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 functional, not security

Project Overview

• Develop and empirically test concrete and actionable API and programming language design principles that lead to more 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 Study: C/C++ Parallelism

Language Extensions [1]

• OpenMP and Cilk Plus are are C and C++ Parallelism Language Extensions being considered by ISO/IEC JTC1/SC22/WG14 CPLEX standards committee.

• Both support shared-memory fork-join parallel computation.

• Preliminary comparison in a master's level course identified usability problems with declaring and using reductions, multi-line compiler directives, and the understandability of task assignment to threads.

• Problems included memory leaks, race conditions, and performance bugs with both mechanisms.

• We found that Cilk Plus's mechanism for defining reducers is more usable than OpenMP's, and has a more familiar syntax.

Current work: Immutability Support in Java

• Characterizations of restrictions of changes [2]
  • Prevent change = immutability
  • Prevent certain clients from changing = readonly
  • Scope: individual objects or entire class
  • Transitive restrictions apply to included objects
  • Enforced statically or dynamically

• Interviews with programmers show unexpected state change causes many bugs [2]
  • Current language support is not adequate
  • Usability & expressiveness issues

• Glacier – statically enforces transitive class immutability in Java [3]
  • User study showed works better than final
    • Prevents real-world bugs and security vulnerabilities
    • Usable with minimal training
  • Glacier enforces immutability of @Immutable classes:
    • Parent class is immutable
    • Immutable public class Person {
        String name;
        int inches;
        int feet;
    }

Future work: Language Support for Blockchains

• Ethereum [4] and other platforms support computation with verifiable, global state ("an unstoppable world computer")

• Programming these platforms is difficult
  • Recent hack stole $60M from TheDAO by exploiting a vulnerability [5]
  • Ethereum limits resource usage by programs, but resource usage cannot be predicted, so programs are sometimes terminated before completion
  • Current programming languages are fairly standard (Go, or an adaptation from JavaScript)

Why a domain-specific language?

• Special characteristics: event-driven, highly stateful, correctness-critical, resource-limited

• Programs are immutable (i.e. bugs are unfixable)

• Bugs have severe consequences (e.g. money disappears or is stolen)

Goals:

• Support usable verification of key correctness properties
• Support reasoning about resource usage
• Make it easy for many kinds of programmers to write correct programs

References


