Governing *AI-generated code* from prompt to production

Salt Code enforces your security policies inside every AI coding assistant your developers use, then validates compliance through CI/CD and runtime. Powered by the Salt Posture Governance Engine.

1st

SOLUTION TO ENFORCE POLICIES INSIDE AI CODING ASSISTANTS

10+

AI CODING TOOLS SUPPORTED VIA MCP

6

ENFORCEMENT STAGES FROM PROMPT TO RUNTIME

THE PROBLEM

AI writes code at machine speed. Your security can't keep up.

AI coding assistants are now the primary way software is built. Cursor, GitHub Copilot, Claude, Codex, Windsurf, and a growing list of agentic IDEs are generating APIs, MCP integrations, agent tools, and application logic at a pace no security team can review. None of them are trained on your internal security standards, your industry frameworks, or your regulatory requirements.

The result is a structural compliance gap that grows with every commit. Insecure patterns ship before anyone qualified has the chance to review them. SAST and DAST catch issues too late in the pipeline, where every fix is a rewrite and every rewrite is a delay. Policy enforcement remains manual and inconsistent.

> THE SHIFT
>
> The goal is no longer to review code after it is written. It is to ensure every line of AI-generated code is compliant the moment it is created. Salt Code is the first solution built to do exactly that.

HOW SALT CODE WORKS

Policy enforcement, *end to end*

Salt Code applies policy-driven security across six stages of the development lifecycle, powered by the Salt Posture Governance Engine. It connects your security standards directly to code repositories, AI coding assistants, CI/CD pipelines, and runtime environments, creating a single continuous chain of enforcement from prompt to production.

STAGE	OUTCOME	SALT CODE CAPABILITY	HOW IT WORKS	COVERAGE
01 Unify	**Unified governance from code to runtime**	Unified Policy Model	Establishes one policy model for how agentic systems are built, configured, and validated in production, spanning APIs, MCP integrations, and agents. Creates the single governance baseline that every downstream stage discovers, enforces, and validates against.	NATIVE
02 Discover	**Visibility into every AI integration in your environment**	API and Agent Discovery	Inventories APIs, MCP servers, and AI agent integrations across code repositories and cloud environments. Surfaces shadow integrations and undocumented endpoints that expand your attack surface without notice.	NATIVE
03 Enforce	**Compliant code generated by default**	Real-time Policy Enforcement in AI Coding Assistants	Translates your security policies into rules that guide AI-generated output at the point of creation. Integrates directly with Cursor, GitHub Copilot, Claude, and any assistant that supports MCP server configuration.	NATIVE
04 Govern	**Policy violations stopped before production**	CI/CD Pipeline Validation	Extends policy validation into CI/CD workflows. Blocks violations from reaching production, reduces downstream SAST and DAST noise, and gives security teams a defensible enforcement point in the pipeline.	NATIVE
05 Validate	**Continuous compliance in production**	Runtime Behavioral Monitoring	Monitors behavior across APIs, MCP integrations, and agents in production. Detects policy violations, posture gaps, and anomalous activity. Closes the gap between what was supposed to happen and what is actually happening.	NATIVE
06 Remediate	**Closed-loop improvement over time**	Findings-to-Fix Feedback	Translates runtime findings into actionable fixes and feeds them back into developer workflows and the AI assistants themselves. The baseline quality and security of your AI-generated code rises with every cycle.	NATIVE

