CSAI Foundation

Cloud Security Alliance AI Safety Initiative

AI Coding Assistants as Attack Surface: Code, Skills, and Secrets

Prompt Injection, Supply Chain Risks, and the ClawHub Ecosystem

Unofficial AI-assisted Research

2026-04-03

Key Takeaways

- Over 30 vulnerabilities across ten major AI-integrated development environments were disclosed in the IDEsaster research campaign (December 2025), with 100% of the ten tested tools found vulnerable to prompt injection leading to code execution or data exfiltration [1] [19].
- The ClawHub skills marketplace — the largest public skills registry associated with OpenClaw and Claude Code workflows — experienced a coordinated malicious skills campaign (ClawHavoc) in which Koi Security's initial audit identified 341 confirmed malicious entries, 335 from a single coordinated campaign [2]; subsequent analysis by Snyk, CrowdStrike, and Antiy CERT raised the total confirmed count to at least 824 [10][20]. These skills delivered commodity infostealers and reverse shells to developer workstations.
- Secrets exposed through AI-assisted development are growing at an accelerating pace: the GitGuardian State of Secrets Sprawl 2026 report found that AI-assisted commits leak secrets at more than double the baseline rate, and over 24,000 unique secrets were found exposed in MCP configuration files alone [3].
- The architectural root cause — that large language models cannot reliably distinguish between instructions and data — has no close equivalent to SQL parameterization as a fix in current production systems, though research into structured context formats and multi-model verification is ongoing. Defense must be organizational and layered, not solely technical.
- Enterprises permitting AI coding assistants in production development workflows should treat these tools as privileged agentic systems and apply commensurate controls.

Background

AI coding assistants have become standard infrastructure in software development. Tools such as GitHub Copilot, Cursor, Claude Code, Amazon Q Developer, and OpenClaw are widely deployed across engineering organizations, providing code generation, automated refactoring, repository navigation, and multi-step agentic task execution. Their adoption has accelerated substantially through 2025 and into 2026, with Cisco's State of AI Security 2026 report finding that 83% of organizations plan to deploy agentic AI systems, though only 29% report feeling adequately prepared to secure them [4].

What distinguishes the current generation of AI coding assistants from earlier tools is their degree of environmental integration. Modern agentic coding tools read and write files across the project workspace, execute shell commands, make API calls, manage Git history, and connect to external services via the Model Context Protocol (MCP). These capabilities — essential to their usefulness — also constitute a substantial attack surface. The same mechanisms that allow an AI assistant to scaffold a full-stack application or triage a CI pipeline can, when manipulated, be turned toward credential theft, reverse shell deployment, or exfiltration of source code and secrets.

This research note examines three converging threat vectors that have materialized across the developer AI ecosystem in 2025 and early 2026: direct and indirect prompt injection attacks targeting AI coding environments, supply chain compromise via skill and extension marketplaces, and source code and credential leakage through AI tool interactions. The ClawHub skills ecosystem — the largest public registry of AI agent skills, associated with OpenClaw and Claude Code workflows — provides the clearest documented case study for how marketplace-scale distribution amplifies these risks.

Security Analysis

Prompt Injection: From Proof of Concept to Systematic Exploitation

Prompt injection against AI coding assistants has evolved from a theoretical concern into a documented and systematic exploitation class. The IDEsaster disclosure (December 2025), by researcher Ari Marzouk, represents the most comprehensive single-researcher audit of the category to date: more than 30 vulnerabilities were identified across ten or more products, with 24 CVEs assigned [1][19]. Every tool tested was found exploitable.

The IDEsaster attack chain follows a consistent pattern: an attacker plants a malicious payload in a file the AI assistant will read — a repository configuration file, a code comment, an issue description, or an MCP tool response — and that payload instructs the AI to weaponize legitimate IDE features. Three techniques proved particularly effective across products. In Remote JSON Schema Exploitation, the injected prompt instructs the AI to write a JSON file referencing an attacker-controlled schema URL; the IDE fetches this schema automatically, leaking context as URL parameters. In IDE Settings Overwrite, the AI modifies `.vscode/settings.json` to redirect code validation tools (linters, formatters) to attacker-controlled executables, achieving remote code execution the next time any file is saved. Multi-Root Workspace Manipulation exploits VS Code's multi-root workspace feature to bypass file scope restrictions and execute code through configuration manipulation alone.

