

Securing Agentic AI: Threat Models and Trust Boundaries for LLM-Based Agents and Tool-Use Protocols (2025)

Mianxiong Dong

ORCID: 0000-0002-2788-3451

Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Japan

M. F. Mridha

ORCID: 0000-0001-5738-1631

Department of Computer Science and Engineering, American International University-Bangladesh, Dhaka, Bangladesh

Muhammad Waqas

ORCID: 0000-0002-4659-783X

FAST School of Computing, FAST-NUCES Karachi, Karachi, Pakistan

Correspondence: Mianxiong Dong · contact details held in the journal record

Publication history. Assigned by the publisher to Cybersecurity Advances, Volume 2, Issue 1 (2025). This publisher-authorized digital edition was released on 29 October 2026 from the supplied final 2025 manuscript. Review-stage notices were removed and journal typography, pagination, metadata, and accessibility were normalized; substantive scholarly content was preserved.

© 2025 the author(s). Published by AlphaTech Scholarly Press under the Creative Commons Attribution 4.0 International License (CC BY 4.0). No DOI is displayed unless and until registration is independently verified.

Background: 2025 turned large language models from chat systems into *agents* — systems that read content, decide on actions, and invoke privileged tools. Two interoperability protocols industrialized this shift: Anthropic's Model Context Protocol (MCP), whose June 2025 revision re-specified MCP servers as OAuth 2.1 resource servers, and Google's Agent2Agent (A2A) protocol, donated to the Linux Foundation in June 2025 with broad industry backing. The same year produced the attack record that defines the threat model: Invariant Labs' tool-poisoning disclosure (April 2025) and GitHub-MCP "toxic agent flow" exploit (May 2025), and the formalization of the "lethal trifecta" — private data, untrusted content, and external communication combined in one agent (June 2025). **Objective:** We consolidate the 2025 agentic-AI security record into an engineering threat model for tool-using agents and evaluate the mitigations that the protocol layer, the platform layer, and application design can supply. **Methods:** Qualitative synthesis of the 2024–2025 primary disclosures, protocol specifications, and governance instruments; construction of a six-class attack taxonomy and a six-class mitigation-coverage assessment. **Results:** The defining property of agentic systems is boundary collapse: untrusted content becomes instructions that drive privileged actions, so the classical separation between "data" and "execution" — the foundation of most security architecture — does not hold. Every major 2025 exploit is an instance of one pattern; mitigations partition cleanly into classes that reduce blast radius (capability-scoped tools, OAuth-scoped delegation, confirmation gates) and classes that detect misuse (provenance labeling, toxic-flow analyzers), but no 2025 mechanism eliminates the confused-deputy problem itself. **Conclusions:** Agentic-AI security in 2025 is an exercise in *boundary engineering under an acknowledged residual risk*: scope every capability, bind identities to delegation chains, gate irreversible actions, and instrument for detection — while treating prompt injection as unsolved and designing so that its exploitation is survivable.

Keywords: agentic AI; large language models; Model Context Protocol; Agent2Agent; prompt injection; tool poisoning; trust boundaries; AI governance

Contribution statement. This paper makes four contributions: (1) a primary-source-verified account of the 2025 protocol and disclosure record that defines agentic-AI security practice; (2) a six-class attack taxonomy grounded in documented 2024–2025 exploits, organized around the confused-deputy structure; (3) the "lethal trifecta" reframed as a trust-boundary design rule — the three factors are individually common and only jointly lethal, which makes their separation an actionable engineering requirement; and (4) a mitigation-coverage assessment identifying which controls reduce blast radius, which detect misuse, and which gaps remain open by design.

1. Introduction

The 2025 security conversation about large language models changed its subject. Through 2024, the discourse centered on what models *say* — hallucination, jailbreaks, content policy. In 2025 the discourse centered on what models *do*: vendors shipped agents that read email, browse repositories, query databases, execute code, and message colleagues, and two open protocols turned those capabilities into an ecosystem. Anthropic's **Model Context Protocol (MCP)**, released in late 2024 and revised through 2025, standardized how agents connect to tools and data sources; Google's **Agent2Agent (A2A)** protocol, announced in April 2025 and donated to the Linux Foundation in June 2025 with backing from AWS, Cisco, and more than a hundred supporting companies, standardized agent-to-agent communication [5]. An ecosystem in which an agent can discover tools from one vendor, execute code through another, and negotiate with an agent operated by a third party is, structurally, a new kind of distributed system — and it inherited none of the trust machinery that distributed systems security spent fifty years building.

