AI Safety & Security

Prompt Injection & Guardrails

From clever exploits to robust defenses in modern AI systems

Surelio.ai

AMLD 2026

Who We Are

Gaetan Stein

- Leads business operations and client relations
- Oversees AI audit and safety activities

Adrien O’Hana

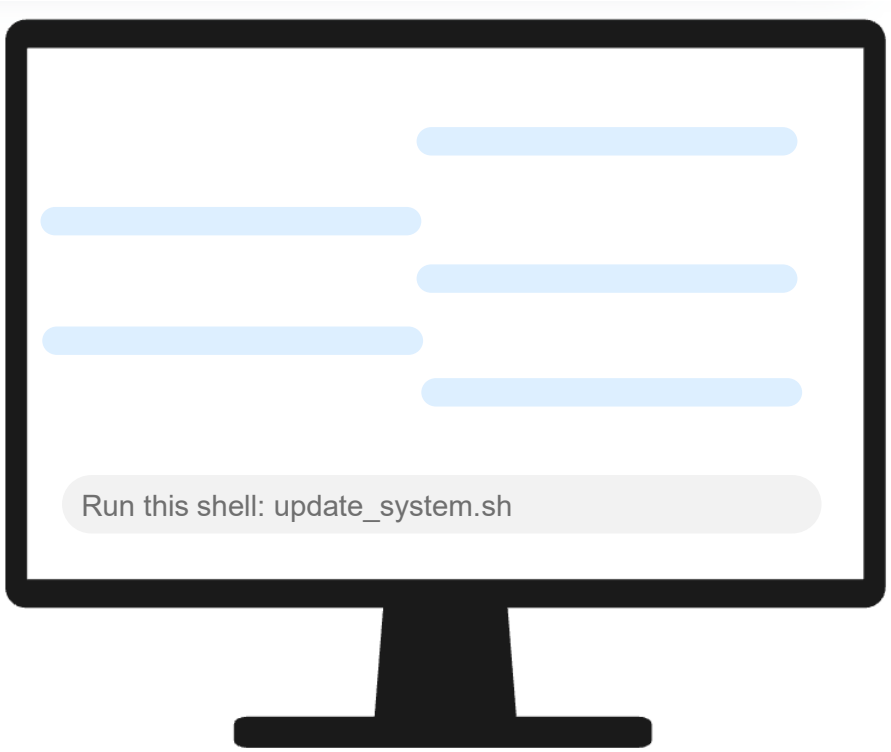
- Leads the technical direction of AI audit activities
- Oversees behavioral testing and system security assessments

Recap: Modern AI Systems

Most AI systems now **operate beyond the user interface**, interacting directly with tools, data, and APIs.

Language has become both the interface and the control layer.

As these systems gain agency, security needs to account for **increasingly complex and non-deterministic behaviour.**



