snyk

SNYK RESEARCH

Inside the Agentic Development Supply Chain

June 2026

EXECUTIVE SUMMARY

The development environment is the new security frontier

AI agents are no longer just assisting developers. They are actively building software: pulling in tools, executing actions, and generating production-ready code at machine speed. The data shows measurable security exposure already present in scanned developer environments.

This report combines anonymized aggregated telemetry from nearly 10,000 developer environments with analysis of agent skills observed across participating enterprise environments. Together, these datasets provide a view into the scale of AI tooling adoption, the growth of the agent supply chain, and the risks emerging as software development becomes increasingly agent-driven.

The state of the agentic development supply chain.

37% of developers run 3 or more AI coding environments

50.8% of developers already have MCP server connections running

18 avg developers with skills installed (22.8%) averaged 18, ranging from one skill to more than 100 across the dataset

28% of skills expose agents to uncontrolled third-party content (W011 — 2,207 of 7,742 skills)

1 in 12 developers with MCP servers have a HIGH or CRITICAL finding today

SECTION 1

The agentic development surface

Growing adoption and expanding exposure across developer environments

AI coding tools are no longer an experiment at the margins of software development. 43% of the developers in our scan are running two or more AI coding environments, with Claude, Cursor, Windsurf, Gemini, Copilot, Kiro, and others frequently coexisting on the same machine. 37% are running three or more.

This creates a compounding challenge: each AI coding environment can independently connect to external services via MCP (Model Context Protocol), which gives agents access to tools, APIs, and data sources. Just over half of the developers we scanned, 50.8% (4,939 out of 9,723), already have at least one MCP server running. That is an active outbound connection from a developer machine to an external tool, often installed directly on developer machines, where review processes may be inconsistent or absent.

The shift is equally significant at the behavioral layer. Skills, the instruction files that define how an agent behaves, its defaults, personas, and capabilities, are being pulled from shared repositories and third-party ecosystems with limited visibility into whether vetting occurred. Across the environments analyzed, 22.8% of developers had at least one skill installed. Among those developers, installs averaged 18 and ranged from one to more than 100.

And at the same time, the definition of a developer is changing. Product managers, operations teams, and business users are using AI tools to build and modify software. The attack surface is expanding beyond engineering teams, and many existing security programs may not yet account for this layer of tooling and configuration.

Why securing the agent supply chain matters

The risk in agentic development is not just that agents may generate code containing vulnerabilities. While that risk remains, a new challenge is that agents consume a supply chain of external components, such as MCP servers, skills, integrations, or models, that operate entirely outside existing security controls.

In traditional software development, the supply chain consists of packages and dependencies with known provenance, scanned with SCA tooling, and tracked in SBOMs (Software Bills of Materials). The agent supply chain operates differently: MCP servers are installed ad hoc directly on developer machines, and skills are downloaded from shared repositories, both of which change the instructions agents follow and the actions they can take. These components may fall outside many existing scanning, inventory, and governance processes.

The consequence is that if an agent installs a malicious MCP server or loads a compromised skill, it may execute instructions that exfiltrate data, bypass controls, or take destructive action, potentially without being surfaced through existing security workflows.

In a recent incident, an AI agent deleted an entire production database, including its backups, in under ten seconds. The agent was attempting to fix a routine issue, but with access to the wrong credentials and no guardrails on its behavior, it took a destructive, irreversible action against the wrong environment. In this case, there was no human approval or a system in place to stop it.

Why traditional AppSec falls short

Traditional application security was designed for a fundamentally different development model consisting of human-written code, controlled pipelines, and artifact scanning after code is created. Each of these assumptions breaks down under agentic development.

The specific gaps:

- Traditional AppSec lacks in visibility into MCP servers, skills, or agent configuration files, which typically live in directories on developer machines that scanners never reach
- SCA and SAST tools scan code artifacts after they exist, not agent instructions before they execute
- Controls that scan artifacts after code is written are not designed to evaluate an agent's actions as they happen.
- AI coding tools introduce risk before code is committed to any repository, outside any CI pipeline or review gate
- The agent supply chain does not have an equivalent of an SBOM, standard vetting process, or centralized governance model

The result is a growing gap between the speed of AI-driven development and the ability to maintain governance and trust over how software is built. The data below quantifies exactly how wide that gap already is.

