16. Usable Security for Developers

Blase Ur, May 17th, 2017
CMSC 23210 / 33210
Today’s class

• Making security usable for developers
  – Motivation
  – Sources of security advice
  – Crypto APIs
  – Additional aspects
Developers Are Users, Too!
Security and human error

“Not long ago, [I] received an e-mail purporting to be from [my] bank. It looked perfectly legitimate, and asked [me] to verify some information. [I] started to follow the instructions, but then realized this might not be such a good idea … [I] definitely should have known better.”

-- former FBI Director Robert Mueller
Security and human error

Hi Blase,
Someone just used your password to try to sign in to your Google Account blaseur@gmail.com.

Details:
Sunday, October 30, 2016 9:38 PM (Central Africa Time)
Victoria Falls, Zimbabwe*

Google stopped this sign-in attempt, but you should review your recently used devices:

REVIEW YOUR DEVICES NOW

Best,
The Google Accounts team
Security and human error

• John Podesta (more precisely an aide) receives the following:

Security and human error

• IT services writes back:

From: Charles Delavan <cdelavan@hillaryclinton.com>
Date: March 19, 2016 at 9:54:05 AM EDT
To: Sara Latham <slatham@hillaryclinton.com>, Shane Hable <shable@hillaryclinton.com>
Subject: Re: Someone has your password

Sara,

This is a legitimate email. John needs to change his password immediately, and ensure that two-factor authentication is turned on his account.

He can go to this link: https://myaccount.google.com/security to do both. It is absolutely imperative that this is done ASAP.

Security and human error

On September 2015, a call was transferred from the main DNC switchboard to the Help Desk; I was handed the phone by a Help Desk staff member who stated that the FBI was looking for the person in charge of technology at the DNC. I took the call, and learned that the FBI thinks the DNC has at least one compromised computer on its network and that the FBI wanted to know if the DNC is aware, and if so, what the DNC is doing about it. I asked if the person calling, who stated he was Special Agent [REDACTED] can provide me with the means of identifying whom he claims to be. He did not provide me with an adequate response, but I did stay on the phone and talked about potential risks to the DNC, without giving him any identifiable information about the DNC, its personnel, or its assets. I did say that the DNC has, in the past, received phishing attack attempts, and ransom-ware type of attacks. The Special Agent told me to look for a specific type of malware dubbed "dukes" by the US intelligence community and in cyber-security circles.

Why are users stupid or lazy?

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

- End Users (> 1.5 billion)
- Developers (~350,000)
- System Designers (Google)

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 of the loop

• Try: Persuasive design
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


http://rpseawright.files.wordpress.com/2013/10/too-much-info.png
YOU GET WHERE YOU’RE LOOKING FOR (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 ...

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
Recruitment

- In/around 3 universities, U.S. and Germany
  - Email, flyers, craigslist, developer forums

- 1+ Android course or 1+ yrs pro

- Pass basic Android knowledge questions
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
Demographics: lab vs. online

Many similarities; Lab had more formal education
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)
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!
Comparing Crypto APIs
Getting crypto right is hard

• Developers must pick:
  – algorithm
  – mode of operation
  – key size, etc.

• Challenging, error prone (ICSE’16)

• 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

<table>
<thead>
<tr>
<th>Library</th>
<th>Current Version</th>
<th>Designed for Usability</th>
<th>Symmetric Key Generation</th>
<th>Symmetric Encryption/Decryption</th>
<th>Secure Symmetric Key Storage</th>
<th>Asymmetric Key Generation</th>
<th>Asymmetric Encryption/Decryption</th>
<th>Secure Asymmetric Key Storage</th>
<th>Certificate Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyCrypto</td>
<td>2.6.1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>M2Crypto</td>
<td>0.25.1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>cryptography.io</td>
<td>1.4</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Keyczar</td>
<td>0.716</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>PyNaCl</td>
<td>1.0.1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

● = fully applies; ○ = partly applies; ○ = does not apply
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
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
Invited vs. participated

Many similarities; Participants slightly more active
Keyczar, m2crypto worst; c&p helps (significant)
Security (among functional)

“simplified” libs are most secure; asymmetric more secure than symmetric
Self-reported data

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

• Experience level:
  – High if python is your job, or programming in python > 5 years
  – Did not matter on any metric

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

• Library experience: maybe helps usability
Takeaways

• Implementing crypto is (still) hard
• Simplified APIs do promote security
  – Sort of!
• Documentation, full-featured-ness are key!
What else can go wrong?
Example from today’s readings

![Not secure warning]

Your connection is not private

Attackers might be trying to steal your information from seclab.nu (for example, passwords, messages, or credit cards). NET:ERRCERTCOMMON_NAME_INVALID

- Automatically report details of possible security incidents to Google. Privacy policy

HIDE ADVANCED
Other Developer Concerns

• AWS (or other) access tokens
  – Don’t commit them to GitHub

• Credentials for MySQL, etc.
  – Don’t leave them in web-accessible directories (in case PHP crashes)
  – Don’t let people pick them
  – Don’t let them be spit out by verbose error messages
Other Developer Concerns

• Don’t keep legacy databases around
  – bcrypt vs. MD5

• Don’t allow password access for SSH

• Don’t allow remote access to your database

• Don’t use outdated Javascript libraries for your website
Configuring HTTPS
What can go wrong?

- Hacking Team *was* a consulting company that contracted with governments
- Many operational security errors
- Sys admin’s password: P4ssword

http://pastebin.com/raw/0SNSvyjJ