CVE	Product	Technique	Impact	Source
CVE-2025-53773	GitHub Copilot	Auto-approve manipulation	RCE (CVSS 7.8)	[1]
CVE-2025-49150	Cursor	Remote JSON schema	Data exfiltration	[1]
CVE-2025-54130	Cursor	Settings overwrite	RCE	[1]
CVE-2025-54135	Cursor (CurXecute)	MCP config rewrite via Slack	RCE without user approval	[18]
CVE-2025-61260	OpenAI Codex CLI	Command injection	Arbitrary shell execution	[1]
CVE-2025-54794	Claude Code	Path restriction bypass	File access outside workspace	[21]
CVE-2025-54795	Claude Code	Command injection	Arbitrary shell execution (CVSS 8.7)	[21]
CVE-2025-59536	Claude Code	Malicious SessionStart hooks	RCE on agent launch	[14]
CVE-2026-21852	Claude Code	Malicious ANTHROPIC_BASE_URL	API key exfiltration	[14]

The Rules File Backdoor, disclosed by Pillar Security in March 2025, illustrated a particularly insidious indirect vector [5]. Configuration files such as `.cursorrules` and `.github/copilot-instructions.md` — shared via GitHub forks and pull requests — can embed invisible Unicode characters (zero-width joiners, bidirectional text markers) that are undetectable by human code review but fully legible to AI models. Injected instructions in these files can direct the AI to perform malicious actions and suppress its own activity logs. GitHub subsequently introduced warnings for files containing hidden Unicode characters [5], but the underlying problem — that AI models treat configuration files as trusted instruction sources — persists. A related technique discovered by Lasso Security involves prompt injection embedded within CLAUDE.md hook configurations, where malicious instructions delivered through seemingly routine project files can direct the AI assistant to execute attacker-controlled commands without triggering visible tool-call prompts [22].

The RoguePilot attack (Orca Security, February 2026) demonstrated that this class of attack can achieve repository takeover without any special privileges [6]. An attacker creates a GitHub Issue containing prompt injection in hidden HTML comments. When a developer opens a Codespace from that issue, Copilot receives the issue text as context and executes the injected instruction to exfiltrate the `GITHUB_TOKEN`, which can then be used to take over the repository. No code execution by the victim is required beyond opening a normal workspace.

Academic quantification of attack success rates is sobering. The AIShellJack framework (arXiv:2509.22040) tested 314 payloads across 70 MITRE ATT&CK techniques against production AI coding tools in 2025 [7]. Cursor in Auto Mode was vulnerable to 83.4% of tested attacks; Cursor with Claude 4 backend reached 69.1%; Copilot with Claude 4 showed 52.2%. Published defenses — including commercial prompt firewall products — showed bypass rates exceeding 85% under adaptive attack scenarios, substantially below vendor-reported effectiveness metrics [8].

ClawHub and the Marketplace Attack Surface

ClawHub is the official skills marketplace for OpenClaw, a self-hosted AI agent platform that integrates tightly with Claude Code workflows. Skills are Markdown-based instruction files that extend agent capabilities; the registry grew from approximately 150 skills at its November 2025 launch to more than 13,700 by February 2026, with a low trust threshold — any GitHub account older than one week could publish [9]. This combination of rapid growth, low friction publishing, and deep host-level trust created conditions that were exploited within weeks of the registry's public launch.

The ClawHavoc campaign, documented by Koi Security researcher Oren Yomtov, identified 341 confirmed malicious skills from a single coordinated campaign in an initial audit published February 1, 2026, with 335 attributable to the ClawHavoc campaign specifically and 6 from other actors in the same

audit window [2]. Subsequent analysis by CrowdStrike, Snyk, and Antiy CERT raised the total confirmed malicious skill count to at least 824 as the marketplace continued to expand [10][20]. VirusTotal independently validated the findings. The skills delivered two primary payloads: Atomic macOS Stealer (AMOS), a commodity infostealer targeting browser credentials, cryptocurrency wallets, SSH keys, and API keys; and functional reverse shells concealed within seemingly legitimate utility skills for cryptocurrency trading and browser automation. On macOS, attack chains used obfuscated base64 commands piped to bash; on Windows, a password-protected ZIP archive bypassed automated scanners. Several skills specifically targeted OpenClaw's own credential store at `~/.clawdbot/.env`, harvesting tokens used by the agent itself.

