Obsidian: a Safer Blockchain Programming Language

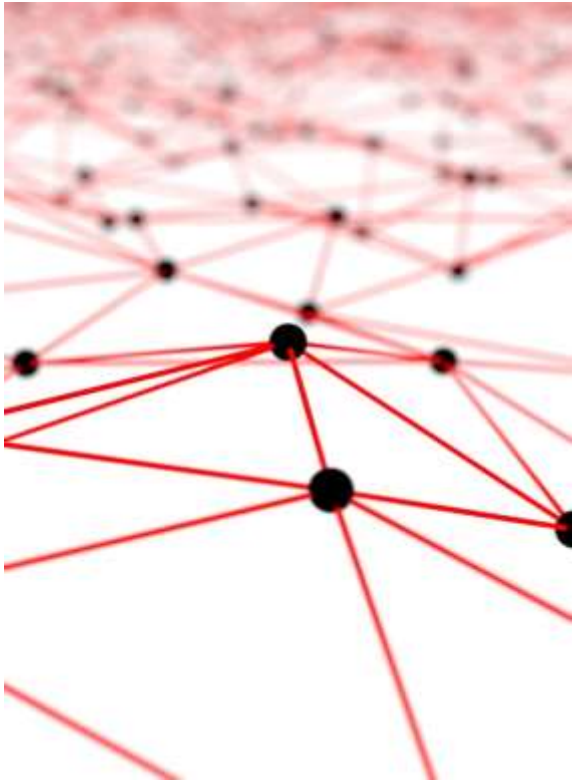
Michael Coblenz: PhD student, CMU Computer Science Department

SEI PI: Elli Kanal

Jenna Wise, Joshua Sunshine, Jonathan Aldrich, Brad Myers (CMU)

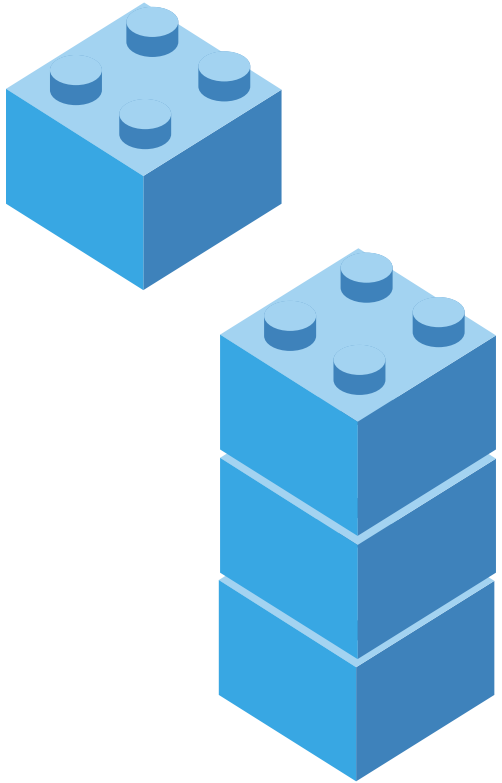
Rick Hull (IBM)

What are Blockchains?



- Support shared, global state on distributed systems
- Resilient to attacks compromising some of the peers
- Programs: “smart contracts”

What are Blockchains?



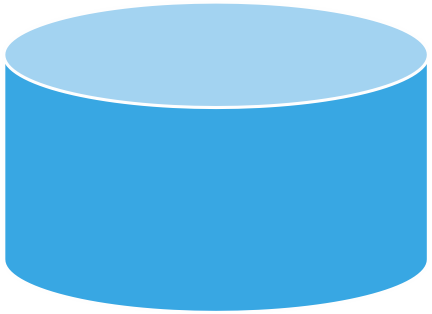
- Transactions can modify state
- Transactions deploy contracts or invoke code in existing contracts
- “Code is law:” code can specify an agreement between parties
- Programs are immutable

What are Blockchains?

A 3-block blockchain

transaction 1 transaction 2 transaction 3	transaction 4 transaction 5 transaction 6	transaction 7 transaction 8

Key-value store with state of all contracts



Blockchain Applications in DoD

1. Health records (VA, CC Innovation Center)
2. Supply chain
   A. Tracking responsibility
   B. Establishing provenance
3. Logistics
4. Resilient Communications (DARPA SB162-004, News Article)
5. Cyber Security (DHS)

Blockchain Programming

Existing blockchain programs are vulnerable to attack

Over **$40M** were stolen from TheDAO due to a bug in the implementation (June 2016)

$32M were stolen due to a bug in a commonly used contract (June 2017)

Bugs in smart contracts cannot be fixed after deployment.

We want to build correct software, but current approaches have been shown to have security vulnerabilities.

Obsidian: a new programming language

A user-centered, domain-oriented design approach.

- Goals:
  - Make certain vulnerabilities impossible
  - Make it easier to write correct programs
  - Show effectiveness and correctness

Components of Obsidian

1. Typestate-oriented programming
   - Shown to be helpful in documentation, but no studies of writing code
2. Resource types
   - Integration into an OO-style language is novel

An Example (Selected for Brevity)

```
resource contract money {…};
contract Bond {
        account seller;
        Bond(account s) {
                seller = s;
                -> Offered();
        }
        state Offered {
                transaction buy(money m, account b) {
                        seller.pay(m);
                        -> Sold(buyer = b);
                }
        }
        state Sold {
                account buyer;
                transaction makePayment(money m) {
                        buyer.pay(m);
                }
        }
}
```

```
contract ErroneousClient {
        transaction badTransaction() {
          Bond.Offered b = new Bond(…);
          b.buy(…);
          b.buy(…);
 }
}
```

Compile error: b is of type Bond.Sold, which has no buy() transaction.

A User-Centered Language Design Process

Traditional approach: design the language and then evaluate it

Our approach: iteratively design parts of the language and evaluate with participants

First: designed a language formalism and surface syntax; implemented a compiler

So far: completed one round of user testing in a text editor

Natural programming technique elicits design input and feedback

Evaluated transition syntax/semantics and one aspect of resource types

Revising language based on results

Currently: evaluating approaches to permissions (to address aliasing), approaches to state transitions

Novel approach: evaluate in Java to apply to Obsidian

Eventually: summative study

Summary

Blockchains offer a promising approach: security, resilience, correctness

Current approaches have resulted in vulnerable, buggy programs

Obsidian is designed to help programmers write correct code more easily

Contact Information

Presenter / Point(s) of Contact

Michael Coblenz

Doctoral Student, CMU Computer Science Department

Email: mcoblenz@cs.cmu.edu

Contributors

Elli Kanal (SEI)

Tyler Etzel (Cornell)

Celeste Barnaby (Wesleyan)

Jenna Wise (CMU)

Joshua Sunshine (CMU)

Brad Myers (CMU)

Jonathan Aldrich (CMU)

Rick Hull (IBM)

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM17-0791