The Hidden Cost of AI Code in Financial Services

(AND HOW TO SPOT IT BEFORE IT HITS PRODUCTION)

Financial services organizations are under intense pressure to innovate.

- **70% of customers expect cutting-edge personalization** in their financial apps.
- Agentic AI coding tools are now embedded directly in developer workflows.
- Modern pipelines rely heavily on open source, containers, and infrastructure-as-code (IaC).
- Most organizations run multiple development pipelines across teams and business units.

AI helps teams ship faster. But faster code and the growing number of APIs, services, and secrets it creates can introduce compounding security risks without proactive security. The result? Rapidly increasing security debt across AI-generated code, third-party dependencies, and automated pipelines.

And the market is already shifting. By 2030, **50% of IT security spending is projected to go toward proactive, preemptive solutions.** The future of security goes beyond reactive triage, into prevention at developer speed.

WHERE THE HIDDEN COST OF AI CODE SHOWS UP

AI redistributes and amplifies risk. Here's where FinServ organizations are feeling the pressure:

01 INCREASING DEPENDENCY SPRAWL

Agentic coding assistants frequently recommend open source packages to accelerate development. But they don't always account for:

- Known vulnerabilities
- License risks
- Abandoned maintainers
- Transitive dependency exposure

Financial services environments already rely heavily on open source, and easily accessible tools like Claude Code, Cursor, and Windsurf multiply that consumption. Without automated, developer-native open source security, teams inherit risk faster than security teams can triage it, leaving the window open for AI-enhanced threat actors.

The hidden cost: An expanding attack surface buried inside your dependency graph.

02 AI-GENERATED CODE THAT LACKS SECURITY CONTEXT

Agentic coding assistants optimize for functionality and speed, not secure coding standards, compliance mandates, or regulatory frameworks.

The same way businesses scan code for vulnerabilities in their application systems, so too must they begin scanning code built within coding assistants.

That means:

- Insecure patterns can be replicated at scale.
- Sensitive data handling logic may lack guardrails.
- Security missteps get copied across repos.

In highly regulated FinServ environments where sensitive customer transaction data is handled, the risk is amplified.

The hidden cost: Security vulnerabilities introduced at creation rather than at deployment.

snyk

03 INFRASTRUCTURE MISCONFIGURATIONS AT SCALE

AI is also influencing how modern infrastructure is built. Instead of engineers scouring documentation for the correct syntax, AI models act as a bridge between high-level architectural goals and the specific YAML or HCL (HashiCorp Configuration Language) required to execute them. AI is increasingly used to generate:

- Terraform templates
- CloudFormation files
- Kubernetes manifests
- I/CD configurations

But AI doesn't understand your organization's cloud policies. A single misconfigured storage bucket, overly permissive role, or exposed container can create material financial and reputational damage.

The hidden cost: Cloud misconfigurations embedded directly into automated pipelines.

04 PIPELINE FRAGMENTATION AND LOSS OF VISIBILITY

Most financial institutions employ numerous development teams, each operating their own development pipelines across:

- Retail banking apps
- Trading platforms
- Payment systems
- Internal systems

AI experimentation increases repo proliferation and microservice growth, and the number of exposed APIs. Security teams lose centralized visibility while developers inherit more ownership.

The hidden cost: Risk scattered across pipelines, services, and APIs with no unified control plane.

THE INDUSTRY SHIFT: FROM REACTIVE TO PROACTIVE SECURITY

Traditional security models rely on:

- Post-development scanning
- Centralized AppSec bottlenecks
- Lengthy review cycles
- Reactive patching

But traditional models cannot keep up with agentic development. Financial services building with AI need to prioritize agent security, focusing on key areas such as:

- Detecting vulnerabilities before the merge
- Enforcing guardrails before deployment
- Preventing risky dependencies before the build
- Continuously monitoring in production

WHAT PROACTIVE AGENT SECURITY LOOKS LIKE

In practice, a FinServ organization keeping pace with agentic development looks like:

- → **Securing code at the point of creation:** proactively prevent new AI-generated vulnerabilities by embedding scanning directly within your agentic development tools, as well as embedding SAST and SCA directly into IDEs and pull requests.
- → **Automated governance:** Deploy consistent security guardrails to every developer, gaining the visibility and control needed to prevent new vulnerabilities at enterprise scale.
- → **Securing containers pre-deployment:** Find and fix misconfigurations in your infrastructure as code before they reach production.
- → **Test running applications and APIs continuously:** Use DAST to identify exploitable vulnerabilities in live web apps and APIs, including those powering banking, payment, and partner integrations.
- → **Reducing noise, prioritizing impact:** Focus developers on the vulnerabilities that matter most.
- → **Empowering developers, not slowing them down:** Security must work with pipelines, not against them.

Explore how financial institutions are operating at AI velocity while maintaining continuous regulatory control, and what it takes to shift from reactive security to AI-scale governance. **Download our whitepaper**, "Operating at AI Velocity in Financial Services While Maintaining Regulatory Control".

snyk