Developer Perspectives on Information-Flow Security Threats and Tooling

Twain Byrnes
Carnegie Mellon University

Pragya Chaubey
Carnegie Mellon University

Ana Fontan
Carnegie Mellon University

Nisarga Gondi
Carnegie Mellon University

McKenna McCall
Colorado State University

Limin Jia
Carnegie Mellon University

1 Introduction

Security vulnerabilities often find their way into to production code, causing various undesirable ramifications. According to IBM, the global average cost of a data breach was $4.44 million in 2025, with an average cost of $4.92 million for malicious insider attacks [6]. It has been estimated that almost a third of data breach victims experience identity theft [4].

Academics and industry professionals have created security tools that aim to find potential security vulnerabilities before they are ever put into production code. One type of tool prevents sensitive (or untrusted) information from flowing to unauthorized (or trusted) locations through information-flow control [1–3, 5, 7, 8]. While these tools can enforce strong security properties, they often leverage unwieldy type systems and are cumbersome to use. Unfortunately, industry adoption of information-flow control security tools remains low.

The goal of this work is to learn more about developer experiences with information-flow threats and desirable security properties so that we can design tools that meet developers where they are. We answer the following research questions:

RQ1. What attackers are most compelling to developers?

RQ2. How do developers perceive information-flow threats? (e.g., SQLi, XSS)

RQ3. What security policies regarding confidentiality and integrity are most aligned with developer goals, and what would support adoption?

This project is a work in progress. We present the results of a small-scale study that will be expanded in the future.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2026.
August 23–26, 2026, Hannover, Germany.

2 Methods

We created a Qualtrics survey and recruited Carnegie Mellon University students, staff, and faculty with at least one year of development experience. Participants were entered into a random drawing for $20 after completing the screening survey and were given $5 for completing the main survey. We received 23 screening survey responses and 21 main survey responses. We evaluated main survey responses to ensure that the survey was taken by a human and not an AI agent, and that free response questions were answered by a person and not a large language model. We also ensured that strategies such as straight-lining were not used by participants, and that they engaged with the survey for long enough to have meaningful answers. After this screening, we had 17 valid completions of the main survey.

Our study was approved by the Carnegie Mellon University IRB. Before the study, participants provided their informed consent. Participant emails were collected to allow for compensation and confimation of Carnegie Mellon University affiliation. Emails were removed prior to analyzing data.

3 Results

RQ1: Attacker Threat Models. When asked to select the single most important attacker to safeguard against, responses were fairly distributed: anonymous external attackers and authenticated users abusing privileges each received 5 selections (29%), authenticated users maliciously using legitimate privileges received 4 (24%), and other developers or maintainers received 2 (12%). Overall, 65% of respondents (11 out of 17) selected some form of insider threat—an authenticated user or internal developer—as the most important attacker to defend against.

RQ2: Threat Perception. When asked which attacks they would address first if responsible for securing the system (select up to two), Auth Failures/BAC and SQL Injection/XXE

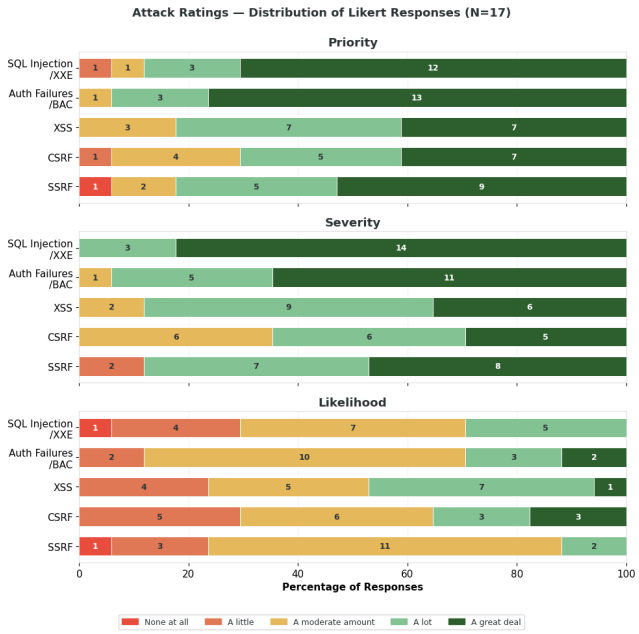

Figure 1: Distribution of priority, severity, and likelihood ratings across five attack categories.

dominated with 12 and 9 selections respectively. The "address last" question showed the inverse pattern: CSRF received 12 selections and SSRF received 7, confirming a clear prioritization hierarchy that aligns with the Likert ratings.

