Glasha: A Secure Blockchain Programming Language

Blockchains have been proposed as a way to support transactions between multiple parties on a distributed shared state. However, security vulnerabilities have already been exploited in existing blockchain programs. For example, the DAO exploit[1] resulted from a vulnerability found in a popular blockchain programming language, Solidity. Not many solutions to these security vulnerabilities exist, yet blockchain programs often deal predominantly with sensitive data that needs to be protected.

Solidity, a popular blockchain programming language that runs on the Ethereum Virtual Machine, employs the use of static analysis to warn developers of potential vulnerabilities. However, these warnings can be ignored. Obsidian, a language being developed by Michael Coblenz and others at Carnegie Mellon University, will not compile potentially insecure programs. While Obsidian can be more effective than Solidity in preventing certain security vulnerabilities, I believe that my domain-specific language (DSL) Glasha can be more effective than Obsidian by changing the programming paradigm from contract-oriented (similar to object-oriented) to functional.

Functional languages support many features that are especially beneficial for blockchain programs. These languages are known for being immutable, which means the objects within a program cannot be changed once they are created. Immutability is an especially important concept for blockchains - one of their key features is that once a transaction has occurred it cannot be changed. Functional languages are also known to be fault tolerant, which describes the ability of a system to continue running even in the event of a failure. Since blockchains support interactions between multiple parties, programs should continue running even if an error occurs in one of the participating parties. Furthermore, functional programs can be run concurrently, and concurrency is a key feature of distributed systems which allows for different components to execute at different times.

Broader impacts: As blockchains are gaining in popularity, and blockchain programs often deal with sensitive data such as money, these programs become a prime target for hackers. In order to combat these security concerns, programming languages can prevent insecure programs from ever being executed. Finally, software developers will no longer have as many security concerns to keep in mind while designing blockchain programs. My new DSL, Glasha, will enable programmers to develop secure blockchain programs in a high level language with ease.

Research Plan:

To address the issues with current blockchain programming languages, I plan on creating a functional DSL with features beneficial for blockchain programs.

Intellectual Merit: Having worked over the summer with the Obsidian language team at Carnegie Mellon, I already have some knowledge of what common security issues arise in blockchain programs and how current languages have addressed them. I also have experience modeling formal semantics for programming languages through other research and class projects. Thus, I will begin with identifying the most common and valuable programming language features that blockchain programs already employ. These features will be modeled in a functional language semantics based on the lambda calculus. As mentioned previously,

[1] Emin Gün Sirer. "Thoughts on The DAO Hack". In: (2016). URL: http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/.

blockchain programs often deal with important data. One way to track this data throughout the program is by using linear types[2], which would be simple to include in the semantics. However, other features such as having run time information about the caller of the function, a feature in Solidity, will be more challenging to integrate into the semantics. Creating the formal semantics will also require either handwritten or mechanized proofs for its soundness and type safety, to ensure that these features useful for secure blockchain programs blend together in a logical way. Following this plan, my work could be completed in year one of my PhD and result in a peer-reviewed publication.

Not only does Obsidian prevent common security vulnerabilities in blockchain programs, but it is also designed in a user-centered way. User studies have been conducted in order to distinguish whether programmers are able to understand some of its complicated features and use them in a meaningful way. User studies have already been conducted for functional languages[3], and I will create similar studies for Glasha. It is important for languages to be considered usable by programmers. A language may provide many safety guarantees, but if a programmer cannot understand how to write common programs in the language, these guarantees are useless. In order to prevent more security vulnerabilities than existing blockchain programming languages, Glasha must function in a user-friendly way. Creating and conducting these user studies could be completed in year two and part of year three of my PhD and result in a peer-reviewed publication.

After conducting these user studies, which will lead to identifying syntax and semantics that programmers find most usable, I will begin implementing the language. My experience in both coursework and research have shown me that language implementation takes a substantial amount of time. Furthermore, I will have to target this language to a particular blockchain platform. Most likely, I will target the Hyperledger Fabric platform, since there is no concept of resource usage on this platform, in contrast to the Ethereum Virtual Machine. However, I will want to chose the platform that is most popular at the time. By targeting a popular blockchain platform, I am giving programmers the power of abstraction to create their programs in a high level secure functional language on a popular platform. They would no longer have to concern themselves with all the potential security issues that may arise in their programs. Furthermore, I would like the compiler of this language to be efficient and fast, for an overall better programming experience. I would use the techniques used in well known fast and efficient functional language compilers such as Chez Scheme. Implementing this language with a compiler to target a popular blockchain platform would take the latter end of my third year and my fourth year of my PhD resulting in a peer-reviewed publication.

Finally, as my PhD is wrapping up, I will have extensive knowledge about implementing domain specific languages for blockchain programming. I will use this knowledge to do a final comparison to other blockchain languages, such as Obsidian, Rholang, and Solidity. I will run a comparison of the usability of my language Glasha to these languages, to show how conducting user studies early in the design process can lead to a more usable programming language. This final work would result in a peer-reviewed publication, and a successful completion of my PhD.

[2]Philip Wadler. "Linear types can change the world". In: *IFIP TC*. vol. 2. 1990, pp. 347–359.

[3]Preston Tunnell Wilson, Kathi Fisler, and Shriram Krishnamurthi. "Evaluating the Tracing of Recursion in the Substitution Notional Machine". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. 2018.