Problem Solving: Getting Security and Development on the Same Page

Emmanuel Dade | Application Security Internship, Summer 2026

Overview

During my internship this summer I was working with the application security team on a summer intern project to automate work within Jira using an AI assistant. My specific focus was on security. I authored security documentation, threat modeling, and abuse case scenarios, and also assisted in integrating security tooling into the CI/CD pipeline.

One of the first problems that we encountered was not related to code. It was about getting security and development to work together such that both sides benefit from the collaboration. We achieved this by having consistent meetings, providing clear explanations of security findings and the reasoning behind them, and walking through code in both directions to ensure everyone was on the same page.

The Project and My Role

The project was a summer intern project led by the application security team. The goal was to automate an AI assistant within Jira. I was responsible for the security aspects of the project, and collaborated with the developers who implemented the functionality as well as the quality engineers who tested the solution.

My specific security tasks were:

Work	What it is
Security documentation	Documentation outlining the security requirements and decisions made in the project
Threat models	Structured analysis of possible issues that can occur within the system as well as potential attack vectors
Abuse case scenarios	Descriptions of how the system could be intentionally misused written from the perspective of an attacker as a user story
CI/CD security tooling	Assistance in integrating security checks into the CI/CD pipeline the code goes through

The Problem

Security and development have different goals. Development focuses on delivering useful features, and security aims to identify potential issues. If these two groups only communicate at handoffs then security can be viewed as a checkpoint rather than a partner.

Instead we sought to avoid this situation. We wanted to work hand in hand with both the developers and quality engineers in such a way that both security and development would benefit. For development, we wanted security findings that they could understand and use, and for security we wanted decisions that were informed by how the system was actually built.

How I Approached It

Rather than assuming that collaboration would happen organically, we made it a part of how we worked together. Three practices were particularly effective at enabling collaboration:

1. **Consistent meetings.** We held regular meetings instead of only meeting when an issue arose. This allowed issues to be identified sooner and ensured no one was surprised by a security finding at the end of a meeting.
2. **Explaining findings and the reasoning behind them.** I explained what a security finding was, why it mattered, and how I reached my finding so that the developers could make an informed decision on how to address the finding instead of being given a task.
3. **Walking through code together.** We walked through our code in both directions. I explained the security tooling and reasoning behind it and the developers explained how the system was built.

What all of these practices had in common was that everyone had access to the same information. By keeping each side's work visible to the other we maintained alignment at all times.

Putting It Into Practice

Each of my security tasks was shaped by this approach to collaboration:

- **Threat models and abuse cases.** These both require an understanding of how the system is actually built, and walking through the code provided the context needed to ask better questions about potential attack and misuse vectors.
- **Security documentation.** By explaining the reasoning behind the security decisions and not just the decision itself, the developers and quality engineers were able to gain access to the same information that was shared during our meetings.
- **CI/CD security tooling.** Integrating security checks into the CI/CD pipeline impacts everyone who uses that pipeline for code delivery. Collaborating on those changes with the developers ensured that the tooling would fit within how the team currently works. The CI/CD pipelines used included Azure DevOps, GitHub Actions, and GitLab CI/CD.

Result

Security and development ended up working together as one team rather than two. Both sides benefited from this collaboration. I gained an understanding of how the system was built and the developers and quality engineers gained an understanding of the reasoning behind the security-related

tasks. The security documentation, threat models, abuse cases and pipeline tooling all came from this collaboration.

What I Learned

Not all problems are bugs. In this case the problem was about collaboration and I approached it the same way I would approach a technical problem:

Step	What I did
Define the problem	Identified the actual problem: how security and development can work together
Choose an approach	Regular meetings, clear explanations of findings and reasoning behind them, and jointly walking through code
Keep it going	Repeated this approach consistently to keep the team aligned
Learn from it	Both sides benefited from the collaboration

My key takeaway is that security work is successful when the people impacted by the work understand it. Explaining the reasoning behind a finding is as important as actually finding it and is a skill I plan to continue developing as I move into a career in application security.