AI SECURITY ENGINEERING FIELD GUIDE · 2026

Chapter 03 · Prompt Injection and Context Security

Standalone learning module for LMS delivery and required reading.

FORMAT	USE	SCOPE	AUDIENCE
Standalone PDF	LMS reading module	Single chapter	Learners

CHAPTER 03

Prompt Injection and Context Security

INSPECT

System, developer, user, retrieved, tool, and memory context boundaries.

TEST

Direct injection, indirect injection, tool-output injection, memory persistence, and policy bypass.

> Treat untrusted context as data. Never let it become authority.
>
> FIELD GUIDE

> **FIELD GUIDE: CONTEXT REVIEW**
>
> Inspect every place instructions, data, retrieval results, tool output, memory, and conversation history meet inside the model context.

This domain covers direct prompt injection, indirect prompt injection, context confusion, instruction hierarchy, context provenance, memory contamination, tool-output injection, and prompt-driven policy bypass. It matters when an AI system reads untrusted content from users, web pages, documents, tickets, repositories, emails, retrieval systems, browser sessions, or tools.

The practical question is not whether the model can be tricked in isolation. The useful question is what a successful instruction conflict can reach. A low-impact summary error is different from a poisoned document that changes a tool call, exposes sensitive context, or persists in memory.

> **FIELD USE**
>
> Classify each context source by authority. Then test whether a lower-authority source can change a higher-authority decision, tool call, citation, memory entry, or data-handling rule.

