Obsidian:
A Safer Blockchain Programming Language

Michael Coblenz, Tyler Etzel, Joshua Sunshine, Jonathan Aldrich, Brad Myers, Eli Kanal, and Mark Sherman
Carnegie Mellon University
Pittsburgh, PA USA

***Abstract*—Blockchain platforms, such as Ethereum [1], promise to facilitate transactions on a decentralized computing platform among parties that have not established trust. Recognition of the unique challenges of blockchain programming has inspired developers to create domain-specific languages, such as Solidity [2], for programming blockchain systems. Unfortunately, bugs in Solidity programs have recently been exploited to steal money [3]. We propose a new programming language, Obsidian, that makes it easier for programmers to write correct programs.**

***Keywords*-blockchain programming, blockchain security, programming language usability**

I. Introduction

Ethereum [1] and HyperLedger [4] are blockchain-based programming platforms that share the goal of enabling parties that have not established trust to conduct transactions in a distributed computing environment. Blockchain platforms maintain consistent distributed global state and enable participants to run programs that update the shared state. Unlike some other distributed computing platforms, blockchains assure correct state even when some of the servers execute the code maliciously. For example, a distributed autonomous organization could sell shares to parties in exchange for virtual currency. The shareholders may then vote on proposals to pursue with the organization's resources. The organization exists as a *contract*, which is a program that can maintain its own state and conduct *transactions* that transform that state according to messages sent by participants. Blockchains allow shareholders to host the organization in a distributed fashion and still trust the platform to execute the agreed-upon contract faithfully. However, if there is a security vulnerability in the contract, then it may be exploited. Such a vulnerability recently resulted in the loss of over $40M [3].

We aim to make blockchain programs less bug-prone by designing a new programming language that encourages writing programs that avoid classes of vulnerabilities such as the one above. Though some safety properties will be guaranteed by the compiler, we also focus on *usability*: can real programmers write correct code in our language, and is their code less likely to be buggy than the code they would have written in Solidity?

Several characteristics of blockchain programs motivate a new language design. First, correctness is critical: many of the proposed applications of blockchains involve financial transactions, so any bugs may result in lost or stolen money or virtual currency. Second, bugs in programs cannot be fixed because the programs are immutable. Once money is committed to an agreement (as implemented in a contract), that money can only be removed according to the existing contract implementation. Third, blockchain programs are commonly *state-based* [5]: finite-state machines provide a simple abstraction for high-level aspects of program behavior. We will exploit this in a novel language design, based on evidence that state-based reasoning facilitates faster, less buggy development [6].

II. Our Approach

Obsidian (now in a prototyping phase) is an object-oriented language that makes state first-class [7]. The methods that can be invoked on an object depend on the object's current state. Obsidian also treats money ("ether") as a *linear* resource, allowing the compiler to statically detect a class of errors involving money and other linear resources. Details and an example will be presented in the poster to be presented.

Acknowledgment

This work was supported by grants from the Department of Defense, the National Science Foundation, and the Software Engineering Institute.

References

[1] Ethereum Foundation, "Ethereum project," http://www.ethereum.org. Accessed Jan. 3, 2017.
[2] ——, "Solidity," https://solidity.readthedocs.io/en/develop/. Accessed Jan. 3, 2017.
[3] E. Gün Sirer, "Thoughts on the DAO hack," 2016. [Online]. Available: http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/
[4] The Linux Foundation, "Hyperledger," https://www.hyperledger.org. Accessed Jan. 3, 2017.
[5] Ethereum Foundation, "Common patterns," http://solidity.readthedocs.io/en/develop/common-patterns.html. Accessed Jan. 4, 2017.
[6] J. Sunshine, J. D. Herbsleb, and J. Aldrich, "Structuring documentation to support state search: A laboratory experiment about protocol programming," in *European Conference on Object-Oriented Programming (ECOOP)*, R. Jones, Ed., 2014.
[7] J. Sunshine, K. Naden, S. Stork, J. Aldrich, and É. Tanter, "First-class state change in Plaid," in *ACM SIGPLAN Notices*, vol. 46, no. 10. ACM, 2011, pp. 713–732.