SECTION 2

What the data shows

Growing adoption and expanding exposure across developer environments

All findings below are based on Evo Agentic Development Security (ADS) telemetry. See the Data Sources appendix at the end of this report for full methodology.

01 · AI coding environments: already widespread

43% of developers run two or more AI coding environments. 37% run three or more. Claude, Cursor, Windsurf, Gemini, Codex, VS Code, and Kiro frequently coexist on the same machine.

STAT	FINDING
43%	of developers run 2+ AI coding environments simultaneously
37%	run 3 or more environments simultaneously
1.84	average AI coding environments per developer

---

02 · More than half of developers have MCP connections

Nearly 5,000 of the 9,723 developers scanned (50.8%) were already running at least one live MCP connection at the time of scan. These are outbound connections from developer machines to external tools and services installed directly on developer machines.

STAT	FINDING
50.8%	of developers scanned had at least one MCP server installed
4,939	developers with MCP servers out of 9,723 scanned

---

03 · The agent supply chain is already at scale

The MCP ecosystem has expanded into a substantial new software supply chain. The data shows proliferation across scanned developer environments, with limited central inventory visible from the scan data.

STAT	FINDING
4,524	unique MCP server configurations discovered across fewer than 10,000 developers
8,146	total MCP server configurations observed across scanned machines
13+	Among developers with MCP servers installed, the top 1% run 13 or more, simultaneously

04 · Agents are connected to production systems

The most common MCP servers connect to functional developers and business systems. They give agents live access to code repositories, browser automation, content management, project management, and file systems.

The most prevalently configured MCP servers

SERVER	CATEGORY	WHAT IT GRANTS	INSTALLS
Playwright	Browser automation	Agent controls a full browser — can navigate, submit forms, extract data	151
GitHub	Code repository (×4 variants)	Agent reads/writes code, creates branches, submits pull requests	142
Context7	Documentation (×3 variants)	Agent fetches and injects live documentation into context	129
Filesystem	Local file access (×2 variants)	Agent reads/writes local files and directories	63
Atlassian	Project management	Agent creates/modifies issues, reads sprint data	39
Figma	Design	Agent reads and modifies design assets	36

---

05 · Risk is already embedded in the supply chain

Across the MCP server population, risk signals are common enough to warrant attention. The finding densities below represent the state of developer environments today, before any specific attack has been attempted.

STAT	FINDING
1 in 7	developers with MCP servers have at least one security finding in their setup
1 in 12	developers with MCP servers have a HIGH or CRITICAL finding in their setup today
8+	findings on average per affected server, suggesting that findings often cluster
1 in 13	MCP servers with any findings contain prompt injection

---

06 · MCP server security findings — full breakdown

Snyk's agent scanner uses a tiered severity system. E-codes (Error) indicate confirmed, high-impact risk. W-codes (Warning) indicate risk signals that require investigation.

The most prevalently configured MCP servers

CODE	SEVERITY	FINDING	SERVERS	% OF ALL	FINDINGS
W001	LOW	Suspicious words in tool description - prompt injection indicators	324 servers	7.2%	2,041
E001	CRITICAL	Confirmed prompt injection in tool description	52 servers	1.1%	392
E002	HIGH	Cross-server tool shadowing	6 servers	0.1%	117

E001 vs. W001

W001 flags suspicious language in MCP tool descriptions. It may indicate language patterns associated with prompt injection. E001 is a different and more serious finding: it confirms adversarial prompt injection: instructions embedded in tool descriptions that could hijack agent behavior. 392 confirmed E001 findings across 52 servers. Existing security scanning focuses on global user-level configurations and does not include project-level directories on developer machines, where project-specific MCP configurations may live. ADS introduces purpose-built detection specifically for this attack surface.

07 · Agent skills — scale of the second supply chain layer

Skills are instruction files that define how an agent behaves, including defaults, personas, and reusable capabilities. They are the second supply chain layer of agentic development. Unlike MCP servers, which define what tools agents can access, skills define what agents are instructed to do.

STAT	FINDING
22.8%	of developers have at least one skill installed
7,742	unique skill files discovered across scanned environments
18 avg	developers with skills installed averaged 18, ranging from one skill to more than 100

---

08 · Agent skills — security findings

The skills risk picture contains both systemic misconfiguration findings and confirmed adversarial content. The E-code findings are higher-confidence detections, distinct from warning-level risk signals. They identify malicious code and prompt injection patterns that could compromise agent behavior, without requiring the report to infer author intent.

