Project Group Members:

Name: Rehan Mumtaz
Roll No: TESE-19036
Email: mumtaz4203271@cloud.neduet.edu.pk

Name: Shiekh Muhammad Ahsan Tariq
Roll No: TESE-19054
Email: ahsantariq792@gmail.com

Name: Kabeer Ahmed
Roll No: TESE-19028
Email: ahmed4201750@cloud.neduet.edu.pk

Smart Security in Smart Contract

Rehan Mumtaz [1], Shiekh Muhammad Ahsan Tariq [2] ,Kabeer Ahmed [3]

[1,2,3] Department of Software Engineering, NED University of Engineering & Technology, Karachi, Pakistan

***Abstract*-Smart contracts are contracts for any kind of agreement, but what makes them smart is that they run as code automatically on a blockchain. They implement the blockchain's decentralization idea by automating transactions in the absence of any centralized control. Although smart contracts have advanced quickly, there are still a number of security issues that need to be explored. Loss in money is frequently caused by such weaknesses. In the present world, a fully automated solution to vulnerability detection is desired. We summarize the security challenges of smart contracts.It is merely a human-made, error-prone computer application. The majority of smart contract flaws are caused by security issues in lines of computer code.Future scholars will receive full analyses and recommendations from this work.We discussed in the paper of our automated and open source tool SmartCheck, a tool for aggressive analysis of vulnerability in a smart contract, the main point is the remediation technique it offers for a certain vulnerability and it severity. In order to identify vulnerability patterns, SmartCheck was created using XPath [xpa] searches on the intermediate representation (IR). Any code that has been translated to IR is protected by SmartCheck, and any items connected to it are found using XPath matching. We examine the methods and tools that developers can use to safeguard data against hacking.**

Keywords— smart contract; Ethereum; SmartCheck, security concerns; vulnerabilities; Xpath

I.Introduction

Blockchain was originally described in 1991 that was used for securing original digital documents. This technology remained unused for many years until in 2009 Satoshi Nakamoto and the first ever digital cryptocurrency was developed called Bitcoin. The blockchain technology has a property that once some data is recorded in them, it is difficult to alter them. Each block contains data with the hash of that block and previous blocks. The hash is unique which is created at the time of storing of data and if the data is changed its hash is also changed. All blocks contain the hash of previous blocks. So, if any block is tempered by all other blocks (nodes) in the network. Instead of a central server, Block chain uses peer-to-peer connection.

This technology is constantly evolving. One of its developments is the smart contracts. It means the code that runs on blockchain and it ensures an agreement between two parties. They are like normal contracts except they are digital and are programmed in a special programming language called solidity. We don't need to rely on third parties. It's just you and the person who is receiving your transactions within the decentralized networks of blockchain. Smart contracts can hold all the received funds until a certain goal is achieved. If the goal is achieved the money is transferred to the creator, if it fails then it is sent back to senders. But why should we trust smart contracts? Because it is immutable and distributed, means that it is not possible to alter the smart contract. Secondly, it is validated by everyone in the network. A part of crowdfunding, the smart contracts are used in insurance claims, postal to deliver payments, electronic voting systems etc. Nowadays there are a number of block chains that support smart contracts but Ethereum is the most popular and famous one. It is developed for creating smart contracts.

But in spite of a number of security measures there is also a risk of detrimental attacks. A wide range of attacks are made on blockchain technologies related to cryptocurrency, E-wallets, transactions, smart contracts etc. Some famous attacks on smart contracts are DAO attack and King of Ether Throne. So, the security factor should be properly focused to avoid any kind of vulnerabilities.

There are some tools available such as Oyente, Osiris, Remix. In this paper we discuss a security tool called SmartCheck. Its performance is more accurate as compared to the others. It is a tool for checking the security in smart contracts. It does not only detect the vulnerabilities but also identifies the cause of vulnerabilities along with details. It also provides some recommendations related to particular security issues. It uses XPath queries to detect the patterns of vulnerabilities. An experiment was held to check the efficiency of SmartCheck by providing around 4.5k contracts and it was successful in detecting vulnerabilities in 99.9%.

Following are the descriptions of sections in this research paper. In Section II the literature review of some research papers is discussed. In Section III the methodology to implement smart security in smart contracts is described. The Section IV, V contains Future Work and Conclusion.

II.RELATED WORK

This section involves reviews concentrated on smart contracts, its security and attacks which helps in investigating the security and vulnerabilities in smart contracts. Some reviews of related work are described as under:

The paper [1] presents some mistakes that are made in smart contracts and guide the people with some methods to prevent these mistakes. In this paper the students are targeted which are involved in the practices of smart contracts. The common mistakes include some programming, logical and security issues. The programming and logical errors commonly occurred in the encoding of complex state machines. The most common pitfall that occurs in smart contracts is the error that causes monetary leaks.

Smart contracts developed on blockchain have a huge impact on new businesses. It is a difficult task and started only in industrial and scientific fields. Wohrer et al. [2] presented security patterns with solutions based on grounded theory, applied on smart contract data. These solutions are based on Solidity which is a popular programming language used in block chain. These security patterns can be applied to solve the security issues to avoid common attack scenarios.

