SALT

TECHNICAL BRIEF

Building the Agentic Security Graph

How Salt Security combines multiple data sources to deliver complete visibility across the agentic stack

Table of Contents

01 Executive Summary

Securing agentic infrastructure requires more than monitoring one layer. Agents connect to MCP servers. MCP servers expose APIs. APIs call downstream services. Each layer generates a different kind of data: external exposure, configuration posture, code-level risk, and live runtime behavior.

No single integration point captures all of it. The Agentic Security Graph is Salt Security's approach to solving this: a unified security context built by combining four distinct data sources, each agentless or with minimal deployment, into one correlated view of your agentic environment.

The Core Insight:
Most security tools see one layer. Salt sees all four: the internet-facing exposure of your agentic stack, the configuration and posture of your infrastructure, the API code and the live behavior of agents and APIs at runtime. The Agentic Security Graph is what you get when you combine all four.

02 The Problem: Fragmented Visibility Across the Agentic Stack

Modern agentic infrastructure is distributed by design. An AI agent running in one environment may call MCP servers hosted elsewhere, which in turn invoke APIs spanning internal, external, and third-party systems. Understanding the security posture of this infrastructure requires answering four distinct questions simultaneously:

- What does my agentic stack look like from the outside? What is exposed to the internet?
- What is configured in my environment, and where are the posture gaps?
- What does my code reveal about my MCPs, API connections, and security risks?
- What is actually happening at runtime between agents, MCPs, and APIs?

Answering any one of these questions in isolation produces an incomplete picture. A posture gap that looks low-risk in isolation may become critical when correlated with live traffic showing active exploitation. An exposed API found in an internet scan takes on greater urgency when code analysis reveals it carries sensitive data.

The Agentic Security Graph brings these four data sources together into a single, correlated security context.

03 How the Agentic Security Graph Is Built

Salt Security collects data from four sources, each targeting a distinct layer of the agentic stack. Together, they feed the Agentic Security Graph.

Component	Data Method	What It Does	Output
Salt Surface	External Internet Scan (Zero deployment)	Points to a domain and performs an internet scan using multiple methodologies to find all exposed APIs, MCP servers, and Agent endpoints.	External attack surface map of your agentic stack before any integration is required.
Salt Connect	Agentless Configuration Integration (Agentless)	Integrates without agents into cloud providers, API gateways, databases, and tools that host MCPs, agents, or APIs.	Configuration-level inventory and posture gap analysis across the full ecosystem.
Salt Code	Code Repository Scan (Agentless (read-only))	Connects with simple read-only access to code repositories and scans to find all MCPs, connected APIs, and security risks at the code level.	Pre-production visibility into agentic infrastructure and code-level security risks.
Salt Collect	Live Runtime Traffic Monitoring (Runtime sensor)	Uses context from Salt Connect to activate live traffic monitoring across 70+ technologies: Kubernetes, load balancers, MuleSoft, F5, legacy and modern stacks. Monitors API, MCP, A2A, and agent traffic.	Real-time behavioral data on how agents, MCPs, and APIs interact at runtime.

Salt Surface: Start With Zero Deployment

The first input to the Agentic Security Graph requires no integration at all. Salt Surface performs an internet scan against a domain, using multiple discovery methodologies to find every exposed API, MCP server, and Agent endpoints visible from the outside.

This answers the attacker's first question: what can I see? Before any configuration data or runtime data is collected, Salt can show an organization exactly how its agentic stack appears to an external threat actor. For many organizations, this reveals exposure they did not know existed.

Salt Connect: Configuration Without Complexity

Salt Connect integrates agentlessly with the tools that make up the agentic stack: cloud providers, API and AI gateways, databases, and the platforms that host MCPs and agents. No agents are deployed. No traffic is intercepted at this stage.

The output is a configuration-level inventory of the environment combined with a posture analysis. Salt Connect identifies where security standards are not being met, where MCPs are misconfigured, and where APIs are exposed in ways that create risk. This configuration context also serves as the foundation for Salt Collect, informing where to activate live traffic monitoring.

Salt Code: Pre-Production Risk from the Repository

Salt Code connects with read-only access to code repositories and scans for MCP definitions, API connections, and security risks embedded in the codebase. This is an agentless operation: Salt reads, never writes.

Code-level analysis gives Salt visibility into the agentic stack before it reaches production. It surfaces which MCPs are defined, which APIs they call, what data they handle, and where developers have introduced patterns that create downstream security risk. This layer is particularly important because agentic infrastructure is often defined in code before it is ever deployed.

Salt Collect: Runtime Visibility at Scale

Salt Collect activates live traffic monitoring across 70+ technologies: Kubernetes, load balancers, MuleSoft, F5, and both legacy and modern stack components. Salt has spent eight years building expertise on how to monitor API traffic, and that capability now extends to MCP traffic, agent-to-agent (A2A) traffic, and the full range of interactions within an agentic environment.

Runtime data is the layer where behavioral anomalies surface. It is where an agent calling an API it has never called before becomes visible. It is where a compromised MCP server begins to exhibit unexpected patterns. Without runtime visibility, detection depends entirely on static analysis of configuration and code: a posture snapshot rather than a live security feed.

Salt Collect uses the inventory and context built by Salt Connect to ensure that monitoring is targeted and comprehensive, covering the components that matter rather than generating undifferentiated traffic noise.

