



Solidity Developer Security Practices Survey - Final Report

06.15.2023

Jessica Mack
UIUC

Introduction

As blockchain technology has risen in prominence, so too have smart contracts as a method of managing assets on blockchains. These self-executing programs currently control millions of dollars worth of assets and have been the subject of several prominent hacking attacks in recent years⁶. While studies have been done on how developers approach security in other fields of software development²³, smart contract development is still relatively unexplored.

Blockchains are distributed ledgers that use a shared protocol to agree on the state of the “chain”. Cryptocurrencies are the most well known use case for blockchain technology. The “coins” of the currency are “mined” as a reward for processing blocks of transactions on the chain. Smart contracts are programs stored on the chain that are increasingly being used to control cryptocurrency assets. Once deployed to the chain, smart contracts can not be upgraded or patched. The most popular chain for smart contracts is Ethereum. Smart contracts in Ethereum run on the Ethereum Virtual Machine (EVM). Of the various languages that used to write for the EVM, Solidity is the most popular. Solidity developers are the main group being examined in this study.


The goal of this study is to answer the following questions: **How do smart contract developers approach or ensure security for smart contracts? Are there any differences between early stage and experienced developers?** and **How do smart contract developers conduct code reviews and whether they are able to identify common smart contract security vulnerabilities in the code?**

By answering those questions we hope to provide data on the experience of developers in this new field of software development and provide guidance for areas that can be addressed to improve security practices among smart contract developers.

Related Work

Developer Security Practices

Smart contract development is a relatively new field of software development and as a result a lack of security standards could be due to the age of the field. However, prior



research shows that even in more established forms of software development, where best practices are known and better tooling is available, there are still often lapses in security due to human behavior and motivation²⁵. The issues of “being unequipped for security because of a perceived lack of security knowledge or the unavailability of necessary tools” and “competing priorities & no plan, where security has a lower priority than other aspects of the software” are shared concerns across the developer groups².

Methodology

This study is a follow up to a previous study done by Tanusree Sharma, Zhixuan Zhou, Andrew Miller, and Yang Wang in the summer of 2021¹.

Study Design

This project was designed as a large-scale survey study on the security practices of Solidity developers. Unlike the previous study, for this study the code review task was done without any interaction with an interviewer. The study was conducted online and involved a survey with multiple choice questions, Likert scale responses, open answer responses, and a random smart contract code review task from a pool of four possible tasks. Four code review tasks were devised based on the most commonly found vulnerabilities in smart contracts. This also provided a chance to gauge whether some vulnerabilities were more identifiable than others to the developer pool.

Participant Recruitment

To gain a wide array of participants, the link to the survey was posted in various developer and blockchain communities. There was no selection of participants as it was open to anyone who wanted to participate. In the end we received 20,000+ responses in total that were filtered down to around 900+ responses that were analyzed in the context of the research questions. We initially received fewer responses than expected so the survey was reopened and reposted.

Participation in the study was completely voluntary and participants were allowed to quit at any time. However, the final results were only taken from responses that had completed the entire survey. The reward of entry into a lottery for a gift card was stated upfront and approved by the IRB. The range of times to complete the survey were from a couple minutes to several hours.

Analysis

Survey response data was downloaded from Qualtrics and analyzed using thematic analysis⁴. Thematic analysis was used to determine high quality from low quality answers using several criteria such as:

- Time to completion
- Open-Source resources referenced
- Development tools referenced
- Security vulnerabilities cited
- Click pattern on code review task page

I performed open coding on the entirety of the data set and shared themes I determined with my graduate adviser. We then discussed the coding, refined it, and then came to a consensus that was then applied to narrowed down higher quality sections of the data set.


Limitations

Because the links to the study were posted in open forums, boards, and groups they were open to the possibility of being spammed. The majority of the responses were in fact flagged as spam or otherwise low quality. They often had empty open response sections or completed the survey in an unrealistic amount of time. This especially impacted the code review task answers. In the previous study¹, the interviewers received much more in depth responses to the code review tasks as someone being present impacted participants' motivation to explain their thought process over having to write it out.

While the respondent group still included a wide range of developers the people reached by the survey will be those who actively engage in online discussion areas for smart contract development which can mean that those who are less likely to engage in online communities are not accounted for.

Findings

56.1% of developers surveyed used more than one information source to learn about smart contract development with Google Search being the most common at 44.4% of all participants. The second most used resource was the Solidity documentation itself at 38.6%. Most developers used one development tool to create smart contracts at 49.5%.



Ganache, Geth, Remix IDE, and HardHat were the top tools referenced at similar rates and then there was a drop off to the 5th place.

Participants were asked what is most important in smart contract development and offered six options: User Experience, Functionality Correctness, Gas Efficiency, Security, Code Optimization, and Maintainability. They were given the ability to rank the options 1 to 5. Among them User Experience was ranked 5 most often, had the highest average, and scored the highest overall. Security had the second highest average but the third highest mode. Gas Efficiency and Maintainability had the lowest mode and the two lowest averages.


For code reviews, Manual Inspection was the most used method followed by the open source static code analyzer Slither. After those two there's a drop to the built in tools in Remix and Hardhat and then the proprietary software MythX by ConsenSys Software Inc.

Conclusion

Similar to the previous study we found that standard documentation, reference implementations, and security tools are not sufficient resources to prepare developers to recognize vulnerabilities in smart contract code. An even smaller percentage of developers noticed the vulnerabilities in the code review task in this study, 0% of early stage and 5% of experienced developers as compared to 15% and 55% respectively in the first study. Participants again pointed out shortcomings of current smart contract security tooling such as its lack of features found in other common development tools and the lack of usability.

References

1. Tanusree Sharma, Zhixuan Zhou, Andrew Miller, Yang Wang. Exploring Security Practices of Smart Contract Developers. April 24, 2022
2. Hala Assal and Sonia Chiasson. 'think secure from the beginning' a survey with software developers. In Proceedings of the 2019 CHI conference on human factors in computing systems., May 4–9, 2019
3. Hala Assal and Sonia Chiasson. Security in the software development lifecycle. In Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018), August 12–14, 2018

- 
4. Richard E. Boyatzis. Transforming Qualitative Information: Thematic Analysis and Code Development. SAGE, April 16, 1998
 5. Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, Matthew Smith. "If you want, I can store the encrypted password." A Password-Storage Field Study with Freelance Developers. May 4–9, 2019
 6. Jamie Redman. Flash Loan Attacks Drain 2 Binance Smart Chain Defi Projects for \$6 Million, May 24, 2021