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

Pls: Brad A. Myers1, Sam Weber2, and Robert Seacord2
Contributing researchers: Michael Coblenz3, David Keaton2, Forrest J. Shull4, Joshua Sunshine5, Robert Schiela2
1Human Computer Interaction Institute 2CERT / SEI 3Computer Science Department 4SEI 5Institute for Software Research
Carnegie Mellon University

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
  o 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
• 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”
    1. 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
