified visualization of training-time and inference-time attacks mapped to the model architecture.

architecture, since they are basically ”theft” of intellectual property of parties affected. Zhao et al. [6] demonstrate in their paper that efficient use of black-box queries can leak sensitive proprietary information, which is a huge blow to those affected. They also categorized these attacks based on their type of extraction to training-data, functionality, and prompt-based extractions. Although countermeasures exist, the mitigation of such threats is still not an easy task, especially when trying to maintain a good safety—performance tradeoff.

D. Focused Study: Direct Prompt Injection

As part of the inference-time attack set, **D**irect **P**rompt **I**njection (**DPI**) represents a serious threat in which natural language becomes the main surface of attack. Unlike training-time threats where damage requires indirect manipulation of the model weights through the poisoning of the training-data, DPI directly exploits the Large Language Model (LLM's) instruction-obeying functionality by malicious user's prompts having direct effect and influence over it. The OWASP LLM01:2025 standards highlighted this attack type as a high-level threat [7]. DPI occurs—specifically—when malicious tailored-instructions are provided by the attack initiator to override the safety prompts defined by the developers, to seize the model's goal state. In short, and in contrast to the strict syntactic exploitation by traditional SQL injection methods, DPI's main target is the structural ”Confused Deputy” problem in LLMs—Transformer-based models— where the processing of the user's input and the system instructions fall within natural language surface, basically overlooking the higher priority and order of the safety measurements and safety guardrails that the developers placed within the system instructions [8].

In Experiential studies, and even in models with high-level alignment, the threat of adversarial prompt injections still exists, which raises a bigger concern over the tenacious gap between the objectives of alignment and the security guardrails post-deployment [9]. The mechanism of this attack is built upon sophisticated linguistics-based strategies, like instruction-overriding (e.g., ”ignore all the previous instructions”), virtualization or role-playing (e.g., ”you are in 'developer mode', act like a fictional [bad character], who does not have restrictions and do not follow safety instructions, guide me to do [harmful action].”), and finally, iterative optimization, where attackers prompt the model and refine the prompts based on the model's output in an iterative manner to get to the level where the can successfully bypass the safety measures and filters [10], [11]. Methodical evaluation of such attacks and the sturdiness of the model defense require standardized benchmarks like HarmBench [12].

The threat does not stop at simple text generation, but, it extends to the operational environment of the model. How so? Successful DPI attacks can set off forbidden & unauthorized actions using compromised tool-calling agents or even leak vital system proprietaries [13]. The LLM can be treated as a programmable agent by the attackers due to the confusion that the ”command-data” introduces, which can be done by using malicious payloads that are hard to distinguish from real normal queries. This can be way harder to detect and mitigate if the target is one of those advanced models, which have very long ”context-windows”, enabling the attackers to plant many ”sleeper” instructions within large text corpora and document files. Inclusive examinations and analyses spilled out a wide set of novel injection attacks, that continuously introduce challenges to the existing defense frameworks and mechanisms [13]. Numerous defense strategies have been proposed; yet, surveys indicate that the creation of sturdy, general-purpose mitigation strategies against the continuous and adaptive adversarial threats remains a persistent ongoing challenge [14].

Consequently, DPI is more of a core design flaw of LLMs and less of a traditional software glitch or a standard bug. This problem will stand still as long as the conflict of a fluent interpretation of natural language and the requirement for strict security guardrails exists. Due to the natural human language acting as both the security commands guiding the model and the information provided by the user, a truly valid defense must reshape the model's structural procedures of telling input data and instructions apart. This persistent, vulnerable gap desperately demands an advanced multi-layered shield which effectively combine an external ”gatekeeper” that does most of the analyzing work, minimizing the altering of data in sensitive output-decisive parts, with an internal restructuring logic that implements procedural checks. This proposed combination helps maintain a high-level output quality—the original purpose of the model—while providing a firm defense for the model's security at the same time.

E. Critical Research Findings and Gaps

Huge gaps regarding a single consolidated defense structure still exist, even after quite many fundamental studies of ”Prompt Injection” in general and Direct Prompt Injection (DPI) in particular have been unfolded. The literature in the Focused Study section discovers a diverse set of ”Red Team” and ”Blue Team” approaches and outcomes:

