SNYK CHEATSHEET

6 Ways AI Agents Act Beyond Traditional Controls

AI agents use tools, take actions, and execute workflows across systems at machine speed. Software is increasingly created by autonomous systems that can pull in external tools, interact with environments, and execute multi-step workflows.

While traditional AppSec focused on securing code solely after it was written, agentic development requires organizations to secure the entire Agentic Development Lifecycle (ADLC) at three vectors – all without slowing innovation:

- → WHAT AGENTS USE
- → WHAT AGENTS DO
- → WHAT AGENTS GENERATE

In each of these vectors, there are new ways agents can operate beyond traditional security controls and introduce risks that didn't used to be part of the software development lifecycle.

snyk

01 Calling untrusted tools

AI agents increasingly discover and invoke external MCP servers, APIs, plugins, and skills autonomously. Unlike traditional software dependencies that undergo formal review, these tools can be pulled in dynamically with little visibility into their origin, permissions, or behavior.

A compromised MCP server, malicious plugin, or unsafe skill can influence agent behavior in real time, introducing risk long before code ever reaches a repository. Because these tools operate dynamically during execution, AppSec tools often lack visibility into how they are invoked or what actions they trigger.

Example scenario: A development agent connects to a publicly available MCP server to automate dependency updates. The server has been compromised and injected a malicious package into the workflow, giving attackers access to internal build systems and deployment pipelines.

Business impact: risk of supply chain compromise, data exposure, and operational disruption.

02 Pulling from unapproved data sources

AI agents can autonomously retrieve information from repositories, applications, internal databases, ticketing systems, cloud storage, and external services as they complete tasks. Unlike applications with tightly scoped integrations, agents dynamically decide what information to access based on context and goals.

Sensitive data from unauthorized sources may be pulled into prompts, workflows, or external systems without developers even realizing it. Static security controls often lack visibility into these agent-driven interactions happening across multiple environments in real time. Organizations risk exposing sensitive customer information, internal intellectual property, or regulated data through AI workflows that were never intended to handle it.

Example scenario: An agent troubleshooting a customer issue retrieves production support tickets containing sensitive customer data, then includes that information in prompts sent to an external AI service to generate debugging recommendations.

Business impact: compliance issues, governance gaps, and increased exposure to data leakage.

03 Executing unsafe commands

Instead of simply recommending actions to a developer, AI agents can directly interact with environments, modify systems, and trigger operational changes in real time. They can execute scripts, terminal commands, infrastructure changes, and API calls autonomously as part of multi-step development workflows.

While traditional workflows rely on human review and approval before sensitive actions are executed. Agentic workflows compress planning, decision-making, and execution into a single automated loop operating at machine speed. Without clear guardrails, agents may take destructive, unauthorized, or high-risk actions while attempting to complete a task.

Example scenario: An agent attempts to “fix” a failed deployment by automatically running a cleanup script on the wrong Kubernetes cluster, deleting active production containers, and causing a widespread service outage.

Business impact: Infrastructure outages, unauthorized system changes, or operational disruption across environments

04 Accessing restricted systems

AI agents often operate with broad permissions across repositories, CI/CD pipelines, cloud environments, internal APIs, and developer tooling so they can complete tasks without constant human intervention. As organizations scale agentic development, agents may access systems, services, or environments beyond the original scope of a workflow simply because they have the credentials or permissions to do so.

Traditional identity and access controls were not built to monitor how autonomous agents move between systems or how quickly they can act once access is granted. When agents operate with excessive or poorly governed permissions, it becomes harder to enforce security boundaries and maintain compliance across environments.

Example scenario: A coding agent troubleshooting a staging application uses inherited cloud credentials to access production infrastructure, retrieves deployment secrets, and unintentionally exposes them in a shared internal debugging workflow.

Business impact: Unauthorized access to sensitive systems, expanded blast radius during incidents, and reduced visibility into how critical resources are being used.

05 Generating insecure code

While AI agents can dramatically accelerate development by generating production-ready code, infrastructure configurations, and application logic in real time, the code may introduce vulnerabilities, insecure patterns, or unsafe logic from the moment it is created, often before traditional security review processes occur.

Traditional AppSec workflows were designed around human-paced development, where code could be reviewed after it was written. Agentic development compresses generation and deployment into much faster workflows, increasing the likelihood that insecure code reaches repositories, pipelines, or production environments before issues are identified.

Example scenario: An AI coding agent generates a new authentication workflow that unintentionally exposes administrative API endpoints without proper authorization checks. The code is committed and deployed before manual review catches the issue.

Business impact: Exploitable security flaws, increased remediation costs, and the spread of insecure patterns across development environments.

06 Generating vulnerable dependencies

AI agents also recommend and introduce packages, libraries, SDKs, and frameworks as part of development workflows. In many cases, agents optimize for functionality or speed without fully evaluating the security, provenance, or maintenance posture of the dependencies they select.

As agents autonomously build applications, vulnerable or unapproved dependencies can rapidly spread across repositories and pipelines before security teams have visibility into what was added or why. Traditional dependency governance processes often struggle to keep pace with the machine-speed of software development.

Example scenario: An AI agent building a new internal application automatically imports an outdated open-source package with a known remote code execution vulnerability because it appears frequently in public code examples.

Business impact: An expansive software supply chain attack surface that introduces known vulnerabilities into production systems, and creates long-term technical debt that becomes difficult to unwind later.

Safely scale agentic development without slowing it down

AI agents are now part of how modern software is built. They accelerate development by connecting tools, executing workflows, and generating code at a scale and speed traditional processes were never designed to support. But as agents become more autonomous, organizations need a new security model built for the autonomous systems now creating software.

Securing agentic development doesn't mean blocking innovation or slowing teams down. It means giving organizations visibility and governance across what agents use, do, and generate. With the right controls in place, developers and AI agents can move quickly while security operates continuously in the background. This allows teams to enforce trusted boundaries, reduce risk, and enable organizations to safely scale AI-driven development.

Want to see how organizations are securing agentic development in practice?

Book a demo today