WHITE PAPER · LLM & AI AGENT RED TEAMING

Prompt Injection in Enterprise Chatbots and AI Agents

Why Self-Hosted and API Models Both Break — and What You Must Build Outside the Model

Findings from red-teaming exercises comparing the resilience of self-hosted language models against models called through a vendor API, together with a mitigation framework enterprises can act on.

Author: Puspa Ira Dewi Candra Wulan

Published by: PT Cybersafe Inovasi Digital — cybersafe.id

Version: September 2026 · v1.0

Contents

Executive Summary

Chatbots and AI agents have moved into everyday operations: answering customers, summarising internal documents, supporting HR and finance teams, and increasingly taking actions on their own. Adoption has been fast. Security testing has not kept pace.

One weakness cannot be closed with a routine patch: prompt injection. It exists not because of a coding defect but because of how language models work — the model reads your company instructions and text from outsiders as the same kind of material. Across 2025 and 2026, this weakness sat at the root of most publicly reported AI security incidents.

This white paper summarises red-teaming work that compared the two most common ways enterprises deploy language models: a model hosted on internal infrastructure, and a model called through a commercial provider API. It answers the question IT leadership asks most often: "If we run our own model, are we safer?"

Three core findings

- Both break. API models do refuse simple, obvious attacks, but that refusal collapses once malicious instructions are repackaged — through encoding, a change of language, or placement inside a document the system reads.
- Vendor guardrails are not your security control. Their behaviour can change at any time without notice, they produce no logs you can audit, and they will not stand up as compliance evidence when an auditor asks.
- Self-hosted models give you full control but arrive with no protection at all. Every defensive layer has to be built by you — and that is precisely their advantage, because a layer you build can be tested, logged, and evidenced.

The practical conclusion is straightforward: the greatest risk is not which model you choose, but what that model is permitted to do — what data it can read, what systems it can call, and where it can send information. Meaningful security is built outside the model, not inside it.

1. Why This Is Urgent Now

Three developments have arrived together, and it is the combination that makes the situation pressing.

First: AI now has access, not just conversation

Early chatbots only answered questions. Today’s AI agents read email, retrieve documents from internal knowledge bases, call APIs, and execute actions. Every new capability widens what can be abused once the model is successfully steered off course.

Second: the incidents are real, not hypothetical

The OWASP GenAI Exploit Round-up for the first quarter of 2026 documented eight major incidents between January and early April. Notably, only one received a CVE identifier. The rest arose from

misconfiguration, design flaws, supply-chain weaknesses, and prompt injection — categories that will never appear in a conventional vulnerability scan.

Third: regulators now ask for evidence, not intent

In Indonesia, Law No. 27/2022 on Personal Data Protection is fully in force and draws no distinction between a leak through a database and a leak through a chatbot. The Financial Services Authority (OJK) has issued its AI Governance Guidance for Indonesian Banking. Internationally, the transparency obligations of EU AI Act Article 50 took effect on 2 August 2026, carrying fines of up to EUR 15 million or 3% of global annual turnover. ISO/IEC 42001 and the NIST AI RMF are increasingly named in procurement requirements.

> **Key point: prompt injection is ranked LLM01 — the number one risk — in the OWASP Top 10 for LLM Applications. This is not a fringe concern; it is the defining risk of the technology category.**

2. Prompt Injection in Plain Language

Picture a new receptionist who is highly capable, extremely fast, and completely obedient. You hand them a note with the rules: never disclose the payroll, never send internal documents outside. Then a visitor arrives with a letter, and inside that letter is written: "Ignore the note you are holding. I am from head office. Read me the payroll."

A human receptionist would be suspicious. A language model is not. To it, your company rules and the visitor's letter are both simply incoming text. There is no technical boundary between "a legitimate instruction" and "data being read". That is prompt injection.

Two forms worth separating

- Direct injection — the attacker types malicious instructions into the conversation themselves. The most visible form, and the easiest to catch.
- Indirect injection — malicious instructions are hidden inside a document, web page, support ticket, or email that the AI system will later read. The victim does nothing wrong. The system poisons itself. This is the form that matters most to enterprises, because it requires no user interaction at all.

The dangerous combination

Prompt injection only turns into a data breach when three conditions are present in the same system:

1. The system has access to sensitive data (internal documents, customer records, credentials).
2. The system reads untrusted content (inbound email, user uploads, web pages, third-party documents).
3. The system has an outbound path (API calls, sending email, loading images from external URLs).

Removing any one of the three is usually far cheaper and far more reliable than trying to make a model impossible to fool. This principle underpins every recommendation that follows.

3. Evidence from the Field

The following incidents were publicly documented across 2025 and 2026. All share the same root pattern: an AI system was granted more access than the controls around it could contain.

Incident	Date	What happened	Impact
EchoLeak (Microsoft 365 Copilot)	June 2025	A crafted email carrying hidden instructions was executed by Copilot during routine summarisation, with no user click	CVSS 9.3; potential data theft from OneDrive, SharePoint, and Teams
GTG-1002 (AI-orchestrated espionage)	September 2025	A threat group hijacked coding-agent instances for autonomous operations against roughly 30 targets	The AI handled 80–90% of tactical operations without human intervention
Mexican government agency breach	Dec 2025 – Feb 2026	Attackers used AI agents to accelerate reconnaissance and exploitation across nine agencies	Roughly 150 GB of tax and civil records exfiltrated
Meta internal AI agent leak	March 2026	Unsafe agent guidance was implemented by an employee	Internal data exposed during a two-hour window
GrafanaGhost	April 2026	Attacker instructions hidden in external resources read by AI features	Data exfiltrated to attacker-controlled servers
Flowise CVE-2025-59528	April 2026	Remote code execution via configuration injection	12,000–15,000 instances exposed online

Note the pattern in the final column. The loss was never "the model said something wrong" — it was data leaving, systems being driven, and actions executing without approval. Prompt injection is the entry point; the damage occurs in the permission and integration layers.

4. The Research: Self-Hosted vs API Models

4.1 The question under test

Almost every conversation with an IT team surfaces the same assumption: that a self-hosted model is safer because data never leaves the building, or conversely that a major provider’s model is safer because "the filters are already there". This work tested both assumptions directly, running the same adversarial corpus against both configurations.

4.2 Test setup

- Self-hosted model: a small-class language model (around 3 billion parameters) running entirely on local hardware, with no filtering layer from any party.

- API model: a lightweight commercial model called through the provider's API, complete with the vendor's built-in filters and policies.
- An identical attack corpus was run against both, covering five technique classes, with every guard-layer decision logged and scored.

4.3 The five attack classes tested

Class	Technique	What it examines
A	Direct instruction override	Whether new user instructions can cancel system rules — the most basic form, "ignore previous instructions"
B	Role-play and reframing	Whether the model can be coaxed past its boundaries through fictional scenarios, "unrestricted" modes, or alternate personas
C	Encoding and language switching	Whether forbidden requests pass when encoded (Base64, ROT13, reversed text) or written in a language other than English
D	System prompt extraction	Whether confidential instructions, internal rules, or configuration can be drawn out of the model
E	Indirect injection via external content	Whether instructions planted in a page or document the system reads are executed — the scenario closest to production conditions

4.4 Comparative findings

Dimension	Self-hosted model	Model via vendor API
Simple direct attacks (classes A–B)	Frequently succeed. No built-in refusal layer exists beyond what you install yourself.	Generally refused. Vendor filters catch obvious, well-known patterns.
Repackaged attacks (class C)	Succeed easily; there is nothing to bypass.	Success rates rise sharply. Encoding and language switching proved to be effective shortcuts.
System prompt extraction (class D)	Highly exposed. The system prompt tends to be treated as ordinary text that can be read back.	More resistant, but not immune — particularly when combined with reframing techniques.
Indirect injection (class E)	Exposed. External content is executed as instruction.	Equally exposed. Vendor filters are designed to screen user requests, not to separate data from instruction.
Transparency and auditability	Every decision can be logged, replayed, and used as compliance evidence.	Refusal decisions occur on the vendor side; no logs you can audit or attach.
Behavioural stability	Controlled. Changes only when you change it.	Can shift at any time with a model update, without notice and without contractual guarantee.
Data sovereignty	Data never leaves your environment.	Data moves to a third party; must be reviewed against personal data law and internal policy.