Agent skills findings

CODE	SEVERITY	FINDING	DESCRIPTION	COUNT
W011	WARN	Third-party content exposure	Skills expose agents to uncontrolled third-party content at runtime	2,041
W012	WARN	Unverifiable external dependency	Fetches live instructions from an external URL — the author can silently alter agent behavior	841
W007	WARN	Insecure credential handling	Skills mishandle API keys, tokens, or secrets	514
E006	ERROR	Malicious code pattern	Confirmed malicious code embedded in skill files	98
E004	ERROR	Prompt injection (skill)	Hidden instructions manipulating agent behavior at install time	43
W009	WARN	Direct money access	Skills grant agents access to financial accounts	28

Evidence of malicious and prompt injection patterns appearing within skill ecosystems

E006 (98 confirmed malicious code patterns) and E004 (43 prompt injection instances in skills) are categorized separately from misconfiguration findings because they involve malicious code or prompt injection patterns. W012 (841 instances) introduces a runtime dependency on an external URL: the instructions an agent receives could change after install, whether through an intentional update by the skill author, an error, or third-party compromise of that endpoint. The security risk is the dependency itself, regardless of intent. W009 (28 instances) grants agents direct access to financial accounts. The combination of broad adoption, low governance, and confirmed malicious code and prompt injection findings makes skills a high-priority surface for security teams.

Implications for security leaders

AI-driven development is not a future-state concern. The gap between development velocity and security coverage is visible in the scanned environments.

Four actions for security leaders:

- Gain visibility into what agents are using. MCP servers and skills are proliferating without centralized oversight. Discovering what is in use is the starting point, and the data above illustrates how broad and fast that proliferation already is.
- Extend policy to cover agent supply chain risk. Existing AppSec policies do not cover MCP servers or skills. Governance must expand to include the tools, instructions, and integrations that agents consume, not just the code they generate.
- Evaluate behavioral controls for high-autonomy agents. As agent autonomy increases, the blast radius of ungoverned action increases with it. The findings suggest teams may need to evaluate policy enforcement closer to the execution layer.
- Align security with how software is now being built. Traditional code-centric AppSec programs must extend into the agentic development environment, operating directly within agent workflows, not only downstream of them.

For organizations adopting AI-driven development, the data points to a need for visibility and governance that does not unnecessarily slow teams down. The findings suggest that security reviews should consider not only the generated code but also the systems and instructions that produce it.

SECTION 4

Evo Agentic Development Security

The findings in this report point to a new security challenge: organizations need visibility and control over the systems AI agents rely on to build software.

Evo Agentic Development Security (ADS) is Snyk's solution for securing AI-driven development. ADS operates directly within agent workflows, helping organizations secure what agents use, what they do, and what they generate.

Secure the agent supply chain

Discover and govern MCP servers, skills, models, and integrations before they enter agent workflows. ADS provides visibility into the components agents rely on and enables organizations to apply policies that allow, restrict, require approval, or block adoption based on risk.

Govern agent behavior

Apply guardrails directly within the agent execution loop. ADS evaluates actions in context and can block unsafe behavior, enforce approvals, redact sensitive data, and apply policy before actions occur.

Ensure trusted output

Integrate security directly into AI coding workflows. ADS validates AI-generated code as it is created, helping organizations identify and address security issues before code reaches repositories or CI pipelines.

Together, these capabilities help organizations adopt AI-driven development at scale without sacrificing visibility, governance, or trust.

ADS is part of Evo by Snyk, alongside AI-SPM and Continuous Offensive Security, providing coverage across how AI systems are governed, tested, and built.

Learn more: **snyk.io/evo**

APPENDIX · DATA SOURCES

About this data

This report draws on two distinct datasets from Evo Agentic Development Security (ADS) scans.

The MCP server and skill analyses are based on different populations and should be interpreted independently.

MCP Server Data
Derived from anonymized scans of developer environments conducted by members of the security and software development community. The dataset includes 9,723 developers and captures MCP server configurations, installations, and associated security findings discovered on developer machines.

Skills Data
Anonymized analysis of participant environments in early Evo ADS programs. Scoped to skills installed directly on developer machines, globally, and at the user level. Skills committed to codebases for shared use are a separate, unmeasured layer not captured here.

All data collected and analyzed by Snyk, 2026. Telemetry is anonymized. Agent runtime behavior is out of scope for both datasets.