API Blindspots: Why Experienced Developers Write Vulnerable Code

Muhammad Sajidur Rahman, Tian Lin, Rad Akefirda, Donovan Ellis, Eliany Perez, Lois Anne DeLong, Justin Cappos, Yuri Brun, Natalie Ebner, Daniela Oliveira

University of Florida New York University University of Massachusetts Amherst Auto1 Inc

Background
- New instances of existing, well studied vulnerabilities, such as SQL injections and buffer overflows, are frequently reported in vulnerability databases.
- 61% of vulnerabilities reported in OWASP Top 10 vulnerability categories (OWASP, 2016).
- 66% of vulnerabilities reported in OWASP Top 10 vulnerability categories (OWASP, 2016).
- Developers often ‘blindly’ trust and use programming language APIs as if they are outsourcing security implications to the APIs.
- API security blindspots: A misconception, misunderstanding, or oversight on the part of the developer when using an API function, which leads to a violation of the recommended API usage protocol with possible introduction of security vulnerabilities.

Methods
Sample:
- n = 109 Java Developers (age range: 21-52 years, 80.7% male).
- 64.2% participants were professional developers, rest were senior/graduate students in Computer Science and Engineering.

Study Instrument:
- Code review task: 24 programming puzzles, two-third had API blindspots, one-third had no blindspots in API usage. All puzzles were functional and error free.
- Professional experience assessment: A self-reported assessment on 17 programming concepts and technologies.
- Personality assessment: Big Five Inventory questionnaire.

Study Procedure:
- Participants were asked to solve a set of six puzzles which were counterbalanced by blindspot and non-blindspot APIs, types of API usage contexts and cyclomatic complexity.
- Personality assessment was done on a 5-point Likert scale.
- A JavaScript plugin recorded audio responses during cognitive assessment.
- Upon completion, participants were presented with the solutions and explanations.

Data Analysis
Hypothesis 1. Developer’s accuracy in API Blindspot contexts
- Multi-level logistic regression
- Significant API Blindspot effect (B = −.81, z = −6.54, p < .001, odds ratio = .44)
- Significant Blindspot x API category interaction (χ² = 24.8, p < .001)

Hypothesis 2. Developer perception to detect API Blindspots
- Multi-level logistic regression
- No significant effect found

Hypothesis 3. Cognitive functioning to detect API Blindspots
- Ordinal logistic regression
- No significant effect found

Hypothesis 4. Technical expertise to detect API Blindspots
- Ordinal logistic regression
- No significant effect found

Hypothesis 5. Personality traits to detect API Blindspots
- Ordinal logistic regression
- Openness and extraversion found to be significant (p < .05)

Results
1. Our results confirmed HI that developers were less likely to correctly solve puzzles with API blindspots compared to non-blindspot puzzles. This finding suggests that developers experienced security blindspots while using certain API functions.
2. This effect was more pronounced for puzzles with I/O-related API functions and when the programming scenario was more complex (i.e., high cyclomatic complexity).
3. Our data did not support that developers’ perceptions of puzzle clarity, confidence, difficulty, and familiarity were associated with their ability to detect blindspots. Our results also did not support that developers’ level of cognitive functioning predicted their ability to detect blindspots.
4. Our data also did not support that professional and technical experience was associated with developers’ ability to detect blindspots.
5. Our results partially supported that more openness and higher extraversion as personality traits in developers were associated with higher likelihood to detect blindspots.

Discussion
- Our data supports the notion that blindspots in API functions lead to the introduction of vulnerabilities in software, even for experienced developers.
- Given these findings, API designers should consider addressing developers’ misconceptions and flawed assumptions when working with APIs to increase code security.
- Software Security training and tools should not come as a “one-size-fits-all”, but consider developer’s decision making process and possible blindspots.
- Future Directions: Explore and rank more variants of API blindspots in code repositories and develop detection and recommender tool for developers to write more secure code.

Acknowledgements: Research supported by the National Science Foundation under grant 1513572.