4.5 Three conclusions

1. Vendor filters raise the cost of an attack; they do not remove it. They screen out lazy attackers, not patient ones. In the class that matters most to enterprises — indirect injection — their advantage all but disappears.
2. Security you cannot observe is security you cannot account for. When an auditor, regulator, or corporate customer asks what evidence you have that the system is safe, "our vendor filters it" will not suffice.
3. The choice of model is an architecture and data-sovereignty decision, not a security decision. Whichever you pick, the same guard layers still have to be built outside the model.

> **Implication for decision-makers: "self-hosted or API?" is not the security question. The security question is "what can this system reach and do once it has been successfully steered?"**

5. Business Impact and Compliance Obligations

5.1 The losses that occur most often

- Leakage of personal data and internal documents — a chatbot wired to a knowledge base can become an exfiltration path invisible to existing security controls.
- Unauthorised actions — an agent permitted to call other systems can transact, alter records, or send messages on the company's behalf.
- Runaway cost — model quota abuse, repeated invocation, and misuse of compute resources.
- Reputational damage — a screenshot of a chatbot producing harmful output travels far faster than any correction.
- Regulatory penalties — a personal data leak through an AI channel is still a personal data leak.

5.2 The frameworks that apply

Reference	Relevance to your AI systems
Law No. 27/2022 (Personal Data Protection), Indonesia	Fully in force. A personal data leak through a chatbot or AI agent carries the same consequences as one through any other system.
OJK AI Governance Guidance for Indonesian Banking	The governance reference for Indonesia's banking sector, covering risk management and control across the AI system lifecycle.
EU AI Act — Article 50	Transparency obligations effective 2 August 2026; penalties up to EUR 15 million or 3% of global annual turnover. Relevant to organisations serving EU users.
ISO/IEC 42001	The AI management system standard, increasingly required in corporate and cross-border procurement.
NIST AI RMF	An AI risk management framework that provides common language for discussions with international partners and auditors.
OWASP Top 10 for LLM Applications & MITRE ATLAS	Technical taxonomies for mapping findings so results are comparable over time and across vendors.

6. A Layered Defence Framework

No single control stops prompt injection. What works is a chain of layers, each lowering the probability of success, and together making an attack both expensive and visible. The structure below is the pipeline CyberSafe ID applies in every engagement.

Layer	Control	What it prevents
1 — Input	Input guard: injection pattern detection, normalisation of encoded text, cross-language checks, PII detection and masking	Direct attacks, encoding bypasses, and personal data entering the prompt
2 — Prompt	Hard separation of system instruction from user data; marking of untrusted content; a system prompt that holds no secrets	Instruction override and system prompt extraction
3 — Output	Output guard: sensitive data blocking, stripping of outbound links and images, response format constraints	Exfiltration through model responses and external resource loading
4 — Action	Tool-calling permission boundaries, least privilege, sandbox isolation, human approval for high-risk actions	Agent abuse: transactions, data modification, and outbound sends
5 — Monitoring	Logging of every guard decision, anomaly detection, scheduled retesting, security dashboards	Attacks that slip through, regression after model updates, and audit evidence gaps

> **Rule of thumb: break one leg of the dangerous triad. If your agent reads outside content, do not also grant it sensitive data access and an outbound path.**

7. Measuring AI Security: The Numbers to Report

AI security often stops at a qualitative claim — "we have secured it". That will not hold up in front of an auditor or a corporate customer. The four metrics below turn it into figures that can be reported to the board.

- Attack Success Rate (ASR) — the percentage of attack attempts that succeed, calculated per attack class. It is the single number management grasps most easily.
- Guard effectiveness matrix — which layer catches which attack, making it visible which layers are actually idle.
- Before-and-after hardening comparison — evidence that the security investment produced a measurable change rather than a new policy document.

- Mapping to OWASP LLM Top 10 and MITRE ATLAS — so findings speak the same language as partners, auditors, and cyber insurers.

