You are not your developer, either
A research agenda for usable privacy and security beyond end users

Yasemin Acar, Sascha Fahl, and Michelle Mazurek
CISPA, Saarland University
University of Maryland
“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

Facebook birthday invite leads to mayhem in Dutch town, authorities say

From Dominique Van Heerden, CNN
updated 8:49 PM EDT, Sun September 23, 2012

(cnn.com)
Why are users 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.

```c
.f ((err = SSLHai
    goto fail;
  goto fail;
.f ((err = SSLHai
    goto fail;
```
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
Research agenda: Beyond end users

• Measuring the status quo
• Understanding developers
• Methodology and validity
Measuring the status quo

• APIs, tools, documentation

• What is actually used and why?
  – Can we make security tools more attractive?

• How effective are security tools in practice?
  – Which are best and why?
  – What design features are effective?
  – Where in the development process to intervene?
Measuring the status quo: Agenda

• Expert review / cognitive walkthrough
• Field measurements in existing software
  – Github, app markets, etc.
• Controlled experiments for direct comparison

• Compare: security APIs, static analysis tools, security training materials, coding standards, etc.
Understanding developers

• What (anti) motivates secure behavior?

• How do developers learn about security?
  – How can we improve information resources?

• Where are knowledge gaps?
  – Can we address them or work around them?
Understanding developers: Agenda

• Ask about: priorities, information sources, acceptance of security tools, how security fits into the development process

• Interviews and surveys

• Diary studies
  – Experience sampling

• In-situ observation
Methodology and validity

• What type of study to use?
• When can you use students (vs. pros)?
• How to design useful study tasks?
  – Sufficiently complex to capture useful data
  – Doable in a study environment
Methodology and validity: Agenda

- Comparative studies of tasks, groups
- Compare study results to field observations
- Develop new measurement tools
  - Online study development platform
  - IDE with telemetry and experience sampling
Usable security for developers

• Lessons learned from end users
  – You are not your user, security is secondary

• A lot we don’t yet know
  – Comparison of existing tools and techniques for usability, effective security in practice
  – Motivations, incentives, knowledge gaps
  – How to best structure studies

mmazurek@umd.edu