Trusted CI Webinar Series

**Title:** Investigating Secure Development In Practice: A Human-Centered Perspective

**Presenter:** Michelle Mazurek, University of Maryland

**Host:** Jeannette Dopheide  

**Slides:** https://tinyurl.com/yvs3kzn8

The meeting will begin shortly.

Participants are muted. Click the chat button to ask a question.

**This meeting will be recorded.**

The Trusted CI Webinar Series is supported by National Science Foundation grant #1920430.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.
2021 Trusted CI Annual Challenge on Software Assurance

**Goal:**
Improve robustness of software used in scientific computing with respect to security.

**Approach:**
(1) Engaging with developers of scientific software to understand the range of software development practices being used, and identify opportunities to improve practices and code implementation to minimize the risk of vulnerabilities.

(2) Developing a guide specifically aimed at the scientific software community that covers software assurance in a way most appropriate to that community.

*We will be looking for community reviewers of the guide draft by mid-fall 2021.*
INVESTIGATING SECURE DEVELOPMENT IN PRACTICE: A HUMAN-CENTERED PERSPECTIVE

Michelle Mazurek
University of Maryland
Why are users foolish or lazy?

How can we make security more usable?
OTHER PEOPLE IN THE SECURITY LOOP

- Developers
- Testers
- Sysadmins
BUT THESE ARE EXPERTS, RIGHT?

Or not.

```c
if ((err = SSLHas) goto fail;
goto fail;
if ((err = SSLHas) goto fail;
```
Why are developers foolish or lazy?

How can we make secure programming easier?
SOUNDS GREAT, BUT WHAT DOES THAT MEAN?

- Studying developers
  - Motivations, decision making, etc.

- Measuring tools and interventions
  - APIs, bug finding tools, educational materials, etc.
  - Existing and new

- Goal: Build better tools and interventions
TODAY: HUMANS IN SECURE DEVELOPMENT

- Stack overflow vs. security
- Measuring crypto libraries
- Examining security mistakes with Build it, Break it, Fix it
- Rust as a case study for secure-language adoption
YOU GET WHERE YOU’RE LOOKING FOR
THE IMPACT OF INFORMATION RESOURCES ON CODE SECURITY

HAS THIS HAPPENED TO YOU?
Answer suggests to trust all certs: Uh oh
Many real apps do this! [Fahl+ 2012]
  Some interviewees: pasted from internet

Can we measure this empirically?
How does it contrast with official docs?
A LAB STUDY OF INFORMATION RESOURCES

- Four short Android programming tasks
  - Designed to have secure/insecure solutions

- Resources constrained by condition:
  - Official docs, Stack Overflow, book, free choice

- 54 participants; 14 professional
  - Required to pass short Android quiz
SKELETON APP, EMULATOR
Correctness: Does it compile and work?

Security: If it works, was solution secure?
  - Coded manually in predefined categories
FUNCTIONAL CORRECTNESS

Official significantly worse than Stack Overflow or Book!
BUT HOW ABOUT SECURITY?

% Correct, Among Functional Only

- Official: 70%
- Stack Overflow: 50%
- Book: 70%
- Free Choice: 65%

Stack Overflow worst; Official best!
A CLOSER LOOK AT STACK OVERFLOW

- Collected via browser history
- 149 unique pages, 41 relevant
- 20 with code snippets
  - 7 only secure, 10 only insecure, 3 both
  - 3 insecure have warnings
STACK OVERFLOW: **quick, functional** solutions
- Official docs don’t

But, **less secure** than official or books

- We need resources that integrate both!
  - Add a security rating to influence upvote?
  - Modify official docs to target troubleshooting?
  - Use SO to identify trouble spots, provide code snippets in the official docs?
COMPARING CRYPTOGRAPHIC APIs

GETTING CRYPTO RIGHT IS HARD

Cryptography failure leads to easy hacking for PlayStation Classic

Plug-and-play hack

KYLE ORLAND - 12/10/2019

"... hackers found that the most sensitive parts of the system are signed and encrypted solely using a key that's embedded on the device itself, rather than with the aid of a private key held exclusively by Sony."
Simplified alternatives claim to be more usable
(fewer functions, fewer parameters)