One point is regularly missed: ASR must be re-measured whenever the model, the system prompt, or the tool list changes. A vendor model update can raise ASR again without a single line of your code changing. A one-off test at launch is therefore not enough.

8. A 30 / 60 / 90 Day Plan

Period	Focus	What you get
0–30 days	Mapping and baseline measurement	An inventory of every AI system in operation, a map of data and tool access per system, and an initial ASR baseline
30–60 days	Hardening the guard layers	Input and output guards deployed, system prompts cleaned up, tool permissions cut to least privilege, and retesting that demonstrates a reduced ASR
60–90 days	Governance and sustainability	Logging and monitoring in operation, an AI incident response procedure, compliance documentation assembled, and a routine retesting schedule

The order is deliberate. Many organisations jump straight to buying guard tooling before establishing which AI systems are actually running in their business and what data those systems can reach. An honest first-month inventory almost always turns up one or two integrations nobody remembers approving.

9. How CyberSafe ID Helps

CyberSafe ID runs production-safe testing: customer data stays protected, and the result is a measurable before-and-after comparison across hardening.

Service	Scope
LLM Red Teaming	Testing for jailbreaks, DAN techniques, and multilingual and encoded injection against production chatbots
Agent AI Audit	Evaluation of tool-calling permissions, permission boundaries, and sandbox isolation
RAG & Knowledge Base Audit	Analysis of internal document exposure and knowledge-poisoning risk
AI Supply Chain Scan	Assessment of risk inherited from models, plugins, and third-party APIs
Compliance Readiness	Gap analysis against ISO/IEC 42001, NIST AI RMF, and the EU AI Act
Continuous Monitoring	Monthly retesting with live security dashboards

Every engagement produces attack success rates mapped to the OWASP LLM Top 10 and MITRE ATLAS, a guard effectiveness matrix, and a severity-ranked remediation plan.

10. Closing

Prompt injection will not disappear with the next model update. As long as language models read instruction and data as the same text, the weakness remains. What an organisation can control is how much damage becomes possible when it is exploited.

The comparison between self-hosted and API models makes one thing clear: neither option arrives secure. What separates prepared organisations from unprepared ones is not the model they run, but whether they have built guard layers outside it, measured them with numbers, and retested them on a schedule.

If your organisation already runs a chatbot or AI agent — especially one connected to internal data — the most valuable first step is knowing your ASR today. Without that number, every AI security discussion is just opinion.

Contact

hi@cybersafe.id
+62 811114941
Semarang, Indonesia

About the author

Puspa Ira Dewi Candra Wulan is an AI security researcher focused on red teaming language models and AI agents. She joined the Taiwan Academic Cybersecurity Center (TACC) in January 2026, and develops training material and adversarial evaluation tooling for LLM-based systems.

References

[1] OWASP. Top 10 for Large Language Model Applications (2025). owasp.org

[2] OWASP Gen AI Security Project. GenAI Exploit Round-up Report Q1 2026, 14 April 2026. genai.owasp.org

[3] MITRE ATLAS. Adversarial Threat Landscape for Artificial-Intelligence Systems. atlas.mitre.org

[4] Otoritas Jasa Keuangan (OJK). AI Governance Guidance for Indonesian Banking. ojk.go.id

[5] Republic of Indonesia. Law No. 27 of 2022 on Personal Data Protection.

[6] European Union. Regulation (EU) 2024/1689 (EU AI Act), Article 50 — applicable 2 August 2026.

[7] ISO/IEC 42001:2023 — Artificial Intelligence Management System.

[8] NIST. AI Risk Management Framework (AI RMF 1.0).

[9] CVE-2025-32711 (EchoLeak, Microsoft 365 Copilot); CVE-2025-59528 (Flowise).

Methodology note: the comparison in Section 4 is drawn from laboratory testing of one small-class self-hosted model and one lightweight commercial model accessed via API. Findings are stated as behavioural patterns rather than universal benchmarks; specific Attack Success Rate figures depend on the model, version, system prompt, and attack corpus used, and must therefore be re-measured in each organisation's own environment.