Smart Contract verification process focusing on IoT applications

Joyce Quintino, Carina T. de Oliveira, Rossana M. C. Andrade

Smart Contract verification process focusing on IoT applications

JOYCE QUINTINO
joycequintinoalves@alu.ufc.br
Federal University of Ceara
Fortaleza, Ceara, BRA

CARINA T. DE OLIVEIRA
carina@lar.ifce.edu.br
Federal Institute of Ceara
Fortaleza, Ceara, BRA

ROSSANA ANDRADE*
rossana@ufc.br
Federal University of Ceara
Fortaleza, Ceara, BRA

ABSTRACT

Blockchain has emerged as a promising technology for improving security in the Internet of Things (IoT), as it makes it possible to record data in a decentralized, encrypted, and immutable way with the consensus of network participants. In IoT applications using Blockchain, smart contracts can eliminate the need for intermediaries or third-party trust, enabling more secure and transparent data transfers between the parties involved in a decentralized manner. Smart contracts are subject to flaws in terms of logic and vulnerabilities that can result, for example, in financial losses or compromise the integrity of IoT devices, causing risks to users, privacy, and security. Given this, conducting comprehensive tests using different analyses before deployment can expose errors in the smart contract code and reduce security risks. However, no solution uses a combination of different analyses to increase the accuracy of detecting vulnerabilities in smart contracts. This paper proposes a process that defines a set of smart contract verification steps for IoT applications to detect vulnerabilities using a combination of static and dynamic analysis tools before deployment. As a result, the work is currently in the development and testing steps of the proposed process.

KEYWORDS

Internet of Things, Security, Blockchain, Smart Contracts.

1 INTRODUCTION

The Internet of Things (IoT) is a network of objects and embedded devices connected to the Internet. As the number of connected devices increases, so does the amount of data traveling through the network, so a secure and reliable IoT environment becomes essential to avoid security vulnerabilities [8].

However, IoT has limitations that challenge the development of a secure and reliable environment. For example, the amount of data generated by applications attracts attackers who can steal this data or alter it. Much of this data is sensitive and requires privacy protection [12].

Authentication is a security requirement for protecting IoT devices, which is crucial in restricting access to resources and data to known and trusted devices only. In this sense, only authorized devices are recommended to access these resources and data. However, it is essential to note that the issue of authentication in the context of the IoT is still a challenge discussed in the literature [5].

Blockchain is emerging as a promising technology to provide trust, privacy, and confidentiality in the IoT. This technology allows data to be recorded, decentralized, encrypted, and immutable, operating by consensus between the participants. This ensures data integrity and prevents manipulation or unauthorized access, making it a trustworthy environment [14].

In IoT applications using Blockchain, Smart Contracts (SC) can be used to build automation and control systems for IoT devices. The SCs are programs with defined rules, autonomous and self-executing, capable of eliminating the need for intermediaries or third-party trust, allowing for more secure and transparent data transfers between the parties involved in a decentralized manner. The SCs can also be used to implement authentication logic in IoT applications using *Blockchain*, and this allows the automatic execution of authentication rules without the need for intermediaries [1].

The CSs are subject to failures in terms of logic and vulnerabilities. Some causes of vulnerabilities in SCs are programming errors, specification flaws, and implementation gaps. