Building a privacy and security layer around LLM models to protect against common AI attacks

Samuel Kadima, Kyle Totorica | kadimas@xavier.edu, totoricak@xavier.edu | Xavier University | CAE Symposium 2026

Background & Overview

Why AI Security Matters?

- LLM(ChatGPT, COPILOT,etc..) are stateless - no built-in memory/safety of their own
- AI can be manipulated into revealing private or sensitive info
- AI can generate unsafe, harmful, or misleading outputs
- LLMs need safeguard/security layer around them to prevent common AI attacks - real safety comes from the implemented security layer

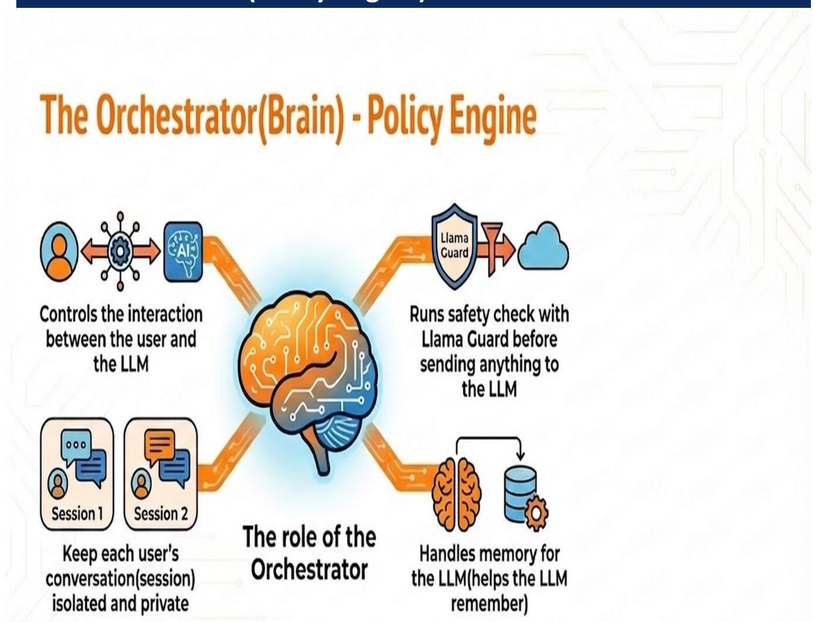
Threat Vectors

- Prompt Injection
- Insecure Output Handling
- Sensitive Information Disclosure

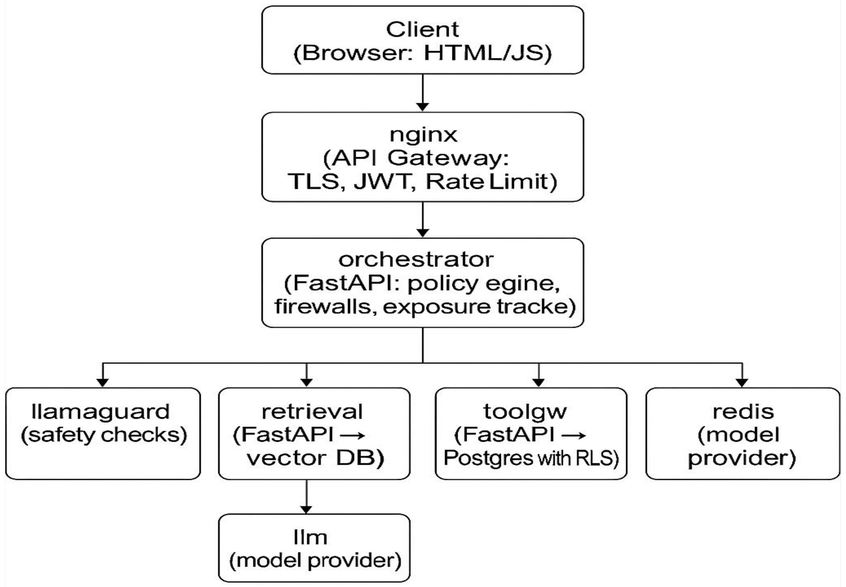
Our Approach

- Treat the LLM as an untrusted source
- An orchestrator(brain/policy enforcer) to enforce privacy and security rules
- Supporting moderation tools(Llama Guard, etc..) to help orchestrator make decision

The Orchestrator(Policy Engine)



System Architecture



Threat vectors

Prompt Injection

- Manipulate the LLM to generate malicious outputs
- Tricking system instructions so the model perform actions outside its intended behavior
- Unsafe LLM responses may be generated if guardrails are bypassed

Insecure Output Handling

- Unrestricted chat model → No guardrails to defend against attacks. LLM allows any input/output
- No proper filtering(Harmful outputs) → LLM can generate misleading, harmful or unsafe outputs to the user
- Poor validation & Sanitization → LLM gives output to user prompts without necessary validation or sanitization

Sensitive Information Disclosure

- Poor session isolation can cause LLMs to reveal sensitive information such as health records, financial details, etc.
- Potential privacy and security risks to occur when the model reveals personal or confidential info
- One user's information can appear in a different user's session unauthorized

Solutions

Solution: Preventing Prompt Injection & Unsafe Outputs

- Content Filtering(LLama Guard) → prevent unsafe and harmful inputs/outputs
- Input filtering → Orchestrator looks over user prompts with Llama Guard before they're sent to the LLM
- The orchestrator checks the models output with Llama Guard before sending it back to the user
- Block any unsafe or sensitive responses

Solution - Sensitive Information Disclosure

- **Session Separation**
  - Each chat session receives its own unique session ID
  - Orchestrator retrieves info only from the user's own session
  - Prevents any conversations from one session(user conversation) appearing another session
- **Llama Guard**
  - Scans user prompts and LLM responses for sensitive or unsafe content
  - Block anything that shouldn't be revealed

Post Safety Implementation Testing