Some of the vulnerabilities along with attacks in the architecture layer of smart contracts are discussed in paper [3]. Three aspects which are vulnerabilities, attacks and defense are presented in this paper. It presents a survey of Ethereum security systems. It presents 40 types of vulnerabilities in the architecture of Ethereum and also provides its root causes. Some defense techniques are discussed. It also highlights a factor that using a better programming language can help in making contract fault tolerant but cannot make the contract secure.

A new way for the transactions of cryptocurrency is smart contracts but there are a lot of vulnerabilities in this technique. An exploitable bug can cause a loss of a huge amount of money. A most common type of bug known as reentrancy bug is discussed in paper. This bug caused an attack known as DAO (Decentralized Autonomous Organization) in 2016, which resulted in a debt of $60 million worth of Ethereum cryptocurrency. Liu [4] presents a technique called *ReGuard* for the detection of bugs. It is a fuzzing-based technology that performs fuzz testing by generating random transactions. It identifies vulnerabilities and automatically flags them.

A framework known as Osiris, is presented in research [5]. This technique is designed to find bugs and errors in smart contracts. This tool works efficiently and detects a wider range of errors than the tools available. An experiment was performed to evaluate its performance, a large dataset is provided that contains around 1.2 million smart contracts. Smart contracts are being used in business and dealing with intellectual properties. But due to certain vulnerabilities in smart contracts many issues are reported which causes great financial losses. Many solutions have also been developed. After analyzing many researches Y Haung [6] addresses some vulnerabilities and like traditional software development lifecycle (SDLC) the contract lifecycle are also divided into four stages.

Checking the mechanism of transaction, seemed to require much attention as it is a critical function in smart contracts when it is dealing with live transfer of money. This makes it critical as to secure this requires different techniques. Some go for manual while some for automated prevention technique,while in **[7]** Jemin Andrew came up with idea of run time validation that found to be a strategic and efficient approach.However there is a downfall for this method which is overhead of runtime validation(excessive performance overhead),which can be costly for domains in blockchain. This issue can be integrated with Solythesis, tool of runtime validation for smart contracts(Ethereum), it uses as source compiler tool for solidity and analyzing/detect of error at compile time, however it also facilitates expressive or mechanism language(which uses quantifiers for allowing users to identify important functions of smart contracts)

In a research study, a method posted in **[9]**about checking of smart contract if is it exhibits vulnerability by using model checking and adopting rules of μ-Calculus taking care of vulnerabilities normally termed as Automata. It was thus evaluated on a dataset and predicting it through a precision and recall rate whether it is vulnerable or not. While other researchers gave their approach, one of the most maximax mechanisms that found to be useful is analyzing access control mechanisms in blockchain as it can lead to deadliest bugs that can be exposed to a greater harm for an entity. In **[8]** the paper, researcher think of this idea of using High Granularity Metrics scheme,building white and blacklisting access control in an adaptive manner to filter out unwanted and unauthorized pickling with data.Thus we create many layers for learning, identifying, and alerting unusual behavior by increasing visibility through aggregated feature-level measurements.Another report by researchers shows in their paper is **[10]**,the introduction of tool Ethainter, a security analytics tool that cleans data from smart contracts to verify the information flow. Ethainter identifies fusion attacks that result in significant breaches by spreading contaminated information across numerous transactions. The analysis covers the entire blockchain, which consists of millions of accounts and hundreds of thousands of different smart contracts. We validate that Ethainter is more accurate than earlier methods for autonomous mining generation.

Researchers nowadays always look for a way to secure the transactions carried out in blockchain due to the critical nature.False positive is an issue usually addressed by many to use methods to overcome it but still it persists which If the problem is either not present or cannot be exploited. In one of the studies**[11]**,a formal approach part work is presented to address that issue. Our work builds upon our earlier work on two levels: first, by taking the notion into consideration. Focus on the switch's function calls and LTL attributes in order to assess the accuracy of the smart contract. These characteristics may be unique to the management or information flow of the contracts under verification. By suggesting formal characteristics ofLTL for vulnerabilities from the literature, we show that they can also be utilized to reveal flaws.In another study,**[12]** another tool HFContract-Fuzzer is introduced ,a Fuzzing-based approach that combines a go-fuzz-based Fuzzing engine with go-written smart contracts to detect Hyperledger Fabric smart contracts. We used HFContractFuzzer to find security flaws in five contracts from common sources and identified that four of them had vulnerabilities, demonstrating the efficacy of the suggested strategy.

In the paper **[13]**, authors present a number of fresh security issues where a malicious party might influence the execution of a smart contract in order to profit. These flaws point to minute weaknesses in the platform's underlying distributed semantics knowledge. They suggest approaches to improve Ethereum's operational semantics in order to make contracts more secure. They created a symbolic execution tool called Oyente to help contract writers for the current Ethereum system detect possible security flaws. Oyente classifies 8, 833 of the 19, 366 currently in use Ethereum contracts as insecure, including the TheDAO flaw that cost the company $60 million in June 2016. Several case studies with published source code that confirm the assaults in the primary Ethereum network are also discussed, along with the severity of additional attacks for each.

