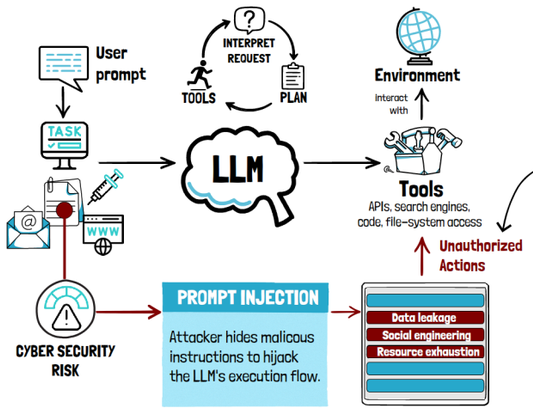
PROBLEM SPACE

DESIGN PATTERNS FOR SECURING LLM AGENTS AGAINST PROMPT INJECTION

arXiv > cs > arXiv:2506.08837

Luca Beurer-Kellner, Beat Buesser Ana-Maria Creţu, Edoardo Debenedetti, Daniel Dobos, Daniel Fabian, Marc Fischer, David Froelicher, Kathrin Grosse, Daniel Naeff, Ezinwanne Ozoani, Andrew Paverd, Florian Tramèr, Václav Volhejn

LLM Agents



DEFENCE STRATEGIES

LLM-LEVEL

Adversarial training
Make LLM more resistant to known attack patterns

Prompt Engineering / Spotlighting
Data/instruction separation
Rules for allowed behavours

Reduce attack surface, but can be bypassed

USER-LEVEL

Human-in-the-loop
Ask user to confirm sensitive actions

Limits automation

Users might get fatigued or distracted

SYSTEM-LEVEL

Input/Output Filtering
Scan for malicious known patterns (e.g. using ML models)

Heuristic, can be bypassed

Isolation Mechanisms
Sandboxing code execution, permission boundaries

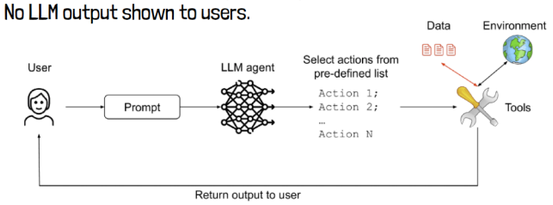
Contain consequences of successful attack

Core Principle

Once un LLM agent ingests untrusted input, it must be constrained to prevent that input from triggering any consequential actions

Pattern 1: Action Selector

Maps instructions to a fixed set of safe actions.
Actions and their allowed inputs are pre-determined.
No LLM output shown to users.

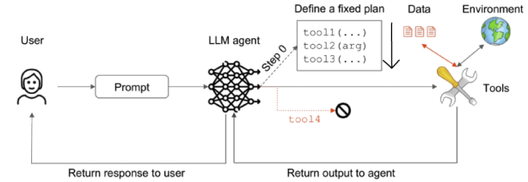


✓ Blocks prompt injection
No unauthorized actions / outputs

✗ No flexibility, can only do what's hardcoded

Pattern 2: Plan-Then-Execute

Agent generates a complete, detailed plan (sequence of tools) BEFORE seeing any untrusted data. Placeholders filled in later.



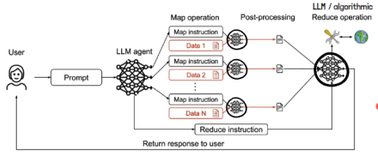
✓ Plan and fixed parameters cannot be changed **--> Control-flow integrity**

✗ Cannot adapt to previous action results

Pattern 3: LLM Map-Reduce

generalization of

Processes each input in isolation. Each LLM call maps input to a structured format (e.g. JSON).
Reduce step never sees untrusted input.

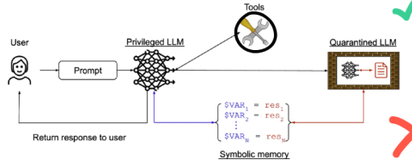


✓ Injections are contained
Schema validation blocks malicous outputs

✗ More complex orchestration

Pattern 4: Dual LLM

Proposed by S. Willison. Privileged LLM never sees untrusted input.
Quarantined LLM handles data (not tool access).



✓ Compromised Q-LLM cannot trigger privileged actions

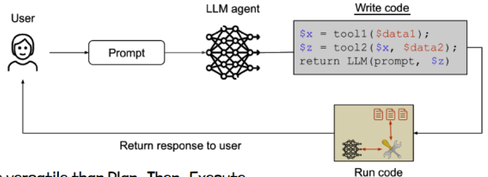
Flexible: P-LLM can prompt Q-LLM with arbitrary tasks

✗ More complex orchestration

Corner cases where Q-LLM output can lead to issues

Pattern 5: Code-Then-Execute

Generates an entire program BEFORE seeing any untrusted input.
Input is supplied only after the code is finalized.

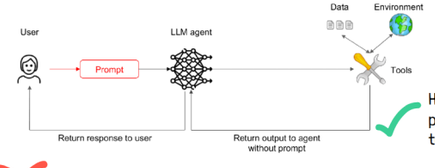


✓ More versatile than Plan-Then-Execute
Security policies to control tool/data flows (CaMeL, Google)

✗ Needs robust interpreter and policy enforcement

Pattern 6: Context Minimization

Extracts a structured representation of user's intent, then discards the original prompt before actions or responses.



✓ Hidden instructions in prompt can't affect tool use or output

✗ Can't respond with full nuance based on prompt phrasing

Case Studies

Case Study	Action-selector	Plan-then-exec.	Map-reduce	Dual LLM	Code-then-exec.	Context-min.
§4.1 OS Assistant	✓	✓	✓	✓		
§4.2 SQL Agent		✓				
§4.3 Email & Calendar Assistant		✓		✓	✓	
§4.4 Customer Service Chatbot	✓			✓	✓	✓
§4.5 Booking Assistant				✓	✓	
§4.6 Product Recommender			✓			
§4.7 Resume Screening Assistant			✓	✓		
§4.8 Medication Leaflet Chatbot						✓
§4.9 Medical Diagnosis Chatbot						✓
§4.10 Software Engineering Agent				✓		

Table 1: Overview of case studies described in this work and design patterns that apply.

BEST PRACTICES for all LLM Agents

Sandboxing
Run risky actions in isolated, low-privilege environments (e.g. run code in containers with r/o access)

Structured Output
Constrain outputs to strict, verifiable formats (e.g. JSON with allow-list of fields)

Permissions
Agent should never act beyond user's permissions (least privilege principle)

User Confirmation
Require approval for high-stakes actions (be mindful of user fatigue)

Data Attribution
Expose what inputs/sources contributed to decisions (robust attribution is an open challenge)

Per-User Rate Limits
Limit requests per minute/day
Prevent resource exhaustion
Slow down jailbreak discovery

Content Moderation Lockouts
Users who repeatedly trigger guardrails should be suspeneded

YouTube
youtube.com/@donatocapitella