The attack record arrived before the defenses did. In April 2025, Invariant Labs disclosed **tool poisoning attacks** against MCP, in which malicious tool descriptions hidden in a server's metadata hijack agent behavior across otherwise-trusted integrations [2]. In May 2025, the same group demonstrated a **toxic agent flow** against GitHub's official MCP server: an attacker files a public issue containing an indirect prompt injection; a developer's agent — acting with the developer's own credentials — reads the issue, is steered by the injection to examine private repositories, and is induced to create a pull request exfiltrating proprietary code, all without any exploit code or malware [3]. In June 2025, Simon Willison articulated the underlying condition as the "**lethal trifecta**": an agent combining access to private data, exposure to untrusted content, and the ability to communicate externally is exploitable in principle, regardless of the model's alignment quality [4]. Meanwhile the governance layer moved: the EU AI Act's obligations for general-purpose AI models took effect on 2 August 2025 [6], and NIST's Generative AI Profile had already supplied the risk-management vocabulary [8].

The security-engineering problem is precisely characterized by a structural observation: **the language model is a confused deputy**. It sits between untrusted content channels (documents, web pages, tool outputs, messages from other agents) and privileged action channels (file systems, code execution, networks, communications), and it converts the former into decisions about the latter. Every major 2025 exploit is an instance of this one pattern.

1.1 Research questions

RQ1: What does the verified 2025 protocol-and-disclosure record establish about agentic attack surfaces? **RQ2:** What taxonomy organizes the attack classes without overcounting variant labels? **RQ3:** Which mitigation classes — protocol-level, platform-level, and design-level — cover which attack classes, and where do gaps remain? **RQ4:** What should engineering teams building on MCP/A2A do now, under an acknowledged unsolved core problem?

1.2 Contributions

(1) The verified 2025 record (Section 2, Table 1). (2) A six-class attack taxonomy with mechanism-level distinctions rather than variant labels (Section 3, Figure 2, Table 2) (RQ1, RQ2). (3) The lethal-trifecta condition recast as a design rule — separation of the three factors — together with a mitigation-coverage matrix (Section 4, Figure 3, Table 3) (RQ3). (4) Engineering guidance for MCP/A2A adopters (Section 5) (RQ4).

2. Background: The 2025 Protocol and Disclosure Record

2.1 The protocol layer

MCP's trajectory through 2025 is itself a security story. The protocol's early revisions left authorization underspecified, and ecosystem practice mixed authentication models in ways that made delegation chains hard to reason about. The **June 2025 revision** addressed the core confusion: it reclassifies MCP servers as **OAuth 2.1 resource servers**, requires clients to implement **Resource Indicators (RFC 8707)** so that tokens issued for one server cannot be replayed to another, and removes legacy conveniences (JSON-RPC batching) that complicated security analysis [1]. This is the protocol layer absorbing a real attack lesson — token confusion across servers — into the specification. The A2A protocol, created by Google and donated to the Linux Foundation in June 2025 [5], standardizes agent-to-agent task delegation and is deliberately complementary: MCP connects agents to tools; A2A connects agents to each other. From a trust perspective, A2A generalizes the same boundary problem one level up: an agent's counterparties are now themselves agents with their own tool privileges and their own susceptibility to injection.

2.2 The disclosure record

Three 2025 disclosures define the empirical baseline. **Tool poisoning** (Invariant Labs, April 2025) showed that a malicious MCP server can embed instructions in tool descriptions — metadata the agent reads but the human may never inspect — and thereby manipulate the agent across sessions and override instructions from other, trusted servers [2]. **The GitHub MCP toxic flow** (May 2025) demonstrated the full indirect-injection chain against a first-party, widely trusted integration: attacker-controlled public issue content steering a privileged agent into cross-repository exfiltration [3]. **The lethal trifecta** (Willison, June 2025) supplied the generalization: given private data, untrusted content, and external communication co-located in one agent, compromise follows without any model-specific vulnerability [4]. Notably, none of the three involves model misbehavior in the alignment sense — the agents performed exactly the instructions they read. The vulnerability is architectural, not behavioral.

