Obsidian: a Safer Blockchain Programming Language

Michael Coblenz, Tyler Etzel, Joshua Sunshine, Jonathan Aldrich,
Brad Myers, Eli Kanal, and Mark Sherman

Carnegie Mellon University

institute for SOFTWARE RESEARCH

HCII

Software Engineering Institute
Carnegie Mellon

So, you want to build an application?

- **Centralized systems** require trusted computing resources
- **Decentralized systems** support safe application development on **untrusted**, **shared** computing resources.
- Blockchains systems:
  - have **shared global state**
  - are **robust to failures**
  - enable **verifying correctness** of state
  - support **interoperation** among diverse organizations
- Proposed applications include: financial transactions; supply chain; health records; organizational management (e.g. voting); renewable energy trading
- Existing platforms include *Ethereum,* a public platform, and *HyperLedger*, designed for government and corporate use.
- Programs for blockchain platforms are called *smart contracts*.

Programming blockchains

- Blockchain applications are typically correctness-critical; suppose a banking application tracked money incorrectly!
- Ethereum is programmed using domain-specific languages, the most popular being *Solidity*; HyperLedger currently supports programming in Java and Go.
- Recently, someone stole over $40M-worth of virtual currency from an organization on Ethereum due to a security vulnerability in the organization's contract [2].
- We are designing a new language, *Obsidian*, for HyperLedger, to prevent many common bugs and security flaws.

Typestate-oriented programming

- Many of the existing and proposed applications implement *state machines*. However, users must implement state machines in an ad hoc manner, making it harder to reason about the states.
- Invocations on contracts that are in undefined states have led to security breaches [2].
- A bond might be *offered*, *sold*, or *expired*. Available operations depend on the state: once a bond is sold, it cannot be sold again.
- Obsidian programs represent states explicitly, including expressing checkable invariants on states.
- Typestate-based documentation has been shown to aid in correct understanding of API preconditions [1].

```
contract Bond {
   account seller;
   state Offered {
      transaction buy() {…}
   }

   state Sold {
        account buyer;
   }
}
```

buy() unavailable on Sold bonds

Linear resources

- In existing languages, money is (internally) tracked traditionally, e.g. as integers in state variables.
  - There can be accounting bugs in which money is created, destroyed, or trapped inside the contract forever!
- Obsidian will prevent this with *linear types*: values must be used exactly once.

```
currency balance = withdrawFunds();
depositFunds(balance);
depositFunds(balance);
```

COMPILE ERROR: cannot spend balance twice!

Evaluation

- We will write proofs of safety theorems for Obsidian.
- In addition, we will do *user studies* in which we evaluate whether:
  - Users can effectively use Obsidian to write correct blockchain programs
  - Programs written in Obsidian tend to have fewer vulnerabilities than equivalent programs written in existing languages

Conclusions/Future Work

- Domain-specific languages offer a promising approach to programming blockchains.
- We are working on the language design and implementation. Evaluation work will follow.
- We plan to collaborate with the HyperLedger community in the hope of having our language adopted by a real blockchain platform.
- You can partner with us to evaluate your favorite applications!

[1] Structuring Documentation to Support State Search: A Laboratory Experiment about Protocol Programming. Joshua Sunshine, James D. Herbsleb, and Jonathan Aldrich. ECOOP 2014
[2] E. Gün Sirer, "Thoughts on the DAO hack," 2016. http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/

HOTSOS Symposium & Bootcamp
HOT TOPICS in the SCIENCE OF SECURITY
APRIL 4-5, 2017 | HANOVER, MARYLAND