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[3]
Contributing researchers: Michael Coblenz[4], Whitney Nelson[5], Jonathan Aldrich[6], Joshua Sunshine[6]
[1]Human Computer Interaction Institute [2]New York University [3]NCC Group [4]Computer Science Department [5]REU summer student [6]Institute for Software Research

www.NatProg.org

Carnegie Mellon University

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:

`Person` class is immutable

```
class PersonHeight {
    int feet;
    int inches;
}

@Immutable public class Person {
    String name;
    PersonHeight height;


    public void setName(String name) {
        this.name = name;
    }
}
```

Error: can't include mutable object in immutable class

Error: can't assign to field of immutable class

	final	Glacier
Users who made errors enforcing immutability (after all tasks)	10/10	0/10

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

[1] Michael Coblenz, Robert Seacord, Brad Myers, Joshua Sunshine and Jonathan Aldrich, "A Course-Based Usability Analysis of Cilk Plus and OpenMP", 2015 IEEE *Symposium on Visual Languages and Human-Centric Computing (VL/HCC'15),* October 18–22, 2015, Atlanta, Georgia.
[2] Michael Coblenz, Joshua Sunshine, Jonathan Aldrich, Brad Myers, Sam Weber, Forrest Shull, "Exploring Language Support for Immutability" *ICSE'2016: The 38th International Conference on Software Engineering*, Austin, TX, May 14 - 22, 2016. pp. 736-747.
[3] Michael Coblenz, Whitney Nelson, Jonathan Aldrich, Brad Myers and Joshua Sunshine, "Glacier: Transitive Class Immutability for Java," *ICSE'2017: The 39th International Conference on Software Engineering*, Buenos Aires, Argentina, May 20-28, 2017. To appear.
[4] Ethereum Project, https://www.ethereum.org/
[5] The DAO Attacked: Code Issue Leads to $60 Million Ether Theft. *CoinDesk*. http://www.coindesk.com/dao-attacked-code-issue-leads-60-million-ether-theft/

National Science Foundation
WHERE DISCOVERIES BEGIN

The 3rd NSF Secure and Trustworthy Cyberspace Principal Investigator Meeting
January 9-11, 2017, Arlington, Virginia

NSF CNS-1423054
9/1/2014 - 8/31/2017

Table 1: Summary of Some Existing Systems (abbreviations are from Table 2)

System	Type	Scope	Trans.	Init.	Abstr.	Compat.	Enforcement	Polymorph.
Java `final`	a	o	n	e	N/A	c	s	n
C++ `const`	r	o	n	e	a	c	s[1]	n
Obj-C immutable collections	i	o	n	e	N/A	c	s[1]	n
.NET Freezable [22]	i	o	n	e	N/A	o	d	n
Java `unmodifiableList`	r	o	n	e	N/A	o	d	n
Guava `ImmutableCollection`	i	o	n	e	N/A	o	s, d[2]	n
IGJ [42]	i, r	c, o	n	e	a	c	s	p
JAC [19]	r	o	t	e	c	c	s	n
Javari [38]	r	c, o	t	e	a	c	s	p
OIGJ [43]	i, r, o	c, o	n	r	a	c	s	p
`immutable` [16]	i, r, o	c, o	t	r	a	o	s	p
C# isolation extension [13]	i, r, o	c, o	t	r	a	c	s	p
JavaScript `Object.freeze`	i	o	n	e	c	o	d	n

[1] These approaches provide static enforcement to the extent possible in these languages.
[2] Static deprecation warning, runtime exception

Table 2: Summary of Dimensions

Dimension	Possible choices
Type	immutability, read-only restriction, assignability, ownership
Scope	object-based, class-based
Transitivity	transitive, non-transitive
Initialization	relaxed, enforced
Abstraction	abstract, concrete
Backward compat.	open-world, closed-world
Enforcement	static, dynamic
Polymorphism	polymorphic, non-polymorphic