Studies have shown security flaws, faults, and vulnerabilities in the Ethereum smart contract. The only method to end a contract on the blockchain system and transfer all of the Ethers in the contract balance is to use the Self Destruct function. As a result, when issues are found, many developers utilize this function to terminate the contract and deploy a new one. A deep learning-based algorithm was used in one study **[14]** to retrieve the updated version of a destroyed contract in order to identify security flaws in Ethereum smart contracts. Then, using open card sorting, we look for security flaws in the upgraded versions. In another study **[15]** Authors personally found their defined contract flaws in 587 actual smart contracts by examining Feedback; they then made their dataset available to the public. Finally, they listed five effects brought on by contract flaws. These aid developers in comprehending the problems' symptoms and order of importance for elimination. Another study's say **[16]** In addition to providing a thorough taxonomy of all known security concerns and reviewing the security code analysis techniques used to spot known vulnerabilities, the authors explore the topic of security of smart contract programming. They investigate security code analysis tools on Ethereum by evaluating their efficacy and precision on known problems on a representative sample of weak contracts. They tested the effectiveness of four security tools—Oyente, Securify, Remix, and SmartCheck—as well as 21 secure and 24 susceptible contracts to determine how well-rounded modern security analysis tools are for Ethereum.

REFERENCES

[1] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, “Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab,” in Proc. 20th Int. Conf. Financial Cryptography Data Security, 2016, pp. 79–94.

[2] M. Wohrer and U. Zdun, “Smart contracts: Security patterns in the Ethereum ecosystem and Solidity,” in Proc. Int. Workshop Blockchain Oriented Softw. Eng., 2018, pp. 2–8

[3] H. Chen, M. Pendleton, L. Njilla, and S. Xu, ``A survey on Ethereum systems security,'' *ACMComput. Surv.*, vol. 53, no. 3, pp. 1_43, Jul. 2020, doi: 10.1145/3391195.

[4] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, ``ReGuard: Finding reentrancy bugs in smart contracts,'' in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. Companion*, May 2018, pp. 65_68.

[5] C. F. Torres, J. Schütte, and R. State, ``Osiris: Hunting for integer bugs in ethereum smart contracts,'' in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, Dec. 2018, pp. 19.

[6] Y. Huang, Y. Bian, R. Li, J. L. Zhao and P. Shi, "Smart Contract Security: A Software Lifecycle Perspective," in IEEE Access, vol. 7, pp. 150184-150202, 2019, doi: 10.1109/ACCESS.2019.2946988.

[7] Ao Li, Jemin Andrew Choi, and Fan Long. 2020. Securing smart contracts with runtime validation. In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020). Association for Computing Machinery, New York, NY, USA, 438–453. https://doi.org/10.1145/3385412.3385982

[8]Venkata Siva Vijayendra Bhamidipati, Michael Chan, Derek Chamorro, Arpit Jain, and Ashok Murthy. 2019. Adaptive Security for Smart Contracts using High Granularity Metrics. Proceedings of the 3rd International Conference on Vision, Image and Signal Processing. Association for Computing Machinery, New York, NY, USA, Article 83, 1–6. https://doi.org/10.1145/3387168.3387214

[9]Giuseppe Crincoli, Giacomo Iadarola, Piera Elena La Rocca, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2022. Vulnerable Smart Contract Detection by Means of Model Checking. In Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI '22). Association for Computing Machinery, New York, NY, USA, 3–10. https://doi.org/10.1145/3494106.3528672

[10]Lexi Brent, Neville Grech, Sifis Lagouvardos, Bernhard Scholz, and Yannis Smaragdakis. 2020. Ethainter: a smart contract security analyzer for composite vulnerabilities. In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020). Association for Computing Machinery, New York, NY, USA, 454–469. https://doi.org/10.1145/3385412.3385990

[11]Ikram Garfatta, Kaïs Klai, Mohamed Graïet, and Walid Gaaloul. 2022. Model checking of vulnerabilities in smart contracts: a solidity-to-CPN approach. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22). Association for Computing Machinery, New York, NY, USA, 316–325. https://doi.org/10.1145/3477314.3507309

[12]Mengjie Ding, Peiru Li, Shanshan Li, and He Zhang. 2021. HFContractFuzzer: Fuzzing Hyperledger Fabric Smart Contracts for Vulnerability Detection. In Evaluation and Assessment in Software Engineering (EASE 2021). Association for Computing Machinery, New York, NY, USA, 321–328. https://doi.org/10.1145/3463274.3463351

[13] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. 2016. Making Smart Contracts Smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 254–269.

[14] J. Chen, "Finding Ethereum Smart Contracts Security Issues by Comparing History Versions," 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020, pp. 1382-1384.

[15] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo and T. Chen, "Defining Smart Contract Defects on Ethereum," in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 327-345, 1 Jan. 2022, doi: 10.1109/TSE.2020.2989002.

[16] R. Sujeetha and C. A. S. Deiva Preetha, "A Literature Survey on Smart Contract Testing and Analysis for Smart Contract Based Blockchain Application Development," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 378-385, doi: 10.1109/ICOSEC51865.2021.9591750.