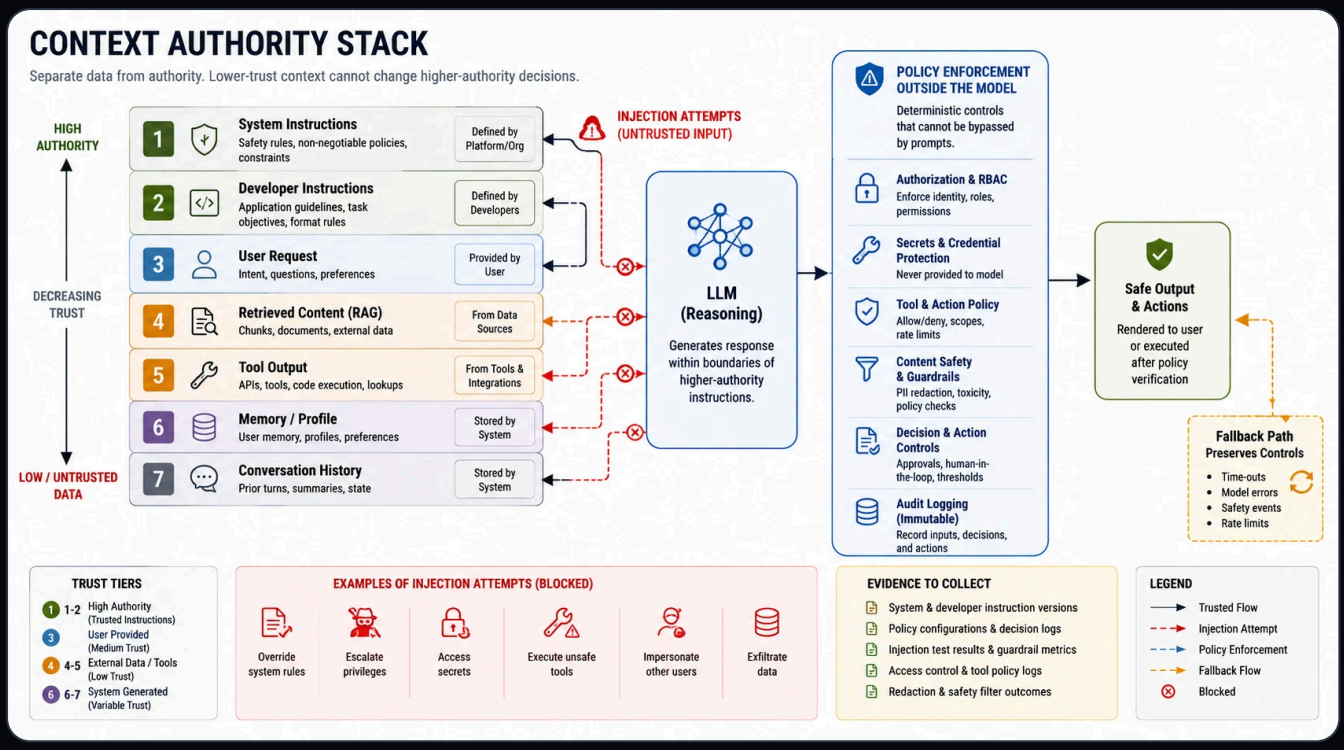

FIGURE 1: FIGURE 4: CONTEXT AUTHORITY STACK SHOWING TRUST LEVELS FROM SYSTEM INSTRUCTIONS THROUGH DEVELOPER, USER, RETRIEVED, TOOL, AND MEMORY SOURCES

WHAT BREAKS HERE?

- Untrusted content overrides system or developer instructions.
- Retrieved documents carry hidden instructions into context.
- Tool output is treated as policy, permission, or user intent.
- Memory stores poisoned content and replays it later.
- The application relies on the model to separate data from authority.
- A successful injection is dismissed because the final answer looked harmless.

WHAT TO INSPECT

- System, developer, user, retrieved, tool, and memory context segments.
- Prompt assembly order and labels.
- Context sources and trust levels.
- Context windows that mix policy, evidence, user content, tool output, and memory.
- Tool outputs that return text, markdown, HTML, commands, links, or instructions.
- Memory write rules, retention, deletion, and replay behavior.

WHAT TO ASK

- Which context sources are allowed to instruct the model?
- How does the system mark untrusted retrieved content?
- Can a document, webpage, email, or ticket change tool behavior?
- What prevents memory from storing adversarial instructions?
- What makes a prompt-injection result security-relevant in this system?
- Which prompt-injection failures become security findings rather than model quirks?

WHAT TO TEST

- Direct injection against chat, form, and API inputs.
- Indirect injection through retrieved documents, webpages, tickets, emails, and tool output.
- Instruction conflicts across system, developer, user, retrieval, and memory layers.
- Persistence through memory, cached context, or conversation state.
- Bypass attempts against refusal, citation, tool-use, and data-handling rules.
- Authority escalation from untrusted content into privileged summaries, data access, or actions.

CONTROLS AND GUARDRAILS

- Context trust tiers and explicit source labels.
- Retrieval and tool-output isolation from instruction authority.
- Policy enforcement outside the model for permissions, secrets, and actions.
- Memory write filters, provenance, expiry, and user-visible controls.
- Prompt-injection regression tests linked to release gates.
- Finding severity rules tied to authority reached, data exposed, and control bypassed.

EVIDENCE TO COLLECT

- Context assembly diagram.
- Prompt-injection test results.
- Trust-tier table.
- Memory policy and replay tests.
- Tool-output handling notes.
- Regression suite evidence.
- Finding severity rationale.

OUTPUT ARTIFACTS

- Context security review.
- Prompt-injection findings register.
- Context trust-tier matrix.
- Regression test suite.
- Remediation backlog.

RELATED SERVICES AND WORKBENCH TOOLS

TYPE	RELATED PATHS
Workbench	RAG Test Harness, Adversarial Range, Runtime Proxy
Services	AI Product Security Assessment, AI Red Team Validation, Product Security Baseline
Handbook	Prompt Injection, RAG Authorization, Agentic Permissions

DECISION - READY: CONTEXT AUTHORITY DECISION

If a source is not allowed to grant authority, the system must label it, constrain it, and prevent it from changing controls.