SALT

SECURING THE NEXT GENERATION OF AI SYSTEMS

A CTO’s Guide to Protecting LLMs, AI Agents, and MCP Servers

SALT

Table of contents

SALT

1. Executive Summary

The rise of large language models (LLMs), autonomous AI agents, and Model Context Protocol (MCP) servers represents one of the most transformative shifts in software architecture since the advent of cloud computing. But it also introduces a new and largely misunderstood security paradigm.

Unlike traditional applications, AI agents don't just respond to inputs. They reason, remember, and act autonomously across systems. MCP servers serve as the connective tissue that defines what agents know, how they think, and what they're allowed to do. And behind it all is an API fabric that is growing more dynamic, decentralized, and difficult to govern.

This white paper explores what makes these components unique, why existing security tools fall short, and how Salt Security provides a modern security framework to protect them.

2. Understanding the New Architecture of AI

2.1 What Is a Large Language Model (LLM)?

An LLM is a machine learning model trained on vast datasets of human language to predict and generate coherent text. Examples include OpenAI's GPT, Meta's LLaMA, and Anthropic's Claude. These models can:

- Summarize and synthesize content
- Answer questions with contextual reasoning
- Write code, generate content, or assist in decision-making

While powerful, they are inherently unpredictable and prone to "hallucinations" or unintentional disclosure of learned context. Even worse, they may retain long-term memory, often persisting sensitive data within embeddings or retrieval chains.

2.2 What Are Agentic AI Systems?

Agentic AI refers to autonomous systems that go beyond passive response to take real-time actions. These agents:

- Set and pursue goals
- Maintain memory across sessions
- Trigger workflows and API calls on their own
- Communicate with other agents or systems via coordination protocols

Think of them as autonomous workers inside your environment, making decisions on behalf of users or systems. They are not bound by linear, human-paced processes. They operate continuously and adaptively.

2.3 What Is the Model Context Protocol (MCP)?

MCP is the emerging standard for how agents and LLMs interact. It defines:

- The agent's memory, tool use, and capabilities
- What data is accessible to the agent
- How it communicates with other agents, APIs, and services

MCP servers act like routers and brokers for agent intelligence. They orchestrate the flow of context, instructions, and capabilities between components. As such, they represent a new class of critical infrastructure.

3. Why This Changes the Security Equation

3.1 Traditional AppSec and API Security Don't Cover AI Behavior

Legacy security tools (WAFs, API gateways, EDR, and even cloud-native controls) are reactive. They inspect payloads, check authentication, and monitor access at the edge.

But agentic AI systems don't follow a traditional request-response model. They:

- Invoke APIs autonomously, not via user sessions
- Maintain memory that can include sensitive or regulated data
- Operate across internal, partner, and third-party systems

- Adapt behavior based on feedback loops

You can't block this risk with signatures or simple allowlists. You need contextual, behavioral, and persistent visibility.

3.2 The Risk Surface Has Grown Rapidly

Agentic AI introduces a new category of risks:

- Prompt Injection and Cross-Agent Contamination
  Malicious inputs can be embedded into agent prompts, leading to downstream compromise or unauthorized actions.
- Memory-Based Data Leakage
  Sensitive information retained in long-term memory or vector databases can be accessed unexpectedly and by different agents or over time.
- MCP Exploitation
  If an attacker gains access to an MCP server or manipulates the context registry, they can redefine what an agent knows or what tools it uses.
- API Abuse at Scale
  Agents make high-frequency, asynchronous API calls that may go undetected by tools that aren't designed for dynamic, machine-initiated behavior.

4. A New Security Blueprint: Behavioral API Defense for AI Systems

Salt Security addresses this emerging architecture with a platform that understands and protects the behavior of AI agents, not just the APIs they touch.

4.1 Unified API Fabric Visibility

Salt automatically discovers all APIs (internal, external, managed, unmanaged) across your environments. This is crucial for AI systems where agents may use hidden or undocumented APIs via the MCP server or internal tooling.

- No agents or traffic duplication required
- Deployed in minutes via native cloud integrations (e.g. AWS Cloud Connect)
- Real-time and historical view of all API traffic and schema

4.2 Agent Attribution and Behavioral Analysis

Salt enriches API traffic with metadata on who is calling the API:

- Is it a user? An AI agent? A third-party assistant?
- What toolset or memory context was invoked?
- Does the call deviate from learned patterns or violate policy?

