TWC: Small: Empirical Evaluation of the Usability and Security Implications of Application Programming Interface Design

PIs: Brad A. Myers[1], Sam Weber[2], and Robert Seacord[2]
Contributing researchers: Michael Coblenz[3], David Keaton[2], Forrest J. Shull[4], Joshua Sunshine[5], Robert Schiela[2]
[1]Human Computer Interaction Institute [2]CERT / SEI [3]Computer Science Department [4]SEI [5]Institute for Software Research
Carnegie Mellon University

9/1/2014 - 8/31/2017

OVERVIEW

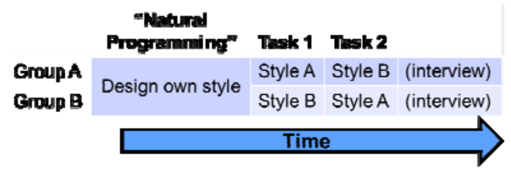
- Develop and empirically test *concrete* and *actionable* API design principles that lead to more secure code
- Investigate the tradeoffs between *security* and *usability* in language and API design
  - o When better usability leads to *more* secure code (e.g. [Wang]), and when it leads to *less* secure code (e.g., [Ellis][Stylos])
  - o Can we design languages and APIs that help programmers write secure code?
- Threat model: well-meaning and benign programmers, but arbitrarily malicious attackers of programmers' code
- Address *all* APIs, not just ones for security
  - o Security impact when programmers are thinking of functionality, not security
- Initial focus on two areas:
  - o Competing C and C++ parallelism language extensions
  - o Using immutability to reduce the likelihood of vulnerabilities, especially in concurrent code

SECURITY / USABILITY TRADEOFFS

- Prior empirical work has shown that guidelines for more secure code recommend coding styles that are less preferred / less usable by developers
  - o Factory pattern is 2.1 to 5.3 times slower [Ellis]
  - o Create-Set-Call preferred [Stylos]

METHODOLOGIES

- *Programmers are people* too – use proven HCI methods
- Investigate initial *learnability*
  - o How understandable?
  - o Fosters exploration during learning?
- Investigate *effectiveness* for novice and experienced users
  - o Error-proneness when coding – avoid security flaws
  - o Ability to find security issues in existing code
- "Contextual Inquiry" [Beyer] field studies
  - o Watch programmers working on their actual tasks looking for *breakdowns* and difficulties
  - o Understand issues with today's APIs and language features with respect to security and usability
- Corpus studies
  - o Look for evidence of usage and problems
  - o Change logs, bug databases, analysis of code
- Expert interviews
  - o Opinions about what is important to study further
- Surveys
  - o How widespread are the identified issues?
- Classroom studies
- Lab studies
  - o Controlled A vs B with different versions of API
  - o "Natural programming" to elicit expectations



C/C++ Parallelism Language Extensions

- **OpenMP** and **Cilk Plus** are being considered by ISO/IEC JTC1/SC22/WG14 CPLEX standards committee
- What are the Usability and Security issues with each?

Immutability

- Experts and books recommend **immutable** objects to reduce errors, especially in concurrent code (e.g., [Bloch][Oracle])
- However, a study found the create-set-call pattern is *more usable* for learnability than required parameters in constructors [Stylos]

INITIAL WORK

- The approach taken by CPLEX experts suggests *reductions* as a good area for study
  - o Especially avoiding race conditions
  - o Homeworks using OpenMP and Cilk Plus planned for Spring graduate class on security
  - o Study learnability and effectiveness of each API by students
- Taxonomy of immutability features in Java, C, C++, Objective-C, etc.
  - o `const, final, readonly`
  - o Reference vs. value immutability
  - o But programmers want "logical immutability"
    - ▪ E.g., if internal cache
- Examined commit logs in code repositories and found that users wanted logical immutability, but languages only provide bitwise immutability

References

[Beyer] Beyer, H. and Holtzblatt, K., *Contextual Design: Defining Custom-Centered Systems*. 1998, San Francisco, CA: Morgan Kaufmann Publishers, Inc.
[Bloch] J. Bloch. *Effective Java Programming Language Guide*. Mountain View, CA, Sun Microsystems, 2001.
[Ellis] Ellis, B., Stylos, J., and Myers, B. "The Factory Pattern in API Design: A Usability Evaluation," in *International Conference on Software Engineering (ICSE'2007)*. Minneapolis, MN: pp. 302-312.
[Oracle] Oracle Corp. Secure Coding Guidelines for the Java Programming Language, version 4.0. http://www.oracle.com/technetwork/java/seccodeguide-139067.html
[Stylos] Stylos, J. and Clarke, S. "Usability Implications of Requiring Parameters in Objects' Constructors," in *International Conference on Software Engineering (ICSE'2007)*. Minneapolis, MN: pp. 529-539.
[Wang] Rui Wang, Yuchen Zhou, Shuo Chen, Shaz Qadeer, David Evans, and Yuri Gurevich. 2013. Explicating SDKs: uncovering assumptions underlying secure authentication and authorization. In *Proceedings of the 22nd USENIX conference on Security (SEC'13)*. USENIX Association, Berkeley, CA, USA, 399-414.

This material is based upon work funded and supported by National Science Foundation award 1423054 and by Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM-0002023