
Securing
AI-Generated Code
The Executive Guide to Vibe Coding
paloalto NETWORKS
CORTEX

vibe coding (vīb-kōd-diŋ), *n.* the practice of using AI and natural language prompts to write code quickly, focusing on intent over exact syntax.

"Vibe coding" is the fastest way to ship new code. It's also the fastest way to introduce risks.

Discover the unique security challenges of using AI to generate code and learn practical strategies executives can use to securely accelerate development.

The Rise of Vibe Coding

AI is changing how software is built. Developers are now turning to AI prompting as the new way to code. This vibe coding approach reflects a shift where developers express design intent and AI translates it into working code. Vibe coding replaces much of the manual syntax and structure used in traditional coding with natural language and automation, making the crafting of precise prompts as critical as writing the code itself.

In a recent survey, 99% of organizations reported having adopted AI-assisted development, indicating having no plans of stopping.[1] Prototyping, testing, and iteration happen in seconds instead of days. Development becomes more creative and less repetitive so teams can focus on ideas rather than implementation.

The speed and scale that vibe coding enables, however, creates new risks. AI-generated code can introduce unseen vulnerabilities or reuse insecure patterns, which is forcing organizations to fundamentally change their approach to securing software.

Hidden Threats in AI-Created Code

AI models are trained to generate code by learning patterns from massive datasets of existing code and natural language so they can predict the next token or sequence based on a given prompt. Because AI has no understanding of an organization's environment or security requirements, it might produce insecure defaults. It also might leverage open-source packages with deep dependency chains and introduce vulnerabilities such as injection flaws, weak authentication, and poor cryptography. In fact, one-third of all AI-generated code is vulnerable.[2]

Considering the velocity at which AI development accelerates, security teams have less time to evaluate this code before it reaches production. Traditional AppSec tools, like static application security testing (SAST), Software Composition Analysis (SCA), and infrastructure as code (IaC) scanning, can't always catch these weaknesses in time, letting vulnerable code slip through. These risks tend to compound quickly. When insecure code reaches production, remediation takes 10x longer on average. Considering that security teams typically remediate only 10% of outstanding risks each month,[3] the security backlogs grow faster than teams can address them.

Soon, a gap widens between the speed of AI-generated code and security teams' ability to keep it safe.

99%

of organizations are using AI-assisted coding.

1/3

of AI-generated code is vulnerable.

1. *The State of Cloud Security Report 2025*, Palo Alto Networks, December 9, 2025.
2. "Every 1 of 3 AI-Generated Code Is Vulnerable: Exploring Insights with CyberSecEval," SOCRadar, updated: March 18, 2024.
3. *State of Software Security 2025: A New View of Maturity*, Veracode, February 26, 2025.

Developers Lose Touch with Their Code

As AI takes on a larger role in writing code, developers become less familiar with their codebase. When code is generated based on vibes or outcome-focused prompts, they might not fully understand why a particular implementation was chosen.

Debugging and incident response become more complex as the human connection to the code diminishes. It's harder to identify vulnerabilities, spot insecure patterns, and respond quickly when issues arise. This situation is similar to having developers revisit decisions made months earlier to fix a problem. Except in this case, they didn't write the code at all.

> **As AI takes on a larger role in writing code, developers become less familiar with their codebase.**

AI, Open Source, and the Supply Chain Compliance Challenge

AI-generated code frequently leans on open-source libraries, which introduce a heightened risk if those dependencies aren't rigorously vetted. The AI models might suggest or include packages that are outdated, insecure, or poorly maintained. These dependencies can also have deep transitive chains, amplifying risk in ways that traditional tooling might not fully detect.

AI coding assistants can also hallucinate plausible-sounding yet nonexistent package names, creating an opening for "slopsquatting" attacks, where threat actors register those fabricated names and publish malicious code to public repositories.

Increasing regulatory requirements mean organizations must treat AI and code dependency management as a critical security priority. Otherwise, organizations might experience real risks where insecure or noncompliant AI-generated code won't conform to evolving supply chain regulations, particularly in heavily regulated markets.

In both the US and EU, compliance with regulatory requirements adds pressure for organizations to actively manage AI-generated dependencies, for example:

United States

Software vendors must follow Executive Order 14028 and NIST SP 800-218 by implementing secure development practices, maintaining software bill of materials (SBOMs), managing vulnerabilities, and providing attestations for federal software procurement.

European Union

Under the NIS2 Directive and Cyber Resilience Act, organizations must ensure security by design, maintain SBOMs, manage vulnerabilities throughout the product lifecycle, and comply with conformity assessments and incident-reporting obligations.

Besides open-source packages and SBOMs, the broader development ecosystem—including workflow tools and ticketing systems—can be exploited to inject malicious changes or compromise the software supply chain if access controls and monitoring are insufficient.

Securing AI Agents and Code Identities to Prevent Secret Exposure

Whether by resurfacing credentials buried in code, reusing sensitive tokens from context, or unintentionally exposing environment details in generated outputs, AI models can leak secrets.[4] The risk of accidental secret disclosure grows significantly when these models gain deeper access to repositories, pipelines, and developer systems.

Modern integrated development environments are also shifting into multiagent environments where multiple AI agents collaborate on coding tasks. Each agent effectively behaves like a developer with its own permissions, decision-making patterns, and code access. This approach introduces a new identity layer that most organizations are not yet equipped to secure.