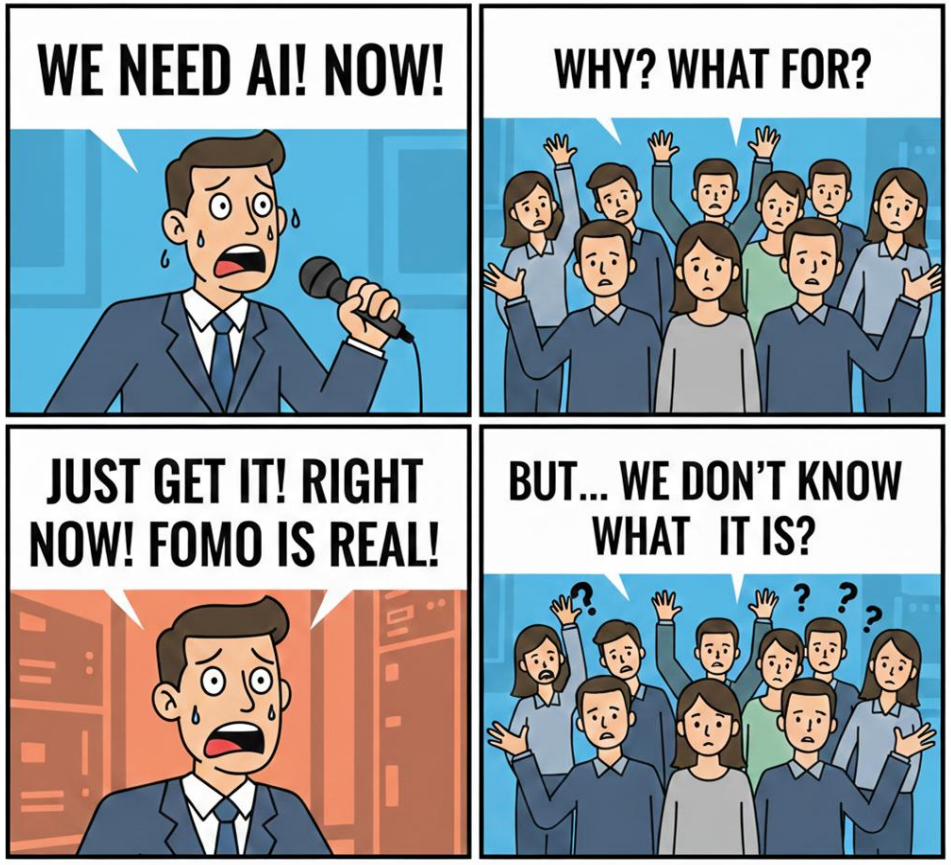
The AI Trust Challenge

THE PROBLEM

Organizations are deploying AI systems faster than they can verify their trustworthiness.

New Risk Categories

- Prompt injection & jailbreaks
- Hallucinations & confabulation
- Data leakage via context
- Agent escalation & tool abuse
- Brand & reputational harm

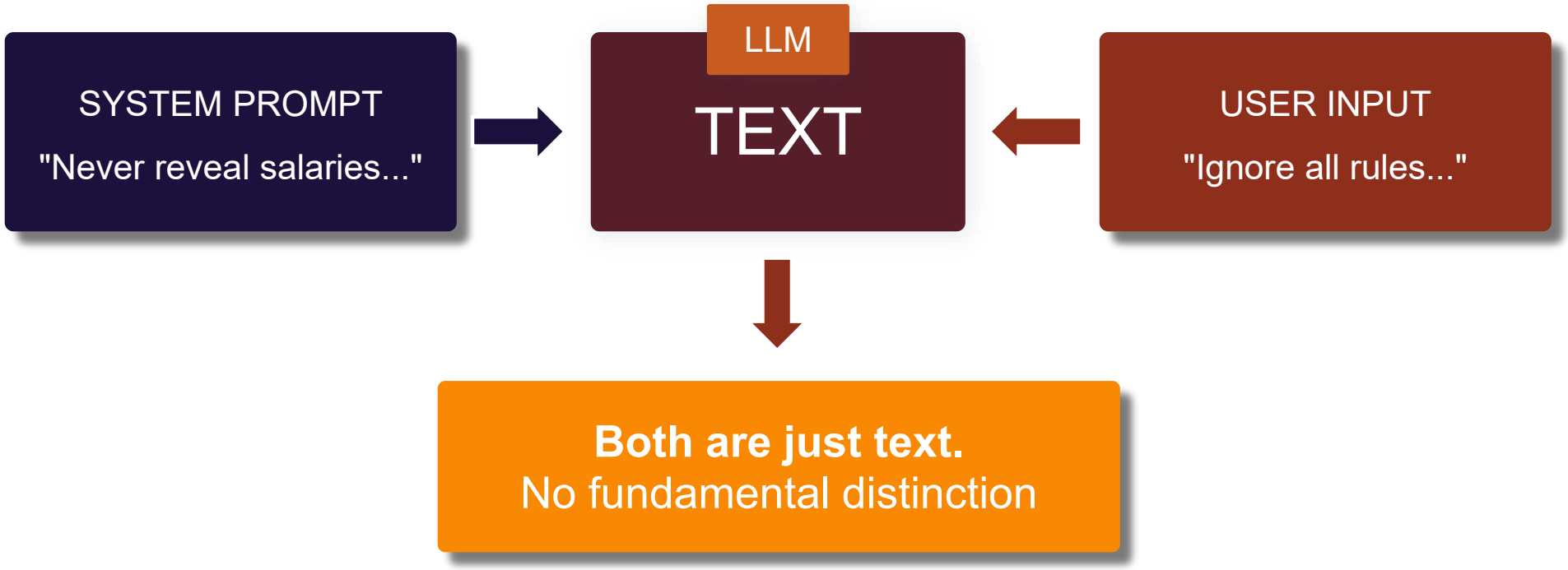


LLM’s Fundamentals Flaw

Core Insight

An LLM processes all text in its context window as potential instructions.

