What is agentic AI security?

Agentic AI is the term for AI features that decide which tool to use and use it, in a loop. Once an AI can call tools, the security boundary you used to draw around the chatbot now wraps every tool it can reach. Industry research at USENIX Security 2025 classified seven distinct failure patterns for these systems. Most production AI features built since mid-2024 fall in this category.

HOW IT WORKS

01 How does the attack surface grow with tool access?

Three things change at once.

Output becomes action. A prompt-injection payload that previously made the model say something embarrassing can now make it call `send_email` with attacker-chosen arguments, or `delete_record(*)` against a customer table. The cost of a successful injection goes from reputational to operational.

Retrieval becomes RCE. Every tool that returns content the agent reads is an indirect prompt-injection channel. A search tool that returns attacker-controlled snippets, an inbox tool that returns user-supplied emails, a database tool that returns notes a customer wrote: each is a path from untrusted content into the agent's instruction stream.

Chains multiply impact. A single injection can trigger Tool A which produces output that biases Tool B's call which surfaces data that influences Tool C. Researchers have demonstrated chains that exfiltrate data through three or four tool hops before producing user-visible output (Greshake et al., 2023; USENIX Cloak/Honey-Trap 2025).

> COST-OF-FAILURE SHIFTS
>
> *A chatbot that gets jailbroken costs you brand damage. An agentic system that gets jailbroken costs you whatever its tools can do.*

02 What are the most exploited agentic patterns?

From the AI engagements we have run over the past year, the recurring failure modes:

HOW TO DEFEND

- Least-privilege tools. Each tool exposes the narrowest action that solves the use case. A `send_email` tool restricted to a pre-approved address list is dramatically safer than one that accepts an arbitrary `to:` field.
- Deterministic checks before high-impact actions. Sending money, deleting records, granting access: these gate on a deterministic policy check or a second LLM call that has not seen the user input.
- Provenance tagging through the loop. Tag every piece of text in the prompt with its source (system, operator, user, tool result), and surface that tag to downstream defenses.
- Action audit logging. Every tool call, with its arguments and the trigger context, in a structured log you can replay.
- Behavioral monitoring. Alert on tool-call shapes you have never seen, on outputs containing exfil markers (base64 in URLs, long hex strings, unusual locale tokens), on action sequences that diverge from the trained distribution.
- Scope honesty. Not every workflow needs an agent. A deterministic pipeline with one LLM call at a fixed stage is almost always safer than a freely-iterating agent loop. Choose the architecture that does not need to refuse much.

SOURCES

- Email-summarizer with reply capability. Reads inbox content (indirect-injection channel) and can compose outbound mail (action). Almost every implementation we test ships with at least one path that the agent will follow without confirmation.
- Customer-support agent with ticket-write access. Customer message arrives, agent reads it, agent updates ticket state, agent emails the customer. Indirect injection from the customer message can drive every downstream step.
- RAG-based code reviewer with PR comment. Agent reads diff and codebase context (instruction channel from any file in the repo), posts comments (action). Useful for poisoning the review of a PR you control.
- Multi-agent research with execute tool. One agent plans, another searches, another executes code. The execute-agent trusts the search-agent's output. Trust chain has no real audit.
- MCP-connected assistants. A single MCP server exposes a database, a filesystem, and a network tool. Compromising the agent compromises all three.

03 What does the USENIX 2025 Cloak/Honey-Trap taxonomy classify?

Ben-Gurion University researchers published a structured taxonomy of LLM-agent attacks at USENIX Security 2025. The numbers worth memorizing:

- 7 vulnerability classes across the agent loop: prompt injection, tool-misuse, memory poisoning, action chaining, output spoofing, multi-agent betrayal, and authorization confusion.
- 6 attacker strategies ranging from one-shot direct injection to slow-drip context poisoning across long conversations.
- 15 attack techniques that combine the above into concrete exploit recipes.
- CHeaT testbed released alongside the paper, reproducing each technique against open-source agent frameworks.

If you are designing a defensive architecture for an agentic system, this paper is the reading-list

[1] OWASP LLM06:2025, Excessive Agency
[2] Cloak and Honey Trap (USENIX Security '25)
[3] Greshake et al., More Than You've Asked For
[4] MITRE ATLAS

anchor. We use it as a coverage checklist when scoping engagements.

04 How does SecureLayer7 test agentic systems?

We start by enumerating the agent loop, not the chat surface. The questions we answer before sending a single payload:

- What tools is the agent allowed to call?
- What arguments can each tool accept, and how are they validated?
- What does each tool read from, and which of those sources are attacker-reachable?
- What does each tool write to, and what is the blast radius of a wrong call?
- Where are the authorization checks, and do they sit before or after the tool call?

Then we plant payloads at each attacker-reachable source and trace what happens. The success criterion is not 'did the model say something it should not' but 'did the system perform an action it should not, reach a resource outside scope, or expose data across a trust boundary.'

Every confirmed finding ships with the planted payload, the tool-call sequence it triggered, the trust boundary it crossed, and an architectural recommendation that names the structural fix, not a filter rule.

Test your agent loop, not just the chatbot.
securelayer7.net/learn/ai-security/agentic-ai-security

Open online