1 Blockchain Smart Contract Fortification using Bytecode Analysis to Address Vulnerabilities

Mohammed Abdul Lateef

Jawaharlal Nehru Technological University Hyderabad, India
E-mail: malateefz2101@gmail.com

Dr. A. Kavitha

Jawaharlal Nehru Technological University Hyderabad, India
F-mail: athota.kavitha@gmail.com

Abstract

Smart contracts and blockchain platforms have revolutionized various industries, offering decentralized and transparent execution of agreements. However, they are not immune to security lapses, and the presence of vulnerabilities has led to security issues. This research delves into the realm of smart contract security, employing bytecode analysis as a powerful tool to unveil and rectify vulnerabilities. By meticulously scrutinizing the intricate low-level instructions and EVM opcodes, our objective is to unearth potential issues and provide actionable insights for enhancement. Through a compelling case study elucidating a smart contract designed for decentralized electoral integrity, we seamlessly integrate automated tool analysis with manual bytecode examination. Our overarching goal is to elucidate the nuanced process of leveraging bytecode analysis to effectively detect and mitigate vulnerabilities inherent in smart contracts, thus fortifying their robustness and reliability.

Keywords: Smart Contracts, EVM Opcodes, Automated Analysis, Manual Examination, Vulnerability

Introduction

Blockchain tech, especially with Ethereum's smart deals(contracts), has changed lots of areas by making deals that run on their own with lots of auto-work. But these deals can be tricky because once they're set, they can't be changed, and everyone can see them. This means they need very careful checks for any problems, by people and with tools.

Smart deals work on a simple "if this, then that" way and have made things easier in places like banks and supply chains. But, their go-it-alone ability also brings risks that need sharp eyes. This study digs into the troubles with smart deals on Ethereum, aiming to understand what goes wrong, how bad it can be, and how to fix these troubles[1]. This could help make blockchain more safe.

This research dives deep into the troubles with smart deals on

Ethereum to grasp how they work, what goes wrong, and how to fix these troubles. Looking at how they're made and the big security issues, the aim is to make blockchain safer[2]. With a close look, builders can make smart deals stronger, even when the rules around them are fuzzy, helping create a tougher blockchain world.

Literature Review

The birth of smart contracts was a major turning point in blockchain technology because it introduced self-executing agreements that are much faster than what is traditionally used [3]. However, the more practical they became, the more problems were discovered. Critics found everything from mistakes in coding to flaws throughout systems as flaws like these were being searched for [4]. This increased scrutiny has shown how important it is to know about security when dealing with smart contracts on blockchains; therefore there should be research done continuously into this area so that risks can be identified such as reentrancy attacks which allow hackers access back into an already ongoing transaction after they've left once thereby stealing all its value since it keeps looping through forever until balance becomes zero even though overflow vulnerabilities were categorized still guiding towards making them stronger[5].

Smart contract vulnerability is a complicated and multifaceted problem that requires careful consideration and proactive measures[6]. With new threats constantly emerging, collaborative efforts among different organizations should be encouraged in order to build strong security foundations around smart contract ecosystems[7]. A good way to avoid this would be using vulnerability assessments as well as setting up strict protocols based off them thus helping concerned parties traverse through issues arising from these weaknesses while ensuring trustworthiness within transactions conducted via block chains[8].

SECURE-AMSVA -- Systematic Evaluation For Comprehensive And Unified Review Of Automated And Manual Smart Contract Vulnerability Assessment

This research is built around SECURE-AMSVA, which is a method used to identify weaknesses in Ethereum-based smart contracts. This formalized system is designed to be comprehensive and meticulous so as to ensure that no potential security risk goes unnoticed. The methodology combines automated and manual techniques in order to carry out broad assessments for different kinds of vulnerabilities.

SECURE-AMSVA serves as an example of how thorough evaluations can be made towards ensuring safety measures for smart contracts built on Ethereum are effective. It is made up of tools that help with automation as

well as human input which makes it holistic. By doing this the approach attempts at benefiting from both methods thus increasing the range and intensity levels in which bugs may be found. Through using such automated programs alongside expert knowledge workers, efficiency gains can be achieved while still maintaining accuracy required during deep analysis aimed at establishing whether a given ethereum based application has any security holes or not.

- Key Components of SECURE-AMSVA

Thorough Evaluation: Smart contracts are methodically evaluated for security flaws in SECURE-AMSVA. For this purpose, some automatic analysis instruments (like Mythril and Slither) are used which help to find known vulnerabilities as well as programming mistakes[9].

Unified approach: The program combines results of automated checks with manual ones so that users can see them all together as one picture about their contract's safety level[9]. This allows them understand risks better and fix any weaknesses properly.