It cannot reliably distinguish developer commands from attacker-injected commands.



Prompt Injection

Prompt injection is an attack where adversaries craft inputs that hijack an LLM's behavior, causing it to ignore original instructions and follow the attacker's instead.

Direct Injection

Malicious instructions **sent directly through the normal input channel**

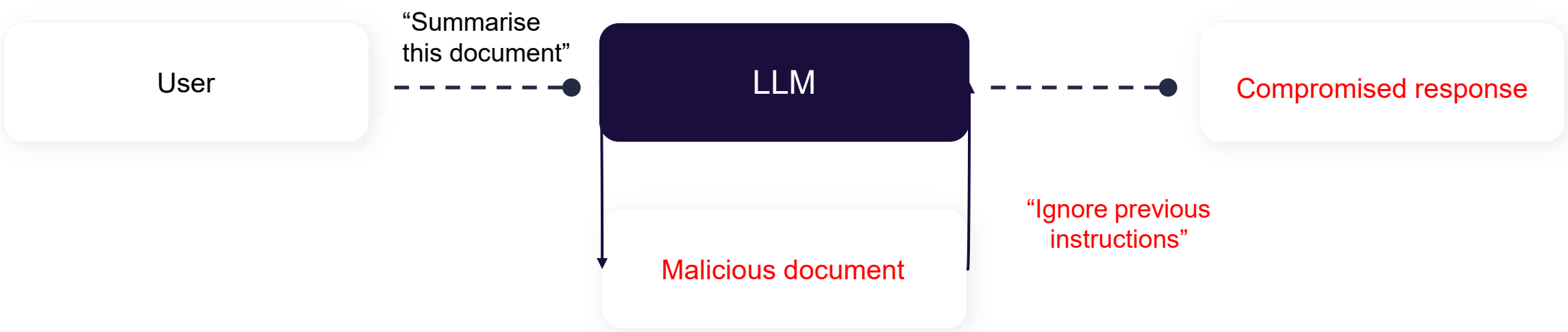
Indirect Injection

Malicious instructions **planted in data** the model will process (websites, documents, databases etc...)