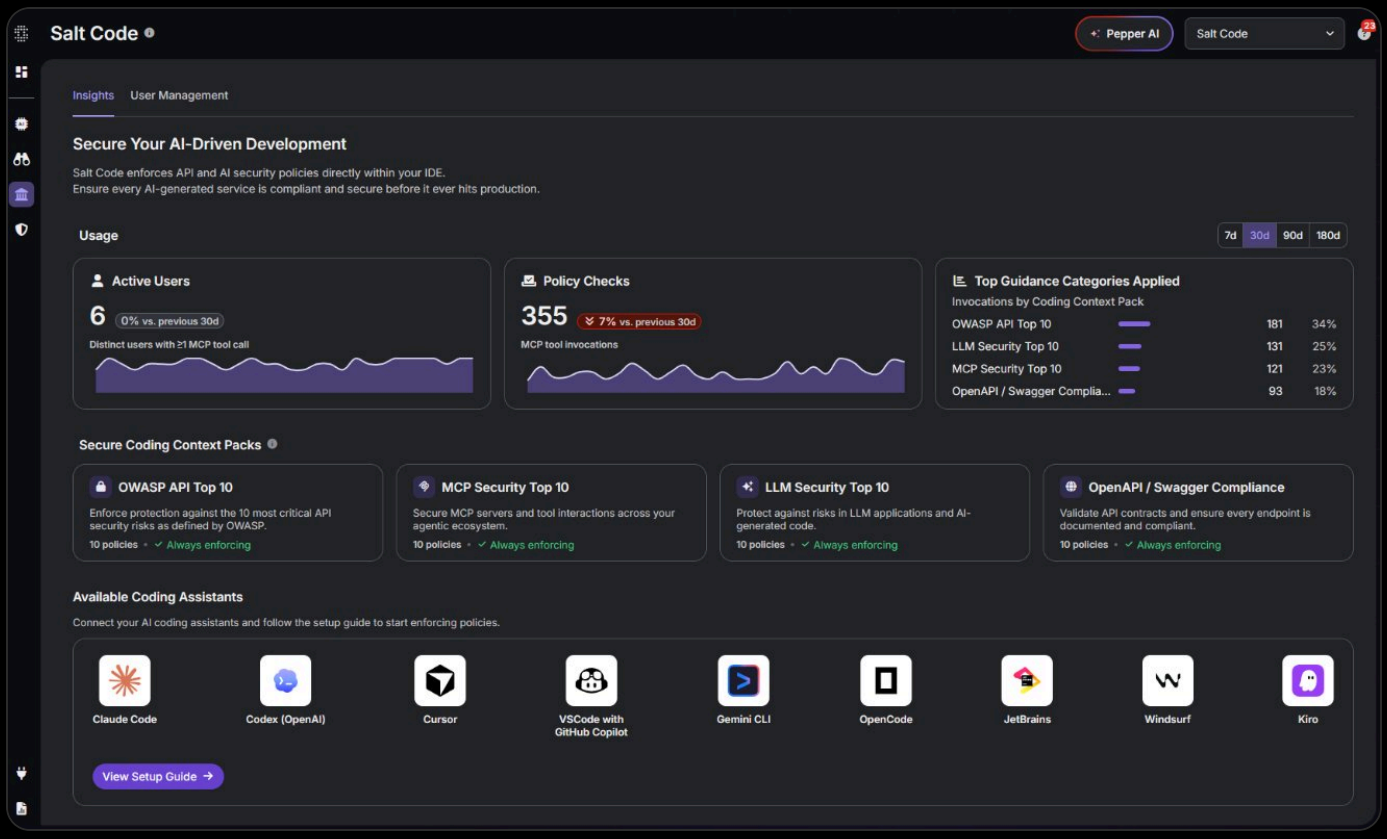
INTEGRATIONS

Built for *every coding assistant*

Salt Code plugs into the tools your developers already use. If it supports MCP, Salt Code governs it.

SUPPORTED CODING ASSISTANTS

- Claude Code
- Cursor
- GitHub Copilot
- Windsurf
- Kiro
- Codex
- Gemini CLI
- Antigravity
- VS Code
- OpenCode
- JetBrains
- Any MCP client



WHY IT MATTERS

What changes when Salt Code is *in your stack*

Salt Code does not bolt onto existing security workflows. It changes where enforcement happens. The downstream consequences touch developer velocity, security team workload, pipeline economics, and compliance posture at the same time.

/ 01

Your policies enforced automatically

Internal security standards, industry best practices, and regulatory requirements applied to every line of AI-generated code without developer effort or security team intervention.

/ 02

Risk prevented at the source

Vulnerabilities never enter your environment. The fastest fix is the one that never had to happen. The cheapest exploit is the one that was never written.

/ 03

Pipeline noise and rework collapse

SAST and DAST findings drop because the underlying issues stopped being created. Developers ship faster. Security reviewers stop drowning in tickets that should never have been opened.

/ 04

Universal AI coding coverage

Any coding assistant that supports MCP server configuration is in scope. Cursor, Copilot, Claude, Codex, Gemini CLI, Windsurf, Kiro, and the next assistant your developers adopt next quarter.

Every week you wait is a week of AI code your policies never touched.

Salt Code can be deployed in days and begins enforcing your policies from the first prompt forward.

Request a Demo →