The Vulnerability Confirmation Engine (VCE) is what makes sure a smart contract vulnerability detection system is accurate and reliable. It acts as an interface between scanners' initial identification of potential vulnerabilities and developers' actionable remediation efforts. In this article we do a deep dive into VCE technicalities, its methods, advantages, disadvantages are discussed here so that people will know more about it.

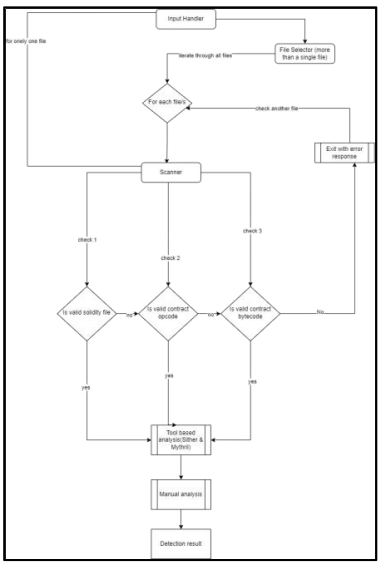

Figure 1: Approach acquired to implement VCE

This is highly customizable and can be integrated into development pipelines for automatic vulnerability checks. Together, these tools offer a comprehensive approach to identifying and mitigating vulnerabilities in Solidity contracts, enhancing the overall security of blockchain applications.

Implementation And Results

The implementation requires the constant and immersive accent of the tools used for assessment. Both the tools i.e. Mythril and Slither play a crucial role in identification process.

Table 1: Dissemination of Smart contract vulnerability analysis

1. Bytecode Analysis					
Tool	**Re-entrancy**	**Unchecked External call**	**Integer Overflow**	**Un-initialized variable**	**Access Control**
Mythril	Yes	Incorrect assessment	Incorrect assessment	Incorrect assessment	Incorrect assessment
Slither	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Manual	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
2. EVM Opcode Analysis					
Tool	**Re-entrancy**	**Unchecked External call**	**Integer Overflow**	**Un-initialized variable**	**Access Control**
Mythril	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Slither	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Manual	Yes	Yes	Yes	Yes	Yes
3. Solidity Code Analysis					
Tool	**Re-entrancy**	**Unchecked External call**	**Integer Overflow**	**Un-initialized variable**	**Access Control**
Mythril	Yes	Yes	Yes	Yes	Yes
Slither	Yes	Yes	Yes	Yes	Yes
Manual	Yes	Yes	Yes	Yes	Yes

Conclusion

This research project embarked on a mission to fortify the security posture of Ethereum-based smart contracts by employing a rigorous and multi-faceted vulnerability assessment methodology, aptly named "SECURE-AMSVA" (Systematic

Evaluation for Comprehensive and Unified Review of Automated and Manual Smart Contract Vulnerability Assessment). Through this comprehensive approach, the project sought to tackle the ever-evolving landscape of vulnerabilities that smart contracts face, including reentrancy attacks, integer overflows, unchecked external calls, uninitialized variables, and access control issues.

References

1. Wenbo Yang, et al. "An Opcode-Based Vulnerability Detection of Smart Contracts." IEEE Access, vol. 8, no. 1, pp. 1384-1395, 2020. doi:10.1109/ACCESS.2019.2962332.

2. Xinyi Zhang, et al. "Block-gram: Mining Knowledgeable Features for Efficiently Smart Contract Vulnerability Detection." IEEE Transactions on Information Forensics and Security, vol. 18, no. 4, pp. 1218-1232, 2023. doi:10.1109/TIFS.2022.3186001.

3. Yuxuan Xie, et al. "HGAT: Smart Contract Vulnerability Detection Method Based on Hierarchical Graph Attention Network." IEEE Access, vol. 11, no. 1, pp. 1154-1165, 2023. doi:10.1109/ACCESS.2022.3155434.

4. Akashdeep Sharma, et al. "Vulnerability Analysis of Smart Contracts." In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 1873-1890. ACM, 2022. doi:10.1145/3572243.3572247.

5. Karthikeyan Natarajan, et al. "Formal Verification of Smart Contracts." In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 1891-1909. ACM, 2022. doi:10.1145/3572243.3572248.

6. Matteo Lusena, et al. "A Survey of Smart Contract Vulnerabilities." IEEE Security & Privacy, vol. 19, no. 3, pp. 84-93, 2021. doi:10.1109/MSP.2021.3080090.

7. Prateek Saxena, et al. "Sniper: A Static Analyzer for Vulnerabilities in Smart Contracts." In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 97-112. ACM, 2018. doi:10.1145/3243734.3243773.

8. Daniel Chen, et al. "Oyente: A Security Analysis Tool for Smart Contracts." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 258-272. ACM, 2017. doi:10.1145/3133956.3134008.

9. Abhishek Tiwari, et al. "Zeus: A Symbolic Execution Engine for Smart Contracts." In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1991-2008. ACM, 2020. doi:10.1145/3319535.3363285.