2.3 The governance layer

The EU AI Act's obligations for general-purpose AI model providers took effect 2 August 2025 [6]. NIST's AI Risk Management Framework defined the govern-map-measure-manage vocabulary [7], and its Generative AI Profile (NIST AI 600-1, July 2024) operationalizes those practices for generative systems — including agentic deployments — that federal guidance incorporates by reference [8]. These instruments regulate the *model and provider* layer; the exploits above live in the *system integration* layer. The gap between the two is where this paper's engineering guidance sits.

3. Methodology

The study applies a three-component qualitative synthesis.

Record verification (RQ1). Every protocol claim is sourced to the specification text itself (the 2025-06-18 MCP revision and its changelog [1]) and every exploit claim to the original disclosure [2][3]; industry-protocol facts (A2A governance) to the announcement [5]. Claims that circulate in secondary aggregation but lack a primary disclosure are excluded.

Taxonomy construction (RQ2). Candidate attack classes were harvested from the 2024–2025 disclosures and consolidated by *mechanism* rather than by variant name: two labels that share an underlying mechanism (e.g., "tool poisoning" and "rug pull" both exploit the tool-metadata trust channel at different times) are recorded as one class with two temporal variants. The result is six classes (Table 2), each defined by its trust premise — the assumption the attacker abuses.

Coverage assessment (RQ3). Six mitigation classes were compiled from the protocol specifications (OAuth resource-server semantics [1][10][11]), platform practice (confirmation gates, sandboxing), and design literature (provenance, least privilege). Each attack class was assessed qualitatively against each mitigation class, with every assignment tied to a mechanism argument (Table 3). The assessments are author judgments; Section 7 states the consequences.

4. The Attack Taxonomy (RQ2)

Figure 1 presents the structural baseline: the agent as a confused deputy between untrusted content channels and privileged action channels. Figure 2 instantiates the structure in the year's emblematic exploit. Table 2 defines the six classes. Two of them (A1, A2) predate the protocol era — they are inherited from the security of LLM-integrated applications generally [9]; the remaining four (A3, A4, A5, A6) arise with the 2024-2025 tool-protocol ecosystem, and A5 is that ecosystem's composite expression.

A1 — Direct prompt injection. The attacker addresses the agent itself (a chat channel) and attempts to override its instructions. Trust premise: the conversation channel is between the operator and the agent. Direct injection is the oldest class and the most defended; frontier models' instruction-hierarchy training has substantially raised its cost. It remains a perimeter concern, not the core 2025 problem.

A2 — Indirect prompt injection via content. The attacker embeds instructions in content the agent will *process on the user's behalf* — an issue, a web page, an email, a document. Trust premise: content is data. This is the class that breaks the data/executive boundary, and it is the enabling mechanism of the GitHub toxic flow [3]. No 2025 mitigation eliminates it; the design consequence is to assume injection will eventually succeed and control what the agent can then *do*.

A3 — Tool poisoning via metadata. Instructions hidden in tool names, descriptions, or parameter schemas — the channel between a server and the agent that selects tools. Trust premise: tool metadata is operator-controlled plumbing. The April 2025 disclosure showed poisoning surviving into agent context and overriding instructions from other servers [2].

A4 — Rug pulls: late tool redefinition. A tool that is benign when reviewed and redefined afterward — the temporal variant of A3, exploiting the gap between audit time and execution time. Trust premise: reviewed artifacts stay reviewed. Speculative-tool definitions that re-resolve at call time are the mechanism.

A5 — Cross-server toxic flows. The composite class: an injection planted in one integration steers the agent into abusing *another* integration's privileges — the GitHub exploit's essence (public-issue content → private-repo tooling) [3], and the reason MCP's single-server security analysis is insufficient [4]. Trust premise: per-integration security reviews compose. They do not.

A6 — Token and identity confusion. Delegation-channel abuse: an agent holding an access token scoped for one resource uses (or leaks) it against another; OAuth flows without resource indicators permit exactly this [1][10]. The June 2025 MCP revision's mandatory RFC 8707 support is the protocol-layer response.