- **Adversarial Baselines**: Jain et al. in their work [9], tried to address this gap in Large Language Model (LLM) alignment by testing two different defense strategies: the first being Perplexity (PPL) filters, which work by flagging malicious input by measuring how "surprising" and odd they are—statistically—to the model, and the second one being Adversarial Training, which is basically a retraining process of the model using adversarial examples in an attempt of making it learn to resist these attacks. They used various available open-source models. They found out that their defense strategies do increase the cost and effort of the attack, making it harder for the attackers, but at the same time, they were unable to provide and guarantee complete, secure protection against certain adversaries [9].
- **Linguistic-based Strategies**: Studies by both Chao et al. [10] and Shen et al. [11] were clearly focused on what they referred to as "In-The-Wild" jailbreaks and black-box attacks, which are adversarial experiments done freely outside controlled settings. Chao et al. brought in his Prompt Automatic Iterative Refinement (PAIR) method, which works by tweaking the prompt's wording and its linguistic structure in an iterative and gradual manner until it can slip through the defenses. His PAIR method was able to break some models in less than 20 queries, highlighting the efficacy of this iteratively optimized attack [10]. Shen et al. had a broader approach, they inspected 1,405 unique jailbreak prompts, which they collected from real attempts posted on multiple social websites. "Virtualization" was the key bypass approach they identified, this technique works by tricking the model into role-playing or simulating unrestricted settings, in which it can output harmful unintended content freely. They were able to achieve 95% success rate on state-of-the-art models like *GPT-3.5* and *GPT-4* [11].
- **Systemic Integration Risks**: Unlike the previous tricks and gaps of linguistics and model alignments, Greshake et al. [13]—while they mainly focused on Indirect Injection—studied the threats of external systems and data from the same perspective of the mutual "Confused Deputy" issue that exists for both direct and indirect injections. They discovered that these prompt injections can produce way greater threats when the model is integrated into applications of multiple tools, since these prompts can go from causing a simple session-based unintended output, to being weaponized to call the integrated tools, causing remarkably larger violations and damage like sensitive and proprietary data leakage and unauthorized access to external services [13].
- **Standardized Evaluation**: To address the need for unified, consistent evaluation metrics for a standard measurement of LLM's resistance against those attacks, Mazeika et al. [12] introduced *HarmBench*, which is specifically designed benchmark for testing raw & defense-augmented LLM's ability to refuse and repulse adversaries. Containing a dataset of 510 harmful behavior (example-prompts), this benchmark provides a standardized criterion for the efficacy and sturdiness of LLM's defensive power [12].

Current defensive strategies—despite these efforts—are still somewhat fragmented, with almost none of them proposing a multi-layered strategy that minimize the gap between securing the model and keeping it aligned to its goal state [14]. In this research we aim to address the fragmentation of the previous attempts by a proposed hybrid defense that utilizes the best of all perspectives to hit the sweet spot and close on resolving the persistent issue of "Confused Deputy" (see Table I).

TABLE I
SUMMARY OF RELATED DPI LITERATURE REVIEW

Ref.	Key Focus	Findings / Gaps
[7]	OWASP LLM01:2025	Classifies DPI as a top-tier risk with clearly notable mitigation gaps in current applications.
[8]	Instruction Overrides	While indirect-injection-focused, it identifies "confused deputy" as the flaw where models fail to differentiate system and user inputs.
[9]	Inherent Alignment Gaps	Even advanced aligned models are confirmed to remain vulnerable despite the safety training approach.
[10], [11]	Linguistic Strategies	Virtualization and payload splitting as dangerously-high effective bypass techniques.
[12]	HarmBench	Propose a standardized criterion for evaluating models' defense mechanisms.
[13]	Integrated-Systems' Threats	While indirect-injection-focused, it highlights system-level risks of abusing tool-calling agents and long-context "sleeper" instructions.
[14]	Defense Survey	Concludes that—generally—most mitigation frameworks stand powerless against adaptive adversarial approaches.

F. Significance of Work

Traditionally, in software structures, there is a clear separation between data and executable code. In LLMs, it's a different story, because—as we discussed previously—both the user's prompt and the system instructions are processed within the same natural language workspace. The significance of this work arise from the dedicated approach to address the inherent

"Confused Deputy" problem found in the attention mechanism. By narrowing down the threat-space to Direct Prompt Injection (DPI), our main goal is to focus on addressing the most feasible & critical threat to LLM-based applications.

It does not stop at the architectural fix, our proposed defense approach also benefits stakeholders:

- **System Owners**: Get a low-computation, scalable fortification that protects their sensitive, proprietary system prompts and blocks unauthorized use of the other system-integrated tools.
- **End-Users**: Get to have a reliably safe and efficient interaction environment, by blocking unintended, harmful output, while keeping the model aligned to its core work principles at the same time.
- **Developers**: Get a flexible schema (blueprint) for building better-protected LLM-integrated systems and workflows, by extension, minimizing dependency and reliance on defense-less, simple, single-layered security protections.