The threat is compounded by the privilege model under which OpenClaw and ClawHub skills execute. CrowdStrike's analysis notes that OpenClaw typically operates with terminal, file system, and in some configurations root-level access [10]. A malicious skill is not merely injecting code into an application — it is executing as the user, with full access to the development environment, secrets, and any connected infrastructure. This is equivalent to a malicious npm package with direct terminal access, without the mitigations that modern package managers have introduced over years of supply chain attack experience.

A separate vulnerability disclosed by Silverfort researchers in March 2026 compounded the trust problem [11]. ClawHub's backend exposed an unauthenticated `downloads:increment` RPC endpoint with no rate limiting, deduplication, or permission checks. Knowing a skill's deployment and skill IDs — both visible in public network traffic — an attacker could inflate download counts arbitrarily. A proof-of-concept test skill achieved over 3,900 apparent executions across 50 cities in six days and was installed by real organizations. This ranking manipulation mechanism allowed malicious skills to artificially inflate apparent popularity, undermining the trust signals that organizations might rely on when evaluating skill safety.

Source Code and Credential Leakage

The third threat vector is the unintended exposure of intellectual property and credentials through AI coding tool interactions. This leakage occurs through multiple pathways, each with distinct characteristics and mitigations.

At the ecosystem level, the GitGuardian State of Secrets Sprawl 2026 report found a 34% year-over-year increase in hardcoded secrets committed to public GitHub in 2025, totaling 28.65 million new exposures [3]. AI-assisted commits carried a secret-leak rate of 3.2%, compared to a 1.5% baseline across all public commits — more than double. Secrets related to AI services specifically showed an 81%

year-over-year increase, with 1.27 million AI service secrets leaked in 2025. Of particular concern for agentic development: 24,008 unique secrets were found exposed in MCP configuration files, with 2,117 confirmed valid at time of discovery.

Malicious extensions represent a direct, targeted exfiltration path. The MaliciousCorgi campaign (Koi Security, January 2026) discovered two AI coding extensions with a combined 1.5 million VS Code installations that captured every file opened and every source code modification, Base64-encoded, and transmitted it to a server with a Chinese domain registration [12]. The extensions remained available on the official Visual Studio Marketplace as of the disclosure date. Reports of VS Code publisher access token compromise have also surfaced, with leaked credentials creating a potential mechanism for malicious update distribution to existing installed user bases — though organizations should verify specific incidents against current vendor advisories.

Vulnerabilities in the tools themselves can also enable exfiltration. CVE-2025-55284, affecting Claude Code prior to v1.0.4, allowed injected prompts to read `.env` file contents and exfiltrate them via DNS subdomain encoding in a `ping` request to an attacker-controlled domain — bypassing network egress monitoring [13]. Check Point Research disclosed CVE-2026-21852, in which a malicious `ANTHROPIC_BASE_URL` set in a project configuration file could intercept API calls before any trust dialog appeared, capturing the user's plaintext Anthropic API key [14]. Three additional CVEs (CVE-2025-68143, CVE-2025-68144, CVE-2025-68145) were discovered in Anthropic's official Git MCP server, where path traversal and argument injection flaws exposed repository data accessible through that integration [23]. These vulnerabilities typically activate on repository clone or workspace open, before any user interaction. In a distinct incident in March 2026, approximately 500,000 lines of Claude Code source code across roughly 1,900 files were exposed in an Anthropic data breach [24], demonstrating that the AI toolchain itself — not only developer-authored code — represents a target for adversaries seeking proprietary implementation details.

The supply chain dimension extends to the AI tools' own dependencies. SANDWORM_MODE, a family of 19 typosquatted npm packages discovered in February 2026, deployed a rogue MCP server into hidden directories after a 48-hour delay, registering deceptive tool names (`index_project`, `lint_check`) with embedded prompt injection payloads that instructed AI assistants to silently extract SSH keys, AWS credentials, npm tokens, and environment variables — and to hide this activity from the user [15]. The Nx malicious package incident (August 2025) specifically invoked Claude Code with `--dangerously-skip-permissions`, Gemini CLI with `--yolo`, and Amazon Q with `--trust-all-tools` to bypass confirmation prompts during exfiltration [16].