This gives teams the ability to identify drift, prevent exfiltration, and detect suspicious activity tied to agents, not just identities.

4.3 MCP Security: Observing the Nerve Center

Salt is building first-to-market capabilities to monitor and secure MCP traffic:

- Understand which agents are registered to which capabilities
- Monitor the flow of memory context and task orchestration
- Detect unauthorized tool use, cross-agent escalation, or shadow agents

MCP servers represent the brainstem of your agentic architecture and Salt is one of the only platforms treating them as first-class security surfaces.

4.4 Sensitive Data Flow and Memory Governance

Salt traces how sensitive data flows across agent interactions:

- Detect when PII or secrets enter memory chains
- Identify retention or retrieval behaviors that violate policy
- Prevent leakage across sessions, agents, or user contexts

This allows security teams to implement practical guardrails and ensure AI memory is governed, auditable, and compliant.

4.5 Customer Story: Securing Healthcare AI Agents

Customer: Global Healthcare Provider
Industry: Healthcare
Use Case: Patient-facing AI agent for intake and scheduling

A major healthcare organization launched an LLM-powered virtual assistant to help schedule appointments, interpret lab results, and answer billing questions for patients.
The assistant integrated with EHR systems, billing APIs, and patient portals. Initially, the development team focused on HIPAA compliance, ensuring that secure endpoints were used and that the LLM was instructed not to retain personal health data.

However, after deployment, the organization noticed that the agent was occasionally surfacing past interactions and summaries that included PHI; data the agent had "remembered" across sessions.

Salt's Solution:
Salt discovered that the agent was calling APIs via an internal MCP server that dynamically assigned toolsets and memory chains. Using Salt's behavioral analytics, the security team:

- Identified that the agent was accessing memory context outside the session scope
- Detected embedded PHI in vector databases used for long-term reasoning
- Enforced new policies to restrict sensitive context access by agent type and use case

Outcome:
The healthcare provider gained full observability into how memory persisted across sessions and APIs. Salt helped them shut down risk paths that could have triggered a HIPAA violation without slowing down AI innovation.

5. Deployment Strategy: Fast, Frictionless, and Dev-Friendly

AI innovation moves fast. Salt is built to keep up.

- No Agents or SDKs: Salt integrates directly into cloud environments with zero friction.
- Built for DevSecOps: We align with CI/CD pipelines and provide actionable findings with developer context.
- Self-Service Setup: Teams can deploy Salt in minutes and start seeing value immediately.

Whether you're experimenting with internal copilots or deploying customer-facing AI agents, Salt gets you protected before problems scale.

5.1 Customer Story: Accelerating Agentic AI with Instant API Insight

Customer: AI-Powered Logistics Startup
Industry: Supply Chain Tech
Use Case: Autonomous agent platform for warehouse management

This logistics company was building a fleet of AI agents to manage warehouse operations, including inventory checks, shipment optimization, and inter-agent coordination using an MCP-based framework.

The team deployed multiple agents running in parallel each with LLM reasoning, memory capabilities, and delegated API access to ERP and IoT platforms. The challenge? The team had no idea what APIs the agents were actually calling or how many undocumented or deprecated endpoints were in play.

Salt's Solution:
Using Salt's agent-aware API discovery, the company was able to:

- Instantly identify all APIs the agents accessed, including shadow and legacy ones
- Attribute API activity to individual agents using Salt's identity enrichment
- Map agent behavior across time to detect unusual or risky sequences

Outcome:
With Salt, the engineering team uncovered over 60 undocumented internal APIs being actively used by agents. They rapidly consolidated, decommissioned unnecessary endpoints, and put proper access controls in place. saving weeks of manual investigation and significantly reducing their attack surface.

6. Conclusion: You Can't Secure AI Without Securing APIs

As AI agents gain autonomy, memory, and access, the risk isn't theoretical, it's architectural.

LLMs are no longer passive generators. Agents are no longer scripts. MCP servers are no longer glue code. They are the nervous system of modern software, and they are powered by APIs.

If you're not securing that API fabric (if you're not tracking behavior, context, and memory) you're leaving the door wide open.

At Salt Security, we believe you can't secure AI without securing APIs. That's why we've built the platform to see, understand, and protect the full lifecycle of agentic AI traffic—before innovation becomes exposure.

Want to explore how Salt can secure your AI architecture?
Visit salt.security or request a custom demo today.