SALT

SELF-ASSESSMENT

The agentic path readiness checklist

For AppSec, Product Security and API Security teams

Your agent's prompt can pass every check while the APIs behind it stay wide open, because every consequential thing an agent does is an API call. This checklist helps you find where your program stops short of the full agentic path, from prompt to MCP server to API to data.

The code works. That's the problem.

The gap between functional code and secure code is bigger than you think.

99%
of AI-generated code is syntactically correct

44%
of AI code generation tasks introduce a known security flaw

90%
of 200 audited, publicly deployed vibe-coded apps had at least one vulnerability

Sources: Veracode, 2026 GenAI Code Security Report; Deng, Fan & Meng, Understanding the (In)Security of Vibe-Coded Applications, 2026.

How to use it: tick an item only if you can demonstrate it today, not if it is planned. Eighteen items across five themes, about ten minutes. Count your unticked boxes at the end.

01 Discover: do you know what exists?

You cannot govern or protect agents, MCP servers and LLM calls you have not found.

- [ ] I can list every AI agent, MCP server and LLM integration running in production, with a named owner for each.
- [ ] I can find shadow MCP servers and LLM usage outside our approved AI gateway without waiting for teams to self-report.
- [ ] My API inventory includes shadow, zombie and internal-only APIs, and shows which of them agents and MCP tools can reach.
- [ ] I can see my external exposure the way an attacker does, including public agents, MCP servers and APIs I did not know were reachable.

02 Govern: do you control what agents can reach?

A uniquely identifiable agent with excessive privilege can still cause high-impact damage. Only 26.1% of organizations give agents dedicated identities with scoped permissions.

- [ ] Every production agent has a dedicated non-human identity with least-privilege permissions, not a shared service account, human credential or embedded API key.
- [ ] I know which sensitive data (PII, financial records) each agent, MCP tool and downstream API can touch.

- [ ] I can see posture gaps across the whole path in one view: excessive tool access, missing authentication, no rate limiting, hard-coded keys in MCP server code.
- [ ] I can see AI guardrail and gateway configurations across clouds, endpoints and managed AI services, and spot the agents that no guardrail covers.

03 Shift left: are you catching issues before release?

AI coding assistants ship code that works but is not always secure. Only 6.8% of teams resolve security issues in the assistant at generation time, and only 22.2% enforce policy across all approved tools.

- [ ] Our security and compliance policies are enforced inside AI coding assistants at generation time, across every approved tool, not just one.
- [ ] Agent-facing APIs and MCP servers are checked for authentication, authorization and data exposure flaws before they ship.
- [ ] AI-generated code and human-written code go through the same API security gates, with findings routed to the team that owns the fix.

04 Detect: can you see attacks at every layer?

An attack can start at the prompt or skip straight to a backend API. Runtime protection has to cover both the prompt layer and the action layer.

- [ ] I detect direct and indirect prompt injection, jailbreak attempts, unsafe model behavior and sensitive data exposure in LLM interactions in real time.
- [ ] I can see prompts at the agent trigger (the chat or agent API), not only at an AI gateway, so coverage does not depend on routing everything through one.
- [ ] I protect MCP servers, APIs and microservices directly, in case an attacker ignores the prompt and attacks the API behind a tool.
- [ ] I would catch a novel attack by its effect: an agent calling APIs or touching data in a way it never has before.

05 Respond: can you trace and stop it at machine speed?

When the actor is software, hours is a long time. Only 15.7% could reconstruct what a specific agent did yesterday in minutes, and 12% can stop a misbehaving agent only by taking the application down.

- [ ] I can reconstruct everything an agent did yesterday in minutes, from the initial prompt through the MCP server to the API calls and data touched.
- [ ] I can connect a prompt injection to its downstream MCP and API activity as one attack campaign, and see the blast radius.
- [ ] I can block a malicious prompt or response inline, and stop a misbehaving agent automatically, without taking the application down.

Score yourself

Count the boxes you could not tick, out of 18.

UNTICKED	WHERE YOU STAND	START WITH
0 to 3	Strong coverage of the agentic path	Close the remaining gaps and test your response speed
4 to 9	Partial coverage, with blind spots between layers	The theme with the most unticked boxes, Discover and Govern first
10 to 18	Exposed: agents are likely acting beyond what you can see or stop	Discover, then add runtime detection on the layers attackers reach first

Most unticked boxes in Discover and Govern means you are protecting what you know about. Most in Detect and Respond means you can see the estate but not stop an attack in progress.

How the Salt Agentic Security Platform covers each theme

Salt secures the full agentic path in one platform: Agentic Security Posture Management (AG-SPM) to discover and govern, and Agentic Detection and Response (AG-DR), now including native AI Detection and Response (AI-DR), to protect at runtime.

THEME	WHAT SALT DOES	WHERE IT LIVES
Discover	Finds agents, MCP servers, APIs and shadow LLM usage from live traffic, cloud and code, with an outside-in view of what is exposed	AG-SPM, Salt Surface, Salt Connect
Govern	Maps agents to their models, MCP servers, tools and downstream APIs in the Agentic Security Graph, and shows guardrail and gateway coverage gaps	AG-SPM, Agentic Security Graph
Shift left	Enforces security policy inside AI coding assistants before code reaches the repository	Salt Code
Detect	Catches prompt injection, jailbreaks and unsafe model behavior, plus attacks on MCP servers, APIs and microservices, with or without an AI gateway	AG-DR with native AI-DR, Salt Protect
Respond	Correlates a prompt attack to its downstream MCP and API activity as one campaign, and responds out of band or inline	AG-DR, Agentic Security Graph

See your own score in minutes

The fastest way to fill in this checklist is to look. Salt runs a non-intrusive external scan that shows the agents, MCP servers and APIs an attacker can see, and most teams find something they did not know was exposed. Connecting a cloud account takes about 5 to 10 minutes and adds the internal picture.

Get your free attack surface assessment

Or request a demo at salt.security/demo-request

SALT