Is this really true?
Online study comparing crypto APIs

- Short python tasks with secure and insecure solutions
- Two traditional and three simplified libraries
- 256 participants recruited from GitHub
  - 208 professionals, 198 with no security background
  - 1571 consented; many dropped out
Certificate validation

**Goal:** Verify that the SSL certificate from the central Citizen Measure server was issued by the Let's Encrypt Certificate Authority to ensure that citizen reports are not being intercepted. You have to validate the certificate’s digital signature and common name. For your convenience, the SSL certificate from the Citizen Measure server is stored in `CitizenMeasureCertificate.pem` and the Let's Encrypt Certificate Authority certificate in `letsca.pem`. You can take a look at the Let's Encrypt [Root CA](https://letsencrypt.org) and the server certificate.

```python
In [0]:
1 import nacl
2
def validate(certificate, root_certificate, hostname="citizen-measure.tk"):
3     """
4     Purpose:
5     Validate the given certificate's digital signature and common name.
6
7     Arguments:
8         certificate: The certificate to validate.
9         hostname: The server's hostname.
10
11     Return value:
12     validation_result: True if validating the certificate is correct, False otherwise.
13
14     Notes:
15         - The Citizen Measure server certificate can be found at `CitizenMeasureCertificate.pem`
16         - The Let's Encrypt Certificate Authority certificate can be found at `letsca.pem`
17         - If you used any other information source to solve this task than the linked documentation (e.g. a post on StackOverFlow, a blog post or a discussion in a forum), please provide the link right below:
18         - additional information sources go here (e.g. https://stackoverflow.com/questions/415611/how-to-get-current-time-in-python)
19     """
20
21     # This is where your code goes
22     return False
23
24     # This is to test the code for this task.
25     certificate = open("CitizenMeasureCertificate.pem").read()
26     root_certificate = open("letsca.pem").read()
27     assert validate(certificate, root_certificate, "citizen-measure.tk"), "Certificate validation failed."
28     print "Task completed! Please continue."
```

Skeleton code, online code editor
Simplified not necessarily easier to use!
OPINIONS ABOUT LIBRARIES

- SUS: Nothing better than mediocre
  - Most disliked: keyczar, m2crypto
  - Very similar to functionality results

- Documentation is key!
  - Keyczar: “Your documentation is bad and you should feel bad.”
Simplified are more secure – if you can use them
SECURITY PROBLEMS

- In traditional libraries only:
  - Bad choice of parameters, combinations
  - Sometimes related to documentation examples
- In both: Not end-to-end (e.g., key generation)
TAKEAWAYS

- Implementing crypto is (still) hard
- Simplified APIs **do** promote security
  - Sort of!
- Documentation, full-featured-ness are key
UNDERSTANDING SECURITY MISTAKES DEVELOPERS MAKE

QUALITATIVE ANALYSIS FROM BUILD IT, BREAK IT, FIX IT

Votipka, Fulton, Parker, Hou, Mazurek and Hicks. USENIX Security 2020
LAB STUDIES ARE GREAT, BUT …

- Pros: Can control conditions, make easy comparisons.
- Cons: Simple problems, ecological validity.
- Field observations also have challenges
- Can we try another point in the tradeoff space?
BUILD IT, BREAK IT, FIX IT

- Secure development contest
- Build to spec
- Then break other teams
- Incentive design is important!

Ruef et al., CCS 2016
RESEARCH QUESTIONS

- What types of vulnerabilities do developers introduce?
- How severe are they?
- How exploitable are they?
ANALYSIS APPROACH

- Examine each project and vulnerability in detail
  - Breaker-identified and researcher-identified
- Iterative open and axial coding
  - Two independent coders; high reliability
- 94 projects, 182 unique vulnerabilities
Vulnerability Classes

- No Implementation
  - No attempt to ensure some security property
Vulnerability Classes

No Implementation

Missed something “Intuitive”
- No encryption
- No access control

Intuitive
Missed something “Unintuitive”
• No MAC
• Side channel leakage
• No replay prevention
Attempted security, but exhibits some fundamental misunderstanding

Vulnerability Classes

Misunderstanding
Made a “bad choice”
- Weak algorithms
- Homemade encryption
- strcpy()
Made a “conceptual error”
• Insufficient randomness
• Disabling default protections
Made a programming “mistake”
- Control flow error
- Skipped algorithmic step
Vulnerability Classes