Recommendations

Immediate Actions

Organizations should audit current AI coding assistant deployments with urgency. Inventory which tools are in use, including VS Code extensions, MCP server configurations, and any OpenClaw or ClawHub skill installations. Extensions and skills should be reviewed against known malicious indicators from the ClawHavoc campaign. MCP configuration files — particularly `mcp.json` and any project-level MCP settings — should be audited for unrecognized or community-sourced servers. AI tool versions should be validated against available patches for the CVEs documented in this note.

Repository configuration files that serve as AI instruction sources (`.cursorrules`, `CLAUDE.md`, `.github/copilot-instructions.md`) should be treated as trust-sensitive artifacts and subjected to the same review process as deployment configuration. Developers should understand that sharing or accepting these files from external repositories introduces an indirect prompt injection risk. GitHub's hidden-Unicode warning feature should be enabled for all organizational repositories.

Short-Term Mitigations

Enterprises should establish an approved AI tool list and block or restrict unapproved tools through technical controls where possible. For approved agentic tools, the principle of least privilege should guide configuration: MCP server approvals should be minimal and explicit, working directory restrictions should be enforced, and shell command execution should require human approval for operations outside a defined safe list. Claude Code's deny rules, Cursor's privacy mode, and equivalent features in other tools should be configured and verified — noting the Adversa AI finding that Claude Code's security deny rules may stop functioning after 50 subcommands, which Adversa attributes to apparent performance optimization [17].

Secret management should be hardened against AI-driven leakage. API keys, credentials, and tokens required by development workflows should be accessed through secrets managers rather than `.env` files, reducing the impact of exfiltration attacks targeting flat credential stores. Pre-commit hooks and CI checks should scan for hardcoded secrets in AI-assisted commits at elevated sensitivity thresholds given the documented higher leak rate. MCP configuration files should be added to secrets scanning scope.

Strategic Considerations

The fundamental architectural challenge is that language models cannot reliably distinguish between instructions and data. Every piece of content that an AI coding assistant reads — including repository files, code comments, issue descriptions, web content, and MCP tool responses — is a potential instruction channel. This is analogous to SQL injection before parameterization, but with no equivalent technical fix available in current production systems; research into structured context formats and multi-model verification pipelines is ongoing but has not yet produced widely deployed mitigations. Defense must be layered and organizational: assume that AI coding tools operating in complex environments will be manipulated, and design controls around limiting blast radius rather than preventing compromise entirely.

Organizations should classify AI coding assistants as agentic systems operating with elevated privilege and apply the same governance frameworks used for service accounts and automation infrastructure. This includes logging and auditability of AI tool actions, periodic review of AI-generated commits for anomalous patterns, and clear policies on which data classifications may be processed in AI coding tool context windows. Vendors should be evaluated on their MCP permission models, tool approval workflows, and patch cadence for prompt injection vulnerabilities. The IDEsaster research found that vendor response quality varied substantially, with some tools receiving CVEs and patches while others opted for security warnings or took no action.

CSA Resource Alignment

This research note connects directly to several active CSA frameworks and working group outputs.

MAESTRO (Multi-Agent Environment Security Threat and Risk Operations) provides the foundational threat modeling framework for agentic AI systems. The attacks documented here — prompt injection via environmental context, tool call manipulation, credential harvesting through agentic workflows — map to MAESTRO threat categories including goal hijacking, unauthorized action chains, and context poisoning. MAESTRO's layered trust model is directly applicable to MCP server approval workflows and skill marketplace vetting.

AI Controls Matrix (AICM), as a superset of the Cloud Controls Matrix (CCM), provides specific control domains relevant to AI coding assistant deployments. AICM's supply chain integrity controls apply to skill marketplace vetting and extension audits. Its data protection controls govern what information may be processed in AI tool context windows. Its identity and access management domain applies to service account principles for AI tool privilege configuration.

STAR (Security Trust Assurance and Risk) for AI provides a mechanism for organizations to assess and communicate the security posture of AI tool deployments, including coding assistants. Vendors and internal teams can use STAR assessments to document MCP permission boundaries, data handling practices, and vulnerability response timelines.