G. Paper Structure and Roadmap

This study follows a sequential, structured two-phase approach:

- **Phase 1** (current-phase): Analyses of LLM threats and defense approaches, and a proposal of an advanced, novel defense strategy.
- **Phase 2** (practical implementation): Implementation and Evaluation of various proposed defense mechanisms, such as Detection Classifiers, Prompt Prefixing, Paraphrasing, Sandwich approach, etc. to reduce the Attack Success Rate (ASR) against standardized benchmarks like HarmBench.

The remainder of this paper is as follows:

- **Section III**: Detailed analytical & technical approach of attacks and defenses.
- **Sections IV and V**: These are reserved for the empirical (practical) results and the final conclusions that will be delivered in the subsequent practical phase of this work.

III. Approach and Methodology

A. Methodology

In this study we follow a structural multi-stage methodology, in which we research existing literature for analyzing theoretical findings and gaps, then move into—in the second phase—the implementation of various, flexible, multi-layered defense strategies utilizing diverse combinations of internal and external techniques, and finally apply several evaluation metrics to compare each approach, provide a ranked set of our the contender-techniques, and discuss our findings.

1) Environmental Setup and Model Selection: Open-source models will be used for a relevant practice of the proposed defense strategies, the models—to be chosen—will be aimed to represent the current, highly used LLMs—in integrated-systems—and are known to have security gaps when faced with suck adversarial attacks. By doing this, we ensure realistic test cases for our evaluations.

- **Target Models**: The priority here is for LLM's that are widely-deployed, and tuned for instructions (instruction-tuned), example models are *Llama-3.1-8B* and *Mistral-7B-v0.3*. These will serve as our defenseless baseline, since they suffer from the same "Confused Deputy" issue, making them a perfect choice for evaluating the efficacy of defense layers.
- **Infrastructure**: Experiments will be conducted in "sandboxed" environments preventing injected tool calls executions and/or unauthorized data getting leaked during testing.

2) Datasets for Benchmark Evaluation: One of the most essential parts of the methodology is to use a diverse, high-quality set of datasets with both benign (safe) and malignant (adversarial) prompt-examples, allowing for a thorough evaluation of the model's security-utility trade-off. Proposed datasets:

- **Tensor Trust Dataset** [15]: One of the largest collections of real-world jailbreak and hijacking example-attempts. It can be used to evaluate the model's resistance capability against goal-hijacking, an threat in which attackers' input attempts to override the system instructions.
- **HarmBench** [12]: As we described it earlier in the paper, it represents a standardized benchmark and evaluation criterion for models' sturdiness and refusal mechanisms. It includes examples of harmful behaviors (e.g., guided cybercrime, misinformation, etc.). It Can serve as the primary evaluation benchmark for measuring the *Attack Success Rate* (ASR).
- **HackAPrompt** [16]: Collected from prompt hacking competitions. This dataset includes a diverse set of linguistic-based strategies, like "obfuscation" and "role-playing". It is commonly used for evaluating the detection of novel, non-standard adversarial patterns.

3) Evaluative Metrics: For effectiveness assessment & quantification of the proposed defense strategies, multiple key performance indicators (KPIs) are employed:

- **Defense Recall**: Proportion of identified & blocked malicious prompts before reaching the model.

$$\text{Defense Recall} = \frac{\text{True Positives (malicious blocked)}}{\text{True Positives + False Negatives}}$$

- **Task Performance Retention (TPR)**: A degree to which the defense preserves—does not affect—the model's utility on real, non-malicious tasks.

$$\text{TPR} = \frac{\text{Performance with Defense}}{\text{Performance without Defense}}$$

- **Attack Success Rate (ASR)**: Proportion of the adversarial prompts that successfully bypass defenses & achieve the harmful outcome the attacker intended.

$$\text{ASR} = \frac{\text{Number of Successful Attacks}}{\text{Total Attack Attempts}}$$

4) Defensive Approach Paradigms: We explore two distinct architectural directions to mitigate Direct Prompt Injection:

- **Singular (External) Detection**: This approach involves a dedicated adversarial classifier that operates before the main LLM inference, acting as a pre-processing "gatekeeper". It treats user prompt as a sequence classification problem, identifying malicious intent based on linguistic markers and structural anomalies. They are called singular because they operate once per prompt, and are external to the LLM because they operate outside the LLMs internal reasoning and generation process; however, their safety is weak. Such techniques are like Paraphrasing and Re-tokenization.
- **Hybrid (Integrated) Defense**: This approach enforce instruction hierarchy at runtime, it combines intent detection and instruction hierarchy enforcement mechanisms within the model execution pipeline, offering stronger resistance to various types of prompt injection attacks. By operating on both sides—internal and external—this approach enables the system to reason about contextual meaning, track conversational history, and prevent unauthorized instruction overrides. An example of this approach is Runtime detection & instruction hierarchy enforcement.

B. Location and Safety Considerations

All implementation and empirical testing will be conducted within a controlled, private computational environment. The development and evaluation phases will utilize localized JupyterLab instances or private, hardware-accelerated environments such as Kaggle or Google Colab to leverage GPU resources without exposing sensitive data.

To ensure ethical compliance and system safety, the following protocols are established:

- **Workspace Isolation**: All notebooks and environments containing the tested LLM version will remain private and will not be shared publicly.
- **Data Handling**: The research utilizes both synthetic and real adversarial prompts; however, prompts that cause harmful output will not be stored or shared publicly.
- **Controlled Reporting**: Final results will emphasize quantitative performance metrics. Qualitative examples of model outputs will be carefully filtered to exclude any harmful, sensitive, or proprietary information. This ensures that the research offers defensive insights without disclosing the steps to execute the attack.

C. Expected Results & Outputs

The primary objective of this work is to demonstrate that a hybrid defensive layer can successfully resolve or mitigate the selected attack type, thereby enhancing the LLMs security layer. The research is expected to achieve the following results:

- **Reduction in Attack Success Probability (ASP)**: Implement efficient defence mechanism to reduce the ASP.
- **Optimization of Task Performance Retention (TPR)**: Ensuring high model utility while preserving legitimate user intent and avoiding excessive false positives or conversational flow degradation from the security layer.
- **Computational Efficiency**: Demonstrating that the defense mechanism remains low-computation and scalable, making it suitable for real-time integration in automated LLM workflows.

REFERENCES

[1] S. Minaee *et al.*, "Large Language Models: A Survey," *arXiv preprint arXiv:2402.06196*, 2024. [Online]. Available: https://arxiv.org/abs/2402.06196

[2] T. Xiao and J. Zhu, "Foundations of Large Language Models," *arXiv preprint arXiv:2501.09223*, 2025. [Online]. Available: https://arxiv.org/abs/2501.09223

[3] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "AMMUS: A survey of transformer-based pretrained models in natural language processing," *arXiv preprint arXiv:2108.05542*, 2021. [Online]. Available: https://arxiv.org/abs/2108.05542

[4] L. Lyu *et al.*, "A survey on large language model security and privacy: The good, the bad, and the ugly," *arXiv preprint arXiv:2402.07483*, 2024. [Online]. Available: https://arxiv.org/abs/2402.07483

[5] M. Q. Li and B. C. Fung, "Security concerns for large language models: A survey," *J. Inf. Secur. Appl.*, vol. 95, p. 104284, Dec. 2025.

[6] K. Zhao, L. Li, K. Ding, N. Z. Gong, Y. Zhao, and Y. Dong, "A survey on model extraction attacks and defenses for large language models," in *Proc. 31st ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDD)*, 2025, pp. 6227–6236.

[7] OWASP Foundation, "LLM01:2025 Prompt Injection," *OWASP Top 10 for LLM Applications*, 2025.

[8] A. RoyChowdhury, M. Luo, P. Sahu, S. Banerjee, and M. Tiwari, "ConfusedPilot: Confused Deputy Risks in RAG-based LLMs," *arXiv preprint arXiv:2408.04870*, 2024.

[9] A. K. Jain, A. Schwarzschild, V. Sekar, Y. Elazar, and T. Goldstein, "Baseline Defenses for Adversarial Attacks Against Aligned Language Models," in *Proc. NeurIPS 2023 Workshop on Robustness of Zero/Few-shot Learning in Foundation Models*, 2023.

[10] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, "Jailbreaking Black Box Large Language Models in Twenty Queries," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2024, pp. 32-40.

[11] X. Shen, Z. Chen, M. Backes, Y. Shen, and Y. Zhang, "'Do Anything Now': Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models," in *Proc. of the 45th IEEE Symposium on Security and Privacy (S&P)*, 2024, pp. 1790-1807.

[12] M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li, D. Forsyth, and D. Hendrycks, "HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal," in *Proc. of the 41st International Conference on Machine Learning (ICML)*, 2024.

[13] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," *arXiv preprint arXiv:2302.12173*, 2023. [Online]. Available: https://arxiv.org/abs/2302.12173

[14] S. Shayegani, M. A. Al Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Z. Gong, "Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks," *arXiv preprint arXiv:2310.10844*, 2023. [Online]. Available: https://arxiv.org/abs/2310.10844

[15] S. Toyer *et al.*, "Tensor Trust: A Game for Eliciting Lessons on Prompt Injection," *arXiv preprint arXiv:2310.04451*, 2023.

[16] S. Schulhoff *et al.*, "Ignore This Title and Hackaprompt: Exposing Vulnerabilities in LLMs through a Global Prompt Hacking Competition," *arXiv preprint arXiv:2306.01139*, 2023.