- No Implementation (50%)
- Misunderstanding (56%)
- Mistake (21%)

Subclasses:
- Intuitive (16%)
- Unintuitive (45%)
- Bad Choice (21%)
- Conceptual Error (44%)
TAKEAWAYS

- Most understood they needed security
  - But struggled with how to implement it correctly
- Many error checking tools focus on mistakes
  - But we need more effort to solve conceptual problems
BENEFITS AND DRAWBACKS OF ADOPTING A SECURE PROGRAMMING LANGUAGE
RUST AS A CASE STUDY

Fulton, Chan, Votipka, Mazurek and Hicks.
MEMORY SAFETY BUGS: STILL A PROBLEM

- 70% of high+ vulns in Chrome (May 2020)
- 70% of Microsoft security updates (2002-2019)
- C/C++ source of most such bugs

Fix or replace (or both)

CASE STUDY: RUST

- Attempt to enable safety + performance (e.g., no GC)
  - Intended to be useful where C/C++ are hardest to replace
- What are the benefits and drawbacks?
- What does adoption look like?
CASE STUDY: RUST

- Interviews with senior developers (n=16)
- Survey of Rust community (n=178)
Most found security, performance benefits as advertised

Few, ad-hoc procedures to manage unsafe code
LEARNING RUST IS A CHALLENGE

- Mostly from curiosity, or because perceived as marketable
- Hard to learn, compared to other languages
  - \(\frac{3}{4}\) survey participants more than a week to get running code
- Specifically, requires paradigm shift
Positive Impacts on Development

- Improved confidence in code correctness
- Long-term productivity improvements
- Better mindset in other languages

“Once you learn Rust, you are one with the borrow checker – it never leaves you.”
Compared to another language,

- More confident it produces bug-free code
- Less time to debug
- Easier to maintain*
- More quickly design/implement
- Less time from start to ship
- Less time to Prototype*
- Less time to compile*

Strongly disagree | Slightly disagree | Neither agree nor disagree | Slightly agree | Strongly agree

% of Participants
ORGANIZATIONAL APPREHENSIONS

- Any language: avoid change, business pressures
- Steep learning curve: hit to productivity
- Maturity and long-term maintenance
- Difficulty hiring Rust devs

“It’s going to take a year, maybe three years, to develop the things, and they care where the ecosystem will be at that point.”
ENCOURAGING ADOPTION

- Start small and pick projects carefully
- Budget for up-front costs
- Support developers through the transition

“Make yourself an expert ... and share your expertise generously.”
TAKEAWAYS

- Documentation, community and feedback matter a lot!
  - Still a need to improve culture around unsafe

- Steep learning curve goes poorly with immaturity
  - Pay now and benefit later … if there is a later
  - Flatten learning curve?
  - Create pipeline, models, templates, libraries?
Human and organizational factors matter!
Empirical measurement helps
  - Interrogate conventional wisdom; identify gaps

Key themes so far:
  - Documentation matters
  - Full-featuredness, abstraction levels matter
  - Concepts, not just mistakes
  - Deferred benefits cause problems

mmazurek@umd.edu
@mmazurek_
ACKNOWLEDGEMENTS

- Fantastic students/collaborators ([sp2.umiacs.io](http://sp2.umiacs.io))
- All of our participants
- Funding from NIST and NSF
USABLE SECURITY FOR DEVELOPERS, TOO

- Human and organizational factors matter!
- Empirical measurement helps
  - Interrogate conventional wisdom; identify gaps

- Key themes so far:
  - Documentation matters
  - Full-featuredness, abstraction levels matter
  - Concepts, not just mistakes
  - Deferred benefits cause problems

mmazurek@umd.edu
@mmazurek_
Questions?

Click on the chat icon to type a question
Community Updates

- See you at PEARC21! July 19-22
  pearc.acm.org
  Trusted CI workshop & Security Log Analysis -- Both on Monday 11am EDT
- Trusted CI Webinar: Mon July 26th at 11am Eastern
  Topic: Open Science Grid
  Presenter: Brian Bockelman, Morgridge Institute for Research
- Trusted CI podcast version of webinar available on Apple and Google