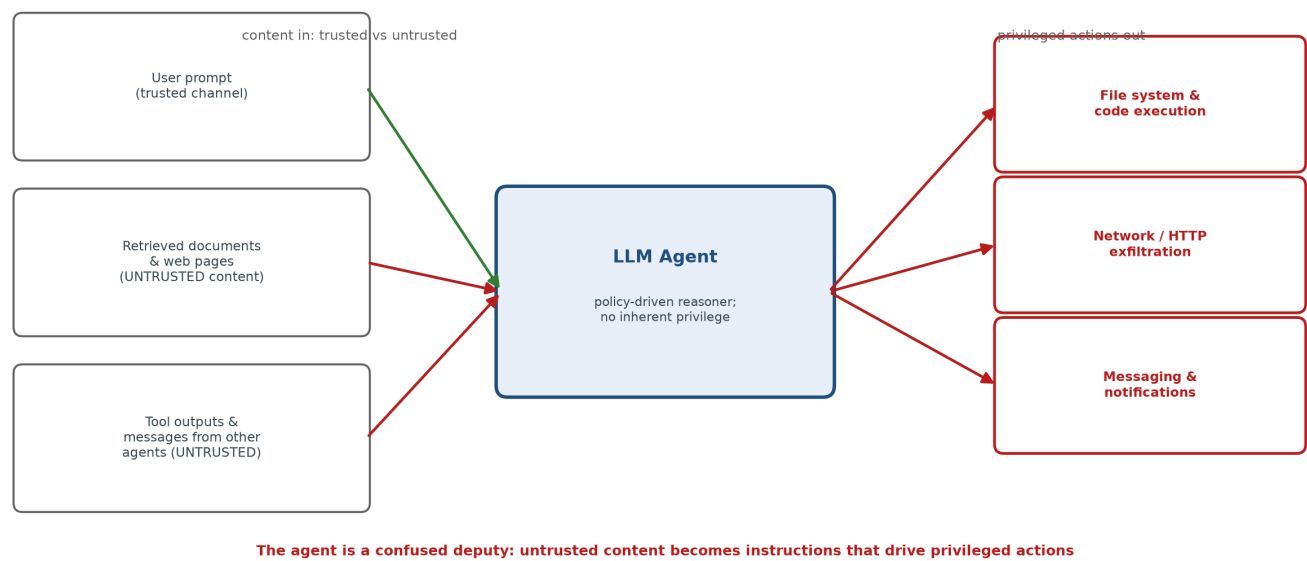


Figure 1. The agentic trust boundary: untrusted content channels (left) drive privileged action channels (right) through the language model. The model is the confused deputy; every 2025 exploit instance is a path through this structure.

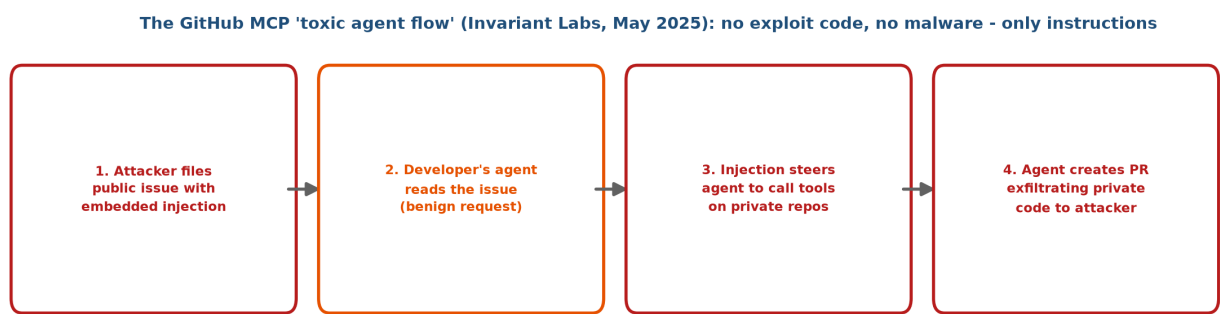


Figure 2. The GitHub MCP toxic agent flow (May 2025) [3]: an indirect prompt injection in attacker-controlled content steering a privileged agent across integration boundaries — the canonical A2 – A5 composite.