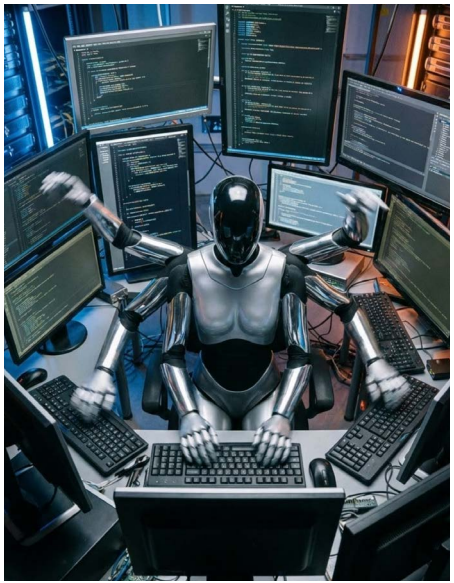
The challenge is that these agents often operate with broad and opaque access rights, making it difficult to track what they can see, modify, or leak. Without clear guardrails, unmanaged agent identities and unrestricted model access create real exposure risks across the software supply chain.

The Risks of AI-Generated Malicious Code

Prompt injection and adversarial inputs further increase risk. Maliciously crafted prompts can trick AI into producing harmful code, amplifying potential threats across systems.

Centralized data sources, such as code repositories or data lakes, might be targeted if AI-generated outputs are not properly monitored. A single malicious or flawed suggestion could propagate widely.

To mitigate these threats, organizations need strict guardrails, continuous monitoring of AI outputs, and clear approval workflows. Prompt injection and adversarial attacks are real and growing concerns, making proactive oversight essential whenever AI has access to critical systems.



4. Ravie Lakshmanan, "12,000+ API Keys and Passwords Found in Public Datasets Used for LLM Training," The Hacker News, February 28, 2025.

Securing AI-Generated Code: Strategies and Best Practices

By taking a prevention-first approach, your organization can enjoy the speed and creativity of vibe coding without leaving security behind. Your teams start by building security into development workflows from day one, with proactive oversight, automated safeguards, clear guardrails, and humans in the loop. By embedding security early, companies can accelerate development, while ensuring vibe coding is secure, compliant, and reliable.

Vibe Coding Security Checklist

- ☐ **Validate inputs and outputs:** Sanitize prompts and separate trusted instructions from untrusted data. Require the AI to validate logic checks and code through SAST after development and before merging.
- ☐ **Use only approved versions of code generation tools and AI models:** Continuously monitor for the use of unapproved systems.
- ☐ **Keep humans in the loop:** Require human review and approval of AI-generated code by using the pull-request (PR) process.
- ☐ **Monitor autonomous refactoring for drift:** Track AI-driven refactoring to prevent security, compliance, or behavior drift over time.
- ☐ **Secure the software supply chain:** Scan all AI-generated code with SCA and monitor for outdated packages or slopsquatting.
- ☐ **Maintain SBOMs and AI-BOMs:** Track and update all dependencies and AI-generated components to quickly identify vulnerabilities and meet compliance requirements.
- ☐ **Implement the principle of least agency:** Grant AI agents only the minimum permissions required and restrict access to sensitive files and any destructive commands.
- ☐ **Restrict AI agents to development and test environments:** Prevent overprivileged AI agents by ensuring they cannot access both development and production environments.
- ☐ **Scan and block unsafe models:** Vet models for malicious code, vulnerabilities, and malware before adoption, preventing the deployment of compromised or unverified AI components.
- ☐ **Proactively red team AI apps and agents:** Conduct adversarial simulation exercises to identify logic flaws, guardrail bypasses, and susceptibility to prompt injection attacks before deployment.
- ☐ **Deploy runtime threat detection:** Implement real-time monitoring to instantly detect and block threats, such as jailbreaks and other prompt injection attacks, while your AI applications are live.

Vibe Code Securely with Cortex Cloud

Adopt AI coding tools without compromising security. Cortex® Cloud™ Application Security Posture Management (ASPM) unifies visibility across the software development ecosystem. It correlates findings from native code scanners—including software supply chain, IaC, SCA, secrets security, and AI-SPM—with third-party scanner data to accelerate secure development. Plus, it enables your developers to:

- Block issues early within developer-native tools.
- Prioritize business-critical risks.
- Automate remediation instantly.

See prevention-first application security in action.

REQUEST A CORTEX CLOUD DEMO

About Cortex Cloud

Cortex Cloud, the next generation of Prisma® Cloud, merges best-in-class CDR with industry-leading CNAPP for real-time cloud security. Harness the power of AI and automation to prioritize risks with runtime context, enable remediation at scale, and stop attacks as they occur. Bring together your cloud and SOC on the unified Cortex platform to transform end-to-end operations. Experience the future of real-time cloud security at www.paloaltonetworks.com/cortex/cloud.

paloalto® NETWORKS

3000 Tannery Way
Santa Clara, CA 95054

Main: **+1.408.753.4000**
Sales: **+1.866.320.4788**
Support: **+1.866.898.9087**

www.paloaltonetworks.com

cortex_eb_securing-ai-generated-code_011526