Substack Archive

Table of Contents

Code security for software engineers

Nover

Stream the latest episode

Listen and watch now on YouTube, Spotify, and Apple. See the episode transcript at the top of this page, and timestamps for the episode at the bottom.

Brought to You by

?? **Statsig** ? - ? The unified platform for flags, analytics, experiments, and more. Statsig are helping make the first-ever Pragmatic Summit a reality. Join me and 400 other top engineers and leaders on 11 February, in San Francisco for a special one-day event. Reserve your spot here.

?? **Linear** ? - ? The system for modern product development. Engineering teams today move much faster, thanks to AI. Because of this, coordination increasingly becomes a problem. This is where Linear helps fast-moving teams stay focused. Check out Linear.

-

In this episode

As software engineers, what should we know about writing secure code?

Johannes Dahse is the VP of Code Security at Sonar and a security expert with 20 years of industry experience. In today's episode of *The Pragmatic Engineer,* he joins me to talk about what security teams actually do, what developers should own, and where real-world risk enters modern codebases.

We cover dependency risk, software composition analysis, CVEs, dynamic testing, and how everyday development practices affect security outcomes. Johannes also explains where AI meaningfully helps, where it introduces new failure modes, and why understanding the code you write and ship remains the most reliable defense.

If you build and ship software, this episode is a practical guide to thinking about code security under real-world engineering constraints.

Interesting quotes from the episode:

How code quality and code security are connected:

Gergely: "How does the quality of code relate to security?"

Johannes: "This is an underrated topic in the industry today. We talked about the null pointer exceptions or these slow regular expressions, right? That can lead to security issues. That's more of the obvious examples of bugs.

Think about unreadable code, not well-maintained code: that's often like spaghetti code. At first, it's not so obvious how this is connected to security.

If you think about how code is not easy to comprehend, not easy to review. Then you do pair programming or code reviews in your development team - then in that spaghetti code, you will more likely overlook security problems of your colleague.

Now, maybe someone found an issue or and issue and reports it back to you. As a developer, you have to fix it. But if it's not maintainable code, you cannot fix the security problem easily. So quality suddenly becomes a security issue in how the attacker window stays open longer!

Your code quality is super related to code security, especially now with AI-generated code. We typically see poor quality of code here. And that becomes a problem for security."

How AI is introducing new security issues:

Gergely: "How do you think AI is changing code security?"

Johannes: "We see a change in how code and applications are built. Traditionally, you had this backend/frontend split.

In the backend, you had a database. Remove that database, then you don't have a SQL injection risk anymore.

As soon as you add an LLM to the backend, you have to deal with prompt injection vulnerabilities. The attacker can modify the system prompt or manage do some prompt engineering and then mess with the LLM's logic or the output. It's taking time both for the attackers to adjust to this, and the industry to adjust in defending against it."

Gergely: "What about with coding assistance? Are you seeing things change in terms of how we think about code security?"

Johannes: "**The big problem [with AI coding assistance] in terms of security is that you produce code much more faster.** Writing code is not the challenge anymore. Suddenly, the new bottleneck is how are you verifying all that code, right? And if you don't verify it:that leads to security issues or quality issues. Quality issues, on the long run, lead to security problems."

On software composition analysis:

Gergely: **"What is software composition analysis (SCA)?"**

Johannes: "It's a technique where we look at manifest files and your list of dependencies, depending on the package manager you use. This list of dependencies is checked against a database of known security problems. Those are called the CVEs. Then you can - with software composition - map, for example, that this specific Log4j version of your library is vulnerable to the Log4 shell vulnerability that is known. And then it can warn you."

The Pragmatic Engineer deepdives relevant for this episode

What is Security Engineering?

Mishandled security vulnerability in Next.js

Okta Schooled on Its Security Practices

Timestamps

References

Where to find Johannes Dahse:

? LinkedIn: https://www.linkedin.com/in/johannes-dahse-112b3057

Mentions during the episode:

? Sonar: https://www.sonarsource.com

? State of Code Security Report by Sonar https://www.sonarsource.com/resources/the-state-of-code-security-report/

? OWASP Top Ten: https://owasp.org/www-project-top-ten

? Software Composition Analysis: https://en.wikipedia.org/wiki/Software_composition_analysis

? CVE Program: https://www.cve.org

? SAST: https://en.wikipedia.org/wiki/Static_application_security_testing

? What is DAST: https://github.com/resources/articles/what-is-dast

? Stack Overflow AI survey: https://survey.stackoverflow.co/2025/ai

? Go: https://go.dev

? Java: https://www.java.com

-

Production and marketing by Pen Name.