YOU ARE NOT YOUR DEVELOPER, EITHER
INVESTIGATING WHY DEVELOPERS MAKE SECURITY MISTAKES

Michelle Mazurek
Why are users not stupid or lazy?

How can we make security more usable?
Beyond end users for more impact

Accessibility

End Users (> 1.5 billion)

Developers (~350,000)

System Designers (Google)

Impact

Example: Android
What about software developers?

Developers are experts, right? Or not.
Why are developers stupid or lazy? How can we make secure programming easier?
Lessons learned: Usec for end users

- You are not your user
- Security is a secondary concern
- More is not always better
You are not your user

- Confusing warnings and error messages
- Too much security jargon
- Don’t assume security knowledge just because they know how to program
- Design for usability, evaluate it explicitly
Security is secondary

• No one turns on their computer to do “security”
  • Functionality, time to market, maintainability, etc.
  • May (appear to) conflict with security
• Attention and time are limited!
• Try: Take developer out
• Try: Persuasive design

https://www.scripps.org/sparkle-assets/images/multitasking-600x375.jpg
More is not always better

• Too much advice is overwhelming
  • Hard to roll it back
• Can’t just keep asking users (developers) to do and remember more stuff
Research: Beyond end users

• Understanding developers
  • Example: Stack Overflow vs. security
  • Example: Build it, break it, fix it

• Measuring the status quo
  • Example: Measuring crypto libraries

• Methodology and validity

IEEE SecDev 2016 for more
The impact of information resources on code security

IEEE S&P 2016
Has this happened to you?
That doesn’t seem right ....

• Answer suggests to trust all certs
  • Many real apps [Fahl+ 2012]

• Some interviewees: pasted from internet
Stack Overflow considered insecure

• “Everyone knows” copy-paste from the internet is bad for security
  • Particularly for “amateur” app devs?

• Can we measure this empirically?
• How does it contrast with official docs?
• What do real devs do?
Online developer survey

• Sent 50k invites, collected from Play
  • 295 valid responses

• Strategy for help with security/permissions

• General use of programming resources
Where do you look up ...

- General
  - Books: 69%
  - Official Android docs: 62%
  - Search engines: 27.5%
  - Stack Overflow: 69%

- Encryption
  - Books: 69%
  - Official Android docs: 62%
  - Search engines: 27.5%
  - Stack Overflow: 69%

- HTTPS
  - Books: 69%
  - Official Android docs: 62%
  - Search engines: 27.5%
  - Stack Overflow: 69%

- Permissions
  - Books: 69%
  - Official Android docs: 62%
  - Search engines: 27.5%
  - Stack Overflow: 69%

69% Stack overflow, 62% search engines, 27.5% official
Next, a lab study

• Complete four short programming tasks
  • Designed to have secure/insecure solutions
• Resources constrained by condition:
  • Official docs, Stack Overflow, book, free choice
• Exit interview

• Not primed for security or privacy!
Skeleton app, emulator
Task 1: Secure networking

- Convert HTTP to HTTPS
  - In presence of X.509 cert error

- Sample secure solution:
  - Accept only this cert

- Sample insecure solution:
  - Accept all certs

http://5zin.com/certificate-of-authenticity-template.html
Task 2: Inter-component comms

- Given a service, limit access to only apps from same developer

- Sample secure solution:
  - Define a “signature” permission

- Sample insecure solution:
  - Export publicly
Task 3: Secure storage

• Store user ID and password locally

• Sample secure solution:
  • Private shared preference

• Sample insecure solution:
  • Public on SD card

http://www.routercheck.com/administrator-password/
Task 4: Least permissions

- Dial a customer-support phone number

- Sample secure solution:
  - Dial but don’t call

- Sample insecure solution:
  - Call (extra permission)

Evaluation

• Correctness: Does it compile and work?

• Security: If it works, was solution secure?
  • Coded manually in predefined categories

• Self-reported sentiment
  • Security thinking
  • Correctness and usefulness of resources
Participants

- 54 total
- 13 or 14 per condition
- 12 U.S., 42 Germany
- Ages 18-40; median 25
- 46 men, 8 women
- 14 professional, 40 non-professional
- Required to pass basic Android quiz
Resource was easy to use

Free choice was easiest; book was worst
Resource was correct

Books, official docs considered most correct
Security thinking

• Observed via think-aloud:
  • 16% thought about it
  • 5% said they ignored it for study / time

• Self-reported: 60% thought about it

• No significant difference in conditions
Functional correctness

- SO (67%) and Book (66%) performed best
- Official (40%) performed worst
  - Significantly worse than SO
But what about security?

SO worst (51%), Official best (86%) (significant)
Security by task

- Storage: 100% of functional solutions secure
- Networking: Only 39%
Professionals vs. students

- More functional
- But not significantly more secure!
Lookup behavior

- Official: scrolling, clicking internal links
- Stack Overflow: many search resets
- Free choice:
  - Everyone used official, all but one used SO
  - One picked up a book!
  - Results closest to SO
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
So now what?

- If you want functional, secure code:
- Cut off the internet, give your devs a book!
Real takeaways

• Stack Overflow: **quick, functional** solutions
  • Official docs don’t
• But, it’s **less secure** than official or books

• We need resources that integrate both!
  • Add a security rating to influence upvote?
  • Integrate Q&A into official docs?
  • Use SO to identify trouble spots, provide code snippets in the official docs?
Comparing Cryptographic APIs

IEEE S&P 2017
Getting crypto right is hard

• Developers must pick:
  • algorithm
  • mode of operation
  • key size, etc.
• Challenging, error prone (ICSE 2016)
• Alternatives claim to be more usable
  • libsodium, keyczar, cryptography.io

• Is this really true?
Online developer study

- Short python tasks, secure/insecure solutions
  - Symmetric: key gen./storage, encrypt/decrypt
  - Asymmetric: also certification validation

- One of 5 libraries:
  - PyCrypto, M2Crypto, cryptography.io, keyczar, PyNacl

- Exit survey
Not all libs support all tasks well
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 ./leca.pem. You can take also a look at the Let's Encrypt X3 Root CA and the server certificate.

```python
In [0]:
import nacl

def validate(certificate, root_certificate, hostname="citizen-measure.tk"):
    
    
    Purpose:
    Validate the given certificate's digital signature and common name.

    Arguments:
    certificate: The certificate to validate.
    hostname: The server's hostname.

    Return value:
    validationresult: True if validating the certificate is correct, False otherwise.

    Notes:
    - The Citizen Measure server certificate can be found at ./citizenMeasureCertificate.pem
    - The Let's Encrypt Certificate Authority certificate can be found at ./leca.pem
    - 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:
      - additional information sources go here (e.g. https://stackoverflow.com/questions/415511/how-to-get-current-time-in-python)

    # This is where your code goes
    return False

    # This is to test the code for this task.
    certificate = open("./citizenMeasureCertificate.pem").read()
    root_certificate = open("./leca.pem").read()
    assert validate(certificate, root_certificate, "citizen-measure.tk"), "Certificate validation failed."
    print "Task completed! Please continue."
```

Skeleton code, online code editor
Evaluation

• Correctness: Runs without errors, “works”
• Security: Manually coded
  • Predefined categories, 2 independent coders
• Self-report
  • Security thinking
  • System Usability Scale (SUS)
  • New API scale we designed
• Primarily analyzed w/ multiple regression
Recruitment via Github

• Scraped committers to 100k Python repos
• Invited random 50k of these
• Final, “valid” sample: 256
  • 208 professionals
  • 198 w/ no security background
  • 1571 who consented; many dropped out
• Slightly more GitHub-active than average
Functionality by library

Keyczar, m2crypto worst; c&p helps (significant)
Security (among functional)

“simplified” libs are most secure; asymmetric more secure than symmetric
Opinions about libraries

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

• From our scale: Documentation is key!
  • Keyczar: “Your documentation is bad and you should feel bad.”
Gap in security belief

• Believed secure but weren’t: 20% of tasks!
  • Not different by library
Participant background

• Experience level:
  • Did not matter on any metric

• Security background:
  • Almost mattered to security results
  • Helps with usability reports
Takeaways

• Implementing crypto is (still) hard
• Simplified APIs do promote security
  • Sort of!
• Documentation, full-featured-ness are key!

• ... Lots more data in the paper!
Contesting Secure Development

Andrew Ruef, Mike Hicks, James Parker, Dave Levin, Michelle Mazurek, Piotr Mardziel

ACM CCS 2016

(In 5 min or less)
Build it, break it, fix it

• There are coding contests and hacking contests ... why not combine them?

• Build: Score performance points
• Break: Another team’s code to reduce their score; gain points for you
• Fix: Fix bugs to get points back
  • And eliminate redundancy
Important things I am skipping

- Designing incentives properly
  - This is really hard!
- Designing interesting contest problems

... Please see paper for all the details
What correlates with *ship* success?

- Fewer LOC
- Using C/C++
What correlates with (at least one) security bug?

- Static >> C/C++
  - But only due to memory bugs!
- Devs who claim to know more languages
Success and failure

- High-level crypto libs helped
  - But not enough (replay)
- Problems with randomness
- Missing encryption/authentication entirely
- Wrong abstraction levels
  - Replay
  - Authenticate log entry but not entire log
Overall takeaways

• Usable security is important for devs, too!
• Empirical measurement helps
  • Interrogate conventional wisdom; identify gaps

• Key themes so far:
  • Documentation matters
  • Full-featuredness, abstraction levels matter

mmazurek@umd.edu
Backup slides
Participants

- 256 valid participants
- Ages 18-63
- 238 men, 18 women
- 208 professional Python programmers
- 195 had no IT security background
asymmetric tasks were harder to solve; certificate validation was hardest
secure key generation and storage was hard;
no secure certificate validation solution