The CSA Zero Trust guidance applies directly to the principle that AI coding tools should not be trusted implicitly based on their position in the developer environment. Every MCP server connection, skill installation, and configuration file introduced into an AI coding workflow should be verified, minimally privileged, and auditable. The RoguePilot and CurXecute attack chains both succeeded specifically because ambient trust — the assumption that content already present in the development environment is trustworthy — was extended to attacker-controlled material.

The **OWASP Top 10 for Agentic Applications (ASI)** framework, referenced extensively in CSA working group materials, provides the most direct mapping for the attack classes documented here: ASI01 (Agent Goal Hijack) for prompt injection attacks, ASI04 (Agentic Supply Chain) for the ClawHub malicious skills campaign, and ASI05 (Unexpected Code Execution) for RCE via IDE configuration manipulation.

References

[1] Marzouk, Ari (MaccariTA). "IDEsaster: 30+ Vulnerabilities in AI-Powered IDEs." MaccariTA Security Research, December 2025.

[2] Yomtov, Oren (Koi Security). "ClawHavoc: 341 Malicious ClawHub Skills Found by the Bot They Were Targeting." Koi Security, February 2026.

[3] GitGuardian. "The State of Secrets Sprawl 2026." GitGuardian, March 2026.

[4] Cisco. "Cisco State of AI Security 2026 Report." Cisco Security Blog, 2026.

[5] Pillar Security. "New Vulnerability in GitHub Copilot and Cursor: How Hackers Can Weaponize Code Agents via Rules Files." Pillar Security, March 2025.

[6] Orca Security. "RoguePilot: GitHub Copilot Vulnerability Enabling Repository Takeover." Orca Security, February 2026.

[7] Arxiv. "Your AI, My Shell: Demystifying Prompt Injection Attacks on Agentic AI Coding Editors." arXiv:2509.22040, 2025.

[8] Arxiv. "Prompt Injection Attacks on Agentic Coding Assistants: A Systematization of Knowledge." arXiv:2601.17548, January 2026.

[9] Felo AI. "ClawHub: The Skills Marketplace for Claude Code." Felo AI Blog, 2026.

[10] CrowdStrike. "What Security Teams Need to Know About OpenClaw." CrowdStrike Blog, 2026.

[11] Silverfort. "ClawHub Vulnerability Enables Attackers to Manipulate Rankings to Become the Number-One Skill." Silverfort, March 2026.

[12] The Hacker News. "Malicious VS Code AI Extensions with 1.5 Million Installs Exfiltrate Source Code." The Hacker News, January 2026.

[13] Embrace the Red. "Claude Code Exfiltration via DNS Requests – CVE-2025-55284." Embrace the Red, 2025.

[14] Check Point Research. "RCE and API Token Exfiltration Through Claude Code Project Files – CVE-2025-59536 / CVE-2026-21852." Check Point Research, February 2026.

[15] Kodem Security. "SANDWORM_MODE: A New Shai-Hulud Style npm Worm Threatening Developer AI Toolchain Security." Kodem Security, February 2026.

[16] Snyk. "Weaponizing AI Coding Agents for Malware: The Nx Incident." Snyk Security Research, August 2025.

[17] Adversa AI. "Claude Code Security Bypass: Deny Rules Silently Disabled." Adversa AI, 2025.

[18] Tenable. "FAQ: CVE-2025-54135 and CVE-2025-54136 – CurXecute and MCPoison Vulnerabilities in Cursor." Tenable, 2025.

[19] The Hacker News. "Researchers Uncover 30 Flaws in AI-Powered IDEs – IDEsaster." The Hacker News, December 2025.

[20] Snyk. "Inside the 'clawdhub' Malicious Campaign." Snyk Security Research, February 2026.

[21] Cymulate. "InversePrompt: CVE-2025-54794 and CVE-2025-54795 in Claude Code." Cymulate, 2025.

[22] Lasso Security. "The Hidden Backdoor in Claude Coding Assistant." Lasso Security, January 2026.

[23] The Register. "Anthropic Quietly Fixed Serious Path Traversal and Injection Flaws in Its MCP Git Server." The Register, January 2026.

[24] Fortune. "Anthropic Leaks Claude Code Source Code." Fortune, March 2026.