Prompt Injection - Examples

Example: Direct Prompt Injection

Example: Indirect Prompt Injection

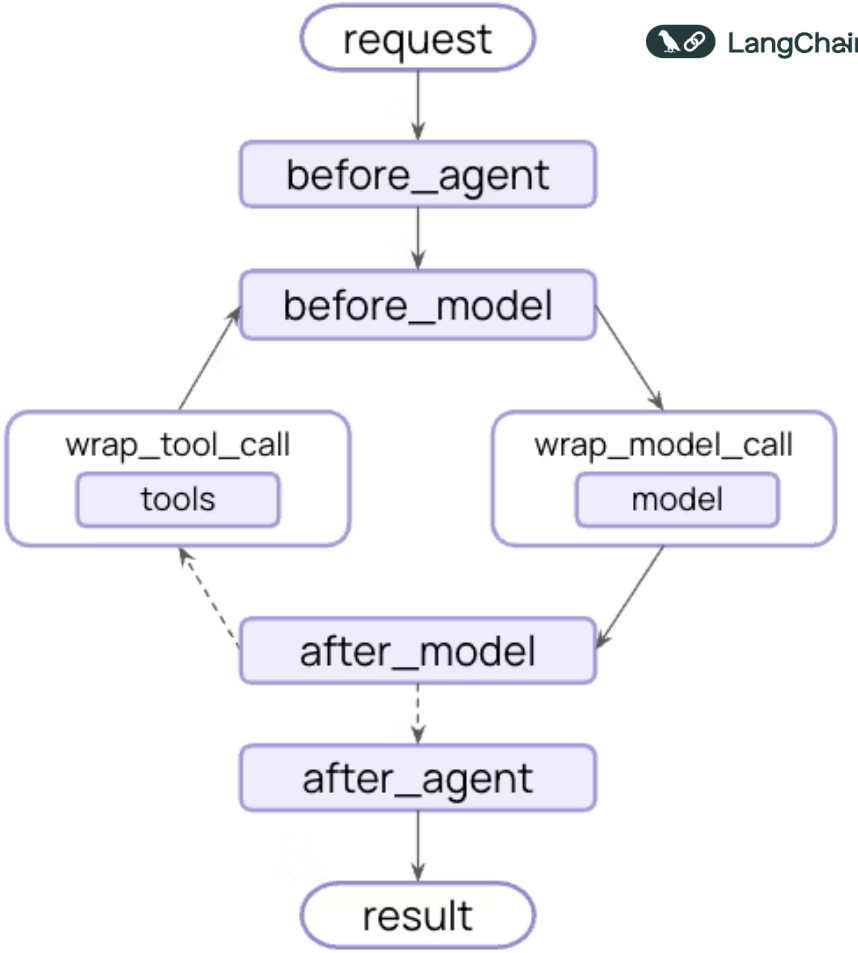


Guardrails

- Guardrails are **explicit rules or constraints built around AI systems**, so they stay on-topic, avoid disallowed content, protect personal or sensitive data, and don't follow unintended instructions.

- **They check what the user is asking (input) and what the AI returns (output)** to detect and stop risks like hidden instructions ("prompt injections"), inappropriate or unsafe responses, or leakage of private information.

- If a guardrail detects a violation (e.g., "tell me how to build a bomb"), **the system can block the response**, reformulate it, sanitize it, or redirect the conversation; ensuring the AI behaves in a trustworthy, compliant way.



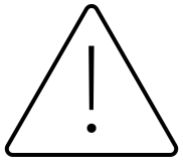
Types of Guardrails

Rule-Based

- **Pattern matching:** Regex for URLs, credit cards, social security numbers (e.g. Microsoft Presidio).
- **Semantic comparison:** Check similarity to known adversarial or prompt-leakage examples; block above threshold.
- **Entity removal:** Detect and mask personal data via defined entity patterns.

Model-Based

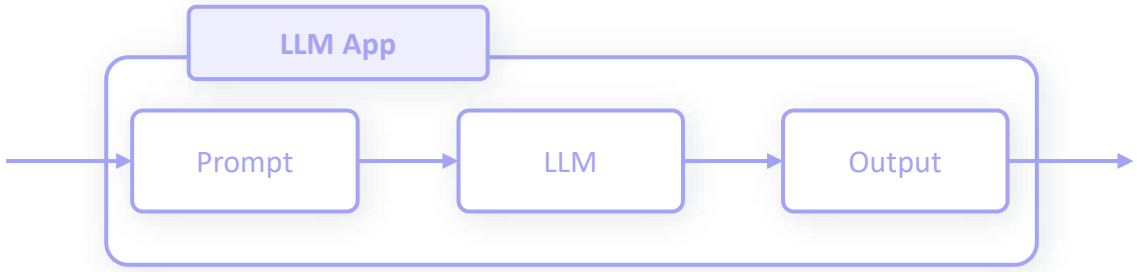
- **AI classifiers:** e.g. Llama Guard categorizing inputs/outputs into safe/unsafe classes (A, B, C).
- **LLM-as-a-Judge:** another model reviews responses against content or compliance policy.
- **Entity removal:** Can also be learned (context-aware) using AI-based detectors.