04 The Agentic Security Graph: Combined Context

The output of these four data sources is the Agentic Security Graph: a correlated security model that shows how agents connect to MCP servers, how MCP servers connect to tools and tools to APIs, and how to prioritize risk based on full context rather than isolated signals.

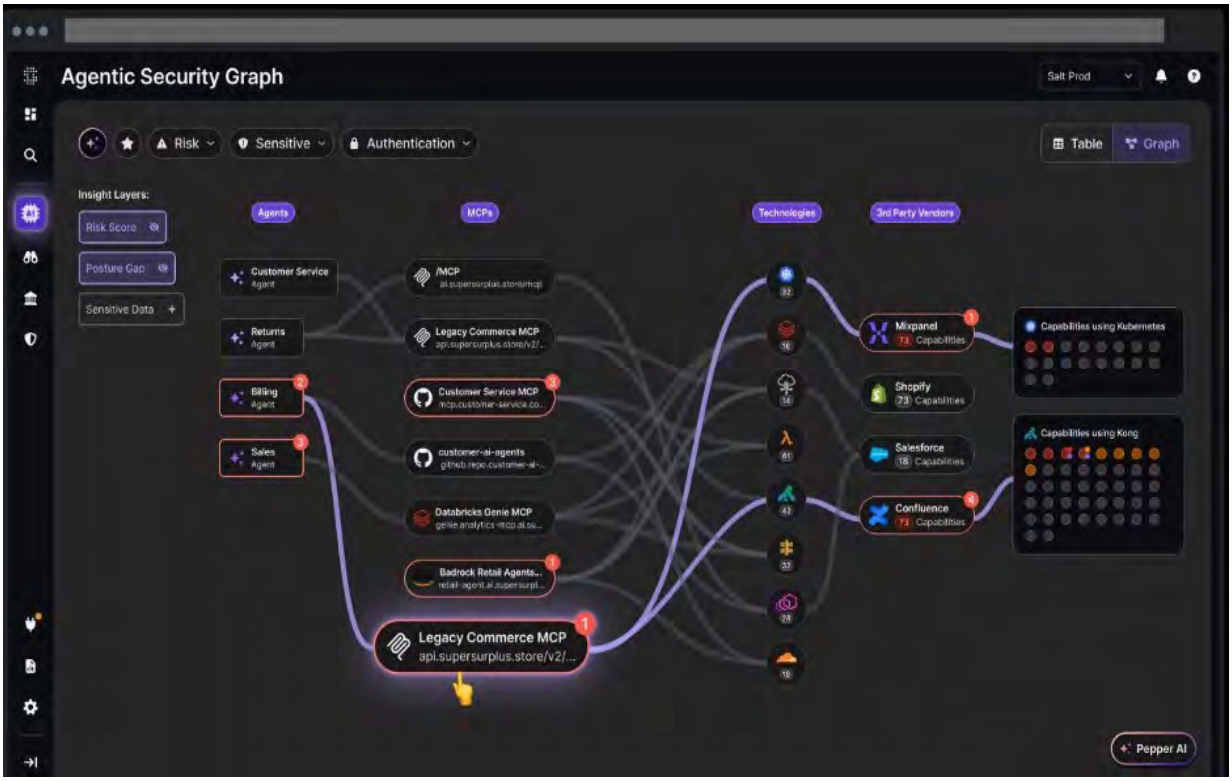

Figure 1: The Agentic Security Graph in Salt Security, showing agents, MCPs,

What the Agentic Security Graph delivers:
Context from outside (internet-facing exposure) combined with configuration data from across the ecosystem combined with runtime data feeds into a single security graph. The result: accurate inventory, prioritized risk, and detection that understands the full chain of agentic activity rather than individual events in isolation.

This matters because agentic security risk is often a chain, not a single event. An exposed MCP server, a misconfigured API gateway, and an anomalous runtime behavior may each appear low-severity in isolation. Correlated across the Agentic Security Graph, they may represent a critical attack path.

The graph makes this correlation possible. It provides the foundation for the three core security outcomes Salt delivers across the agentic stack:

- **Reduce attack surface and create a unified AI inventory of LLMs, agents, MCP servers, MCP tool APIs, downstream APIs, and rogue or zombie endpoints:** Discover
- **Embed security standards across the SDLC and extend data security to MCP servers and APIs:** Govern
- **Stop behavioral attacks, lock down agents and MCPs with guardrails, and detect anomalies at runtime:** Protect

05 Why This Approach Is Different

Most security tools are built around a single integration point: a traffic mirror, an agent, or an API gateway connector. Each of these captures one layer of the agentic stack, and each has blind spots.

A WAF sees web traffic. It does not see MCP configuration, code-level API definitions, or the behavioral patterns of agents calling APIs at runtime. A static analysis tool sees code. It does not see live traffic or runtime deviation from expected behavior.

Salt's four-source architecture is designed around the reality that agentic infrastructure does not live in one place. The Agentic Security Graph is the result of eight years of API security expertise applied to a new problem: what does complete visibility look like when the infrastructure itself is intelligent, distributed, and acting autonomously?

Take the Next Step:

Do not rely on inadequate tools to protect your API and AI infrastructure.
See the Salt difference for yourself.

Get a demo