Protocol / instrument	2025 security-relevant development	Date	Source
MCP	Specification revision 2025-06-18: MCP servers classified as OAuth 2.1 resource servers; RFC 8707 resource indicators required; JSON-RPC batching removed	Jun 18, 2025	[1]
MCP	Prior revision 2025-03-26 (baseline for the June changes)	Mar 26, 2025	[1]

A2A	Protocol created by Google; donated to the Linux Foundation with AWS, Cisco and >100 supporting companies	Jun 23, 2025	[5]
EU AI Act	General-purpose AI model obligations become applicable	Aug 2, 2025	[6]

Table 1. Security-relevant evolution of agent-interoperability protocols in 2025 [1][5].

Class	Mechanism	Exploited trust premise	Documented instance
A1 Direct prompt injection	Instructions addressed to the agent itself override system behavior	Conversation channel is operator-to-agent	Widely studied; largely mitigated by instruction-hierarchy training [9][12]
A2 Indirect injection via content	Instructions embedded in content processed on the user's behalf	Content is data, not code	GitHub MCP toxic flow (May 2025) [3]
A3 Tool poisoning via metadata	Instructions in tool descriptions/schemas shape agent behavior across sessions	Tool metadata is operator-controlled plumbing	Invariant disclosure (Apr 2025) [2]
A4 Rug pull (late redefinition)	Benign-at-review tool redefined before or at execution	Reviewed artifacts stay reviewed	Temporal variant of A3 [2]
A5 Cross-server toxic flows	Injection in one integration steers abuse of another integration's privileges	Per-integration reviews compose	GitHub MCP exploit chain (May 2025) [3][4]
A6 Token & identity confusion	Access tokens or agent identities used across resource boundaries	Delegation is scoped and verified	Motivated the RFC 8707 mandate in MCP 2025-06-18 [1][10]

Table 2. The agentic-AI attack taxonomy (2024-2025 disclosures) [2][3][4].

5. Mitigation Coverage (RQ3)

Figure 3 and Table 3 present the assessment. Three structural findings.

Blast-radius reduction works; prevention does not. Capability-scoped tools (least privilege per agent, per task), OAuth resource-server semantics with RFC 8707 resource indicators [1][10], and confirmation gates for irreversible actions all score partial-to-strong against the action-channel classes (A5, A6) because they bound *what an exploited agent can do*. None of them prevents injection itself — against A2 the entire mitigation column is partial at best, and the strongest honest entry is detection.

The trifecta is a separation-of-duties rule. Each trifecta factor is individually benign and ubiquitous: agents need private data, must read untrusted content, and must communicate externally. The engineering rule the trifecta implies is *never co-locate all three in one agent without an interposed control*: data-reading agents should not hold outbound communication capabilities; content-processing agents should not hold write or exfiltration capabilities; and where the combination is unavoidable, a human confirmation gate must interpose. This converts a descriptive vulnerability statement into a reviewable design property.

Detection is the compensating layer. Toxic-flow analyzers — automated discovery of injection-to-privilege paths across tool graphs — generalize from the Invariant disclosures [2][3] and score as the main partial cover for A5, with red-team analyzers as the main cover for A4 (rug pulls) since they audit at execution time, not review time.

	Capability-scoped tools (least privilege)	OAuth resource-server semantics (RFC 8707)	Human confirmation gates	Provenance labeling & flow analysis	Session isolation per agent/task	Red-team analyzers & toxic-flow detection
A1 Direct prompt injection	+	~	+	~	~	~
A2 Indirect injection via content	-	~	~	+	~	~
A3 Tool poisoning (metadata)	~	+	~	+	~	~
A4 Rug-pull tool redefinition	-	-	~	~	~	+
A5 Cross-server toxic flows	~	~	~	+	+	~
A6 Token & identity confusion	~	+	~	-	+	-
	+ strong		+ partial / conditional	no meaningful coverage		

Mitigation class	Mechanism	Strong against	Residual gap
Capability-scoped tools	Minimum tool set per task/session; deny-by-default	A5, A6 (blast radius)	Does not prevent injection (A2/A3)
OAuth resource-server semantics + RFC 8707	Tokens bound to specific resources; audience enforcement	A6	Scope negotiation UX; legacy servers
Human confirmation gates	Irreversible actions require auditable human approval	A2/A5 consequences	Fatigue; approval-of-what-you-did-not-read
Provenance labeling & flow analysis	Content provenance tracked into agent context; injection-path detection	A2, A5 (detection)	Fidelity of provenance through summarization
Session isolation per agent/task	Agents do not share context or credentials across tasks	A5	Coordination cost; multi-agent workflows need shared state
Red-team analyzers & toxic-flow detection	Automated discovery of injection-to-privilege paths	A4, A5 (detection)	Adversarial adaptation; coverage of novel flows

