

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



PIs: Brad A. Myers<sup>1</sup>, Sam Weber<sup>2</sup>, and Robert Seacord<sup>2</sup>

Contributing researchers: Michael Coblenz<sup>3</sup>, David Keaton<sup>2</sup>, Forrest J. Shull<sup>4</sup>, Joshua Sunshine<sup>5</sup>, Robert Schiela<sup>2</sup>

<sup>1</sup>Human Computer Interaction Institute <sup>2</sup>CERT / SEI <sup>3</sup>Computer Science Department <sup>4</sup>SEI <sup>5</sup>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
  - When better usability leads to *more* secure code (e.g. [Wang]), and when it leads to *less* secure code (e.g., [Ellis][Stylos])
  - 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
  - Security impact when programmers are thinking of functionality, not security
- Initial focus on two areas:
  - Competing C and C++ parallelism language extensions
  - Using immutability to reduce the likelihood of vulnerabilities, especially in concurrent code

## SECURITY / USABILITY TRADEOFFS

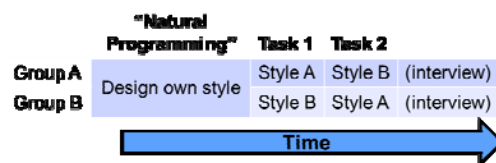
- Prior empirical work has shown that guidelines for more secure code recommend coding styles that are less preferred / less usable by developers
  - Factory pattern is 2.1 to 5.3 times slower [Ellis]
  - Create-Set-Call preferred [Stylos]

## 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 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.

## METHODOLOGIES

- *Programmers are people* too – use proven HCI methods
- Investigate initial *learnability*
  - How understandable?
  - Fosters exploration during learning?
- Investigate *effectiveness* for novice and experienced users
  - Error-proneness when coding – avoid security flaws
  - Ability to find security issues in existing code
- "Contextual Inquiry" [Beyer] field studies
  - Watch programmers working on their actual tasks looking for *breakdowns* and difficulties
  - Understand issues with today's APIs and language features with respect to security and usability
- Corpus studies
  - Look for evidence of usage and problems
  - Change logs, bug databases, analysis of code
- Expert interviews
  - Opinions about what is important to study further
- Surveys
  - How widespread are the identified issues?
- Classroom studies
- Lab studies
  - Controlled A vs B with different versions of API
  - "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
  - Especially avoiding race conditions
  - Homeworks using OpenMP and Cilk Plus planned for Spring graduate class on security
  - Study learnability and effectiveness of each API by students
- Taxonomy of immutability features in Java, C, C++, Objective-C, etc.
  - `const`, `final`, `readonly`
  - Reference vs. value immutability
  - 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

Copyright 2014 Carnegie Mellon University

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.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:\* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:\* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

\* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University.

DM-0002023