Write, Read, or Fix? Exploring Alternative Methods for Secure Development Studies

Kelsey R. Fulton* (Colorado School of Mines), Joseph Lewis (University of Maryland), Nathan Malkin* (New Jersey Institute of Technology), and Michelle L. Mazurek (University of Maryland)

* - Work performed while at University of Maryland
Developers struggle with security

- NVD reported 28,831 vulnerabilities in 2023 [1]
  - 25,081 in 2022
- Often caused by developers:
  - Making mistakes
  - Misunderstanding security
- Addressing this requires understanding it
  - Studying developers as they build code

How do we study developers?

• Interview studies
• Surveys
• Code writing studies
Challenges with code writing tasks

- Code writing is time consuming
- Tasks are difficult to scope
- It is hard to effectively design studies
- Developers are hard to recruit and retain:
  - Hard to find
  - Participate outside of work hours
  - Participate for less money than they are paid at work

Dropouts
Samples

Are there alternative approaches that will yield similar results while reducing stress?
Using code review

• In 2021, Danilova et al. explored the use of code review [1]
• Participants wrote code reviews about snippets from a prior study
• Code review is potentially useful in place of long programming tasks
  • Able to identify issue developers faced

Expand on this by directly comparing a Read and Fix condition

Write, Read, and Fix

**Write**
- Write code to complete spec
- Provided tests

**Read**
- Read completed code
- Identify any bugs/vulns
- Describe fixes
- Do not actually alter code
- Cannot run code

**Fix**
- Read completed code
- Identify any bugs/vulns
- **Fix** bugs/vulns
- Provided tests
Research questions

- Do the Read and Fix conditions provide the same results as Write?
  - Functionality and security

- Do participants in Read and Fix experience fewer negative effects?
  - Drop-out rate
  - Frustration
  - Time spent
Study design

- Partially replicated prior study [1]
- Participants completed self-contained, short Write tasks
- Utilized 1 of 5 Python libraries
- Tasks were focused on (a)symmetric encryption
- Allowed us to compare our Write results
- While allowing us to compare Write, Read, Fix
Study flow

Consent → Condition assignment → Tasks → Final survey

- Write
- PyCrypto
- Encrypt/decrypt data
- Performance on tasks
- Read
- Crypto.io
- Generate and store a key
- Frustration and fun
- Fix
- Background

Write, Read, or Fix?

Methods
Data analysis

• Manually reviewed code for bugs/vulnerabilities
  • Leveraging the vulns/bugs from [1] and our known list
• To compare results among conditions:
  • Ran various regressions for impact of library and condition
Recruitment and participants

- Recruited 112 valid participants from Upwork and CS student mailing lists
  - **Write**: 35 participants
  - **Read**: 37 participants
  - **Fix**: 40 participants
- Our participants were fairly experienced, but not in security:
  - Avg 6.8 years programming experience
  - Avg 4 years Python experience
  - Avg 1.2 years security experience
Research questions

• Do the *Read* and *Fix* conditions provide the same results as *Write*?
  • Functionality and security

• Do participants in *Read* and *Fix* experience fewer negative effects?
  • Drop-out rate
  • Frustration
  • Time spent
Takeaway #1: Use Write to measure the efficacy of code writing tools

- **Write** was able to reveal important differences between crypto APIs
  - Specifically, in the security of solutions participants produced
- Also revealed documentation issues
- These differences were substantially less visible in **Read** and **Fix**
- Security APIs are designed to prevent developers from making security mistakes
  - Rather than identifying or fixing them
Takeaway #2: Use Read to measure developers’ knowledge

- *Read* participants pay close attention to the code
  - Identified fewer, but more diverse bugs than *Fix* participants
  - Identified more vulns than *Fix*, even identifying 8 out-of-scope vulns
- Making *Read* useful for identifying overall security awareness and knowledge
Takeaway #3: Use Fix to measure quick fixes

- *Fix* participants heavily focused on passing provided tests
  - All of our *Fix* participants started by running the code
  - Causing them to miss bugs and vulnerabilities
- *Fix* may be useful for identifying vulns and bugs developers can quickly find
  - Offer lower bound on their abilities
Takeaway #4: Use *Read* and *Fix* to minimize time, frustration

- *Read* and *Fix* participants spent less time than *Write* participants
  - And had fewer dropouts
- *Read* and *Fix* participants actually enjoyed their tasks
- *Read* and *Fix* may offer an appropriate option when recruitment is a concern
• We explored two alternatives *(Read and Fix)* to code writing studies *(Write)*

*Write* more clearly identifies security differences between security APIs

*Read* participants paid close attention to the code

*Fix* participants focused on passing tests, missing key vulns

Participants felt fewer negative effects *(frustration, time spent)* in *Read* and *Fix*

• Possibly helping in retention and recruitment