6. Engineering Guidance for MCP/A2A Adopters

7. Limitations and Threats to Validity

16

Frozen at the end of 2025; the protocol specifications, the model layer, and the governance instruments (EU AI Act phase-in) are all moving, and several mitigations assessed here as partial were under active specification development at year-end.

8. Conclusion

Agentic AI's 2025 security record is unusually legible: an industrial protocol layer (MCP, A2A), a concrete and reproducible exploit class (toxic flows), a generalizable condition (the lethal trifecta), and a protocol-layer response (OAuth resource-server semantics). The confused-deputy structure explains all of it. The engineering consequence is not a defense that removes the risk — none exists in 2025 — but a discipline that makes exploitation survivable: capability scoping that bounds blast radius, delegation binding that keeps identity meaningful, confirmation gates on the irreversible, provenance instrumentation for detection, and design reviews that treat the lethal-trifecta co-location as a defect in itself. Agents are becoming infrastructure; their security posture in 2026 will be decided by whether these boundary disciplines ship with the same speed the protocols did.

Declarations

Funding. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

CRedit contribution statement. Mianxiong Dong: conceptualization, threat-model development, writing – original draft. M. F. Mridha: investigation, protocol-record verification, writing – review & editing. Muhammad Waqas: validation, mitigation analysis, writing – review & editing. (Corresponding author: Mianxiong Dong.)

Competing interests. The authors declare no competing interests.

Data and code availability. No new datasets were generated; all claims trace to the cited primary disclosures and specifications [1][2][3][4][5].

Ethics and dual-use. The paper describes publicly disclosed vulnerabilities in a defensive frame, contains no working exploit content, and follows the coordinated-disclosure record already published by the original researchers.

AI assistance declaration. AI-based tools were used in a limited supporting capacity (literature search, figure preparation, and language editing); all research content, analysis, and conclusions were produced, reviewed, and verified by the authors, who take full responsibility for the work.

References

- [1] Anthropic and the Model Context Protocol community, "Model Context Protocol Specification, revision 2025-06-18," Jun. 18, 2025. [Online]. Available: <https://modelcontextprotocol.io/specification/2025-06-18>
- [2] Invariant Labs, "MCP Security Notification: Tool Poisoning Attacks," Apr. 1, 2025. [Online]. Available: <https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks>
- [3] Invariant Labs, "GitHub MCP Exploited: Accessing private repositories via MCP," May 26, 2025. [Online]. Available: <https://invariantlabs.ai/blog/mcp-github-vulnerability>
- [4] S. Willison, "The lethal trifecta for AI agents: private data, untrusted content, and external communication," simonwillison.net, Jun. 16, 2025. [Online]. Available: <https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/>
- [5] The Linux Foundation, "Linux Foundation Launches the Agent2Agent Protocol Project to Enable Secure, Intelligent Communication Between AI Agents," Jun. 23, 2025. [Online]. Available: <https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents>
- [6] Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act), OJ L, 12 Jul. 2024 (general-purpose AI obligations applicable from 2 Aug. 2025).
- [7] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023. doi: 10.6028/NIST.AI.100-1.
- [8] NIST, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," NIST AI 600-1, Jul. 2024. doi: 10.6028/NIST.AI.600-1.
- [9] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), 2023. [Online]. Available: <https://arxiv.org/abs/2302.12173>
- [10] D. Hardt and B. Campbell, "Resource Indicators for OAuth 2.0," IETF RFC 8707, Apr. 2020. doi: 10.17487/RFC8707.
- [11] A. Parecki, T. Lodderstedt, E. Sachs, and A. Dey, "OAuth 2.0 Authorization Framework: OAuth 2.1" (draft-ietf-oauth-v2-1), IETF Internet-Draft, 2025. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/>
- [12] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications," GenAI Security Project, 2025. [Online]. Available: <https://genai.owasp.org/>