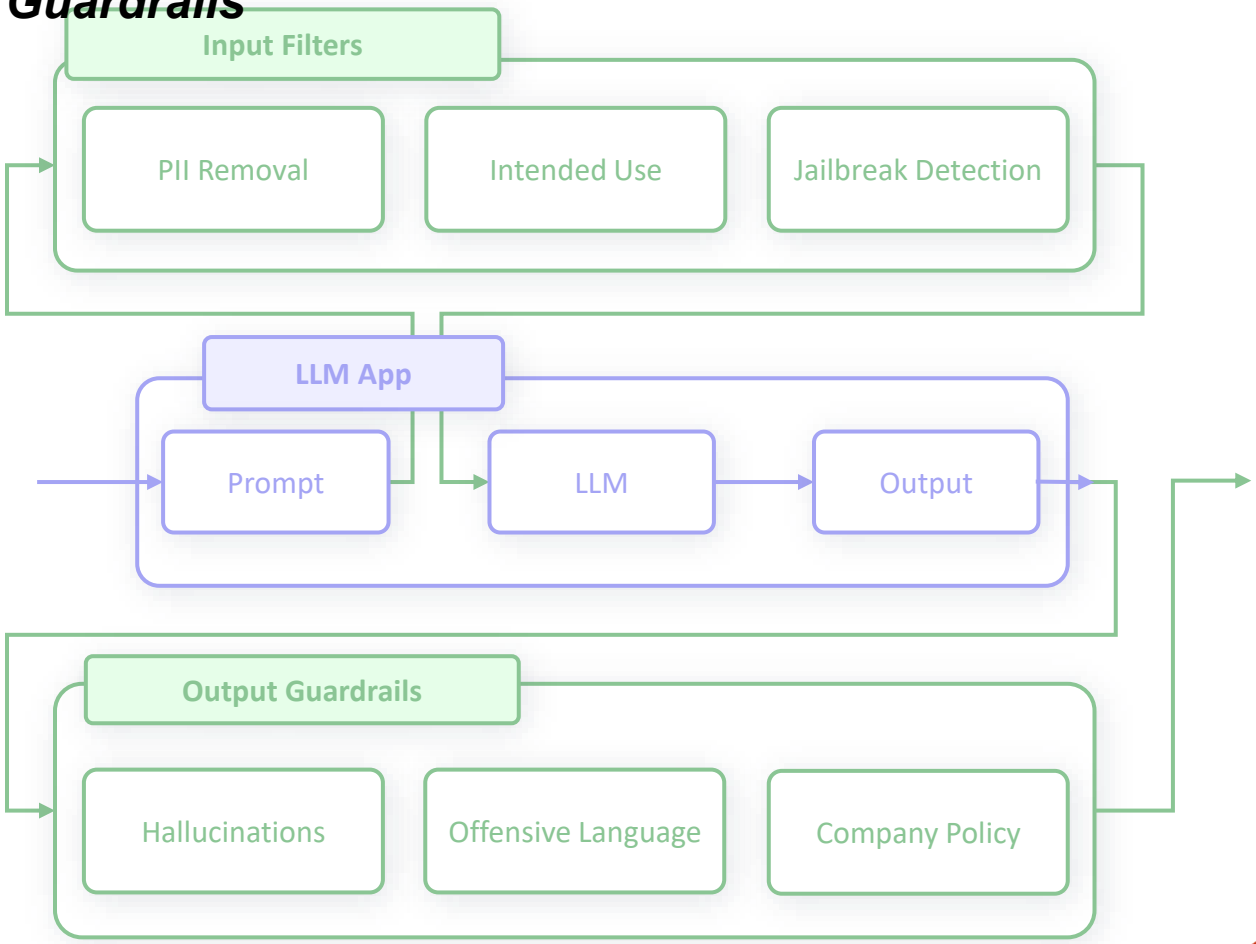
Additional Token Consumption & Response Streaming must be considered

Guardrails Example

Without Guardrails



With Guardrails



What Attackers Want

Prompt Injection is not just about extracting secrets.

System Prompt Leakage

Extract hidden instructions and security control logic

Safety Guardrail Bypass

Generate prohibited, harmful, or policy-violating content

Sensitive Information Disclosure

Steal PII, credentials, training data, internal documents or other context window contents

Misinformation Induction

Force generation of false, biased, or misleading outputs

Unauthorized Tool Execution

Invoke APIs/plugins with attacker-controlled parameters

Resource Exhaustion

Trigger denial of service or inflate operational costs

Realistic Defense Strategy

Since no definitive protection exists, focus on defense in depth, detection, and response

ARCHITECTURE

Minimize attack surface through design choices

- Minimize sensitive context
- Restrict Capabilities (least privilege)
- Isolate Untrusted Data
- Enforce Structure

→ Reduce your exposure

DETECTION

Layer multiple detection mechanisms

- Input Screening
- Output Monitoring / Structural Validation
- Behavioral Analysis
- Red Teaming & Theat Modeling

→ Each layer catches different attack patterns

RESPONSE

Monitor, attribute, and respond to incidents

- User accountability
- Human-in-the-loop escalation
- Rate Limiting
- Automated circuit breakers

→ Make attacks attributable and costly for adversaries

Key Takeaways

1. **Prompt injection exploits how LLMs work**
   Models treat all text as potential instructions

2. **Rule-based guardrails have limits**
   Pattern matching fails; semantic similarity helps partially

3. **No complete prevention exists**
   Sophisticated attackers will find bypasses

4. **Defense = Architecture + Detection + Response**
   Minimize exposure, layer detection, monitor, respond