We found that developers tended to have similar priority and perceived severity for all attacks except SQL injection (and XXF), which was consistently rated high priority by nearly all respondents regardless of severity assessment. This suggests that SQL injection as an attack vector may benefit from a "reputation effect"—developers have sufficient prior exposure to this attack that they assign it high priority regardless of their perceived severity.

In an open-ended response question, we asked developers to identify the most vulnerable components in a system and most important to protect. Interestingly, there is a gap between perceived vulnerability and protection priority. The component viewed as most vulnerable was APIs, with seven participants. However, twelve participants viewed login systems as the most important to protect. This suggests that developers may over-invest attention in well-understood, user-facing entry points (such as login pages) while underestimating the risk posed by API-level data flows—which are often more complex and less visible in the development workflow. This is interesting to contrast with the results of RQ1, which suggest that developers consider insiders to be the most important to safeguard against: 9 of the 17 responses believed legitimately authenticated users to be the most important to safeguard against, which does not align with prioritizng protection of

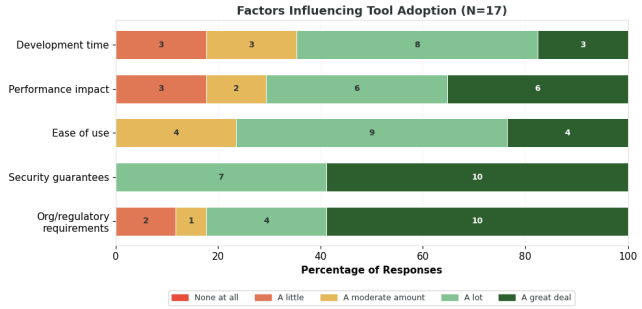

Figure 2: Distribution of influence ratings for factors affecting security tool adoption.

login systems.

RQ3: Security Goals and Tool Adoption. We provided participants with potential security goals and asked them to evaluate how important the goals were to acheive. All of the goals were enforceable using information-flow control techniques, and all were highly rated, with no goal significantly higher rated than any other.

Most respondents were willing to invest meaningful time in learning and using security tools. Almost half of respondents were willing to invest either 4-8 hours or over a week to learn a security tool, and almost half were willing to spend an additional 1-4 hours per week on the tool. Two respondents indicated they would only adopt a tool if required by regulation or organizational policy.

Fourteen respondents said they would accept runtime cost in exchange for strong security guarantees, with median acceptable overhead 15%, mean = 23.4%, and range = 2–50%.

4 Conclusion and Next Steps

Conclusion. Through an observational study of 17 developers, we found that, while developers recognize well-known vulnerabilities such as SQL injection as high priority regardless of severity, they may under-prioritize equally critical, less publicized threats. They tend to view insider threats as the most critical risk. There is a disconnect between components believed to be most vulnerable (APIs) and components prioritized for protection (login systems). Participants expressed a strong willingness to adopt tools that offer powerful security guaranties, even those requiring significant learning time and runtime performance overhead.

Next Steps. We will recruit participants from Prolific and UpWork to obtain a larger body of participants that is more representative of developers. On this larger, representative sample, we will be able to complete a statistical analysis.

References

[1] Vincent Beardsley, Chris Xiong, Ada Lamba, and Michael D. Bond. Carapace: Static-dynamic information flow control in rust. *Proc. ACM Program. Lang.*, 9(OOPSLA1), April 2025.

[2] Niklas Broberg, Bart van Delft, and David Sands. Paragon–practical programming with information flow control. *Journal of Computer Security*, 25(4-5):323–365, 2017.

[3] Pablo Buiras, Dimitrios Vytiniotis, and Alejandro Russo. Hlio: mixing static and dynamic typing for information-flow control in haskell. *SIGPLAN Not.*, 50(9):289–301, August 2015.

[4] John W Coffey. Difficulties in determining data breach impacts. *Systemics, Cybernetics and Informatics*, 17(5):9–13, 2019.

[5] Willem De Groef, Dominique Devriese, Nick Nikiforakis, and Frank Piessens. Flowfox: a web browser with flexible and precise information flow control. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 748–759, New York, NY, USA, 2012. Association for Computing Machinery.

[6] IBM Security and Ponemon Institute. Cost of a data breach report 2025: The ai oversight gap. Technical report, IBM Corporation, 2025.

[7] Andrew C. Myers. Jflow: practical mostly-static information flow control. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '99, page 228–241, New York, NY, USA, 1999. Association for Computing Machinery.

[8] Marco Vassena, Alejandro Russo, Pablo Buiras, and Lucas Waye. Mac a verified static information-flow control library. *Journal of Logical and Algebraic Methods in Programming*, 95:148–180, 2018.