

Secure AI Prompt Sandbox (LLM Gateway Security)

Semester Project Proposal

Secure Software Design and Development

Muhammad Afeef Bari 2023356

Mahad Aqeel 2023286

Muhammad Daniyal 2023406

1. Introduction and Motivation

The rapid adoption of Large Language Models (LLMs) in web applications, developer tools, and enterprise systems has introduced new security challenges. One of the most critical risks is prompt injection, where adversarial inputs manipulate model behavior, override system instructions, or attempt to extract sensitive information. Many existing LLM-integrated systems directly forward user prompts to external APIs without structured validation, contextual isolation, or policy enforcement. This significantly increases the attack surface and exposes systems to instruction hijacking, data leakage, and privilege escalation. To address this issue, this project proposes the development of a Secure AI Prompt Sandbox, a web-based gateway that acts as an intermediary between users and an LLM API while enforcing structured prompt security policies.

2. Problem Statement

Modern LLM-based applications often lack dedicated security layers for prompt validation and injection detection. User inputs are frequently treated as trusted data, even though they may contain adversarial instructions designed to manipulate system behavior. Additionally, system-level instructions are sometimes exposed indirectly through poor context management. The absence of monitoring, logging, and structured policy enforcement mechanisms makes it difficult to detect misuse or abuse. The proposed system aims to reduce these risks by introducing a controlled, secure middleware layer that validates, filters, and monitors all prompt interactions.

3. Proposed System Architecture

The Secure AI Prompt Sandbox will consist of three primary components: a frontend user interface, a secure backend gateway, and a controlled LLM API interface. The frontend will allow users to submit prompts and view security feedback. The backend will function as the core security layer, implementing prompt validation, injection detection, context isolation, policy enforcement, and logging. The LLM API will only be accessible through this backend, ensuring that no direct communication occurs between users and the model.

The architecture will enforce strict separation between system instructions and user inputs to prevent system prompt leakage. API credentials will be securely stored and inaccessible from the client side. This layered design ensures that security is embedded at the architectural level rather than treated as an afterthought.

4. Core Functional Components

The system will implement a Prompt Validation Engine responsible for analyzing input prompts and identifying suspicious patterns, instruction override attempts, and adversarial structures. Injection detection heuristics will target known attack strategies such as sandwich attacks, attention blink exploitation, role manipulation, indirect prompt injection, and multilingual injection attempts. Given that multilingual inputs can sometimes bypass safety mechanisms, the system will include analysis of cross-lingual prompts to assess potential alignment gaps.

A Context Isolation Layer will ensure that system-level instructions remain protected and cannot be overridden by user input. A Policy Enforcement Module will evaluate prompts against predefined security rules and determine whether they should be allowed, modified, or blocked. An Audit Logging subsystem will record prompt metadata, system decisions, and security events. To ensure integrity, log entries will implement hash chaining mechanisms, making them tamper evident.

5. Security Implementation

The project will incorporate several secure software design principles. Role-Based Access Control (RBAC) will differentiate between administrative and standard users. API keys required for LLM access will be stored in an encrypted vault to prevent exposure. Rate limiting and abuse detection mechanisms will mitigate denial-of-service risks and repeated policy violations. Input validation and sanitization will protect backend components from injection-based attacks unrelated to LLM prompts.

A structured STRIDE-based threat modeling approach will guide system design. Threats including spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege will be systematically identified and mitigated through architectural controls, authentication mechanisms, logging, and policy enforcement strategies.

6. Usable Security Features

In addition to strong security controls, the system will incorporate usable security principles to ensure that users understand and respond appropriately to security feedback. When a prompt is flagged as potentially malicious, the system will display real-time warnings such as “This prompt may attempt to override system instructions.” A visual risk score indicator (low, medium, high) will provide intuitive feedback regarding the severity of the detected risk.

Rather than silently blocking requests, the system will provide clear explanations describing why a prompt was restricted and offer guidance for safe reformulation. This transparency promotes user awareness while maintaining system protection, demonstrating the integration of human-centered design with technical security controls.

7. Expected Outcomes

By the end of the semester, the project will deliver a functional web-based security gateway capable of detecting and mitigating prompt injection attempts. The deliverables will include documented security requirements, system architecture diagrams, threat models, implementation of core modules, security testing results, and a secure code review. A live demonstration will showcase injection detection, policy enforcement, logging integrity, and user-facing security feedback mechanisms.

8. Conclusion

The Secure AI Prompt Sandbox addresses an emerging and highly relevant security challenge in AI-integrated systems. By combining secure software design principles, structured threat modeling, and usable security mechanisms, this project demonstrates how LLM applications can be deployed in a safer and more controlled environment. The system integrates architectural security, policy enforcement, and human-centered feedback into a cohesive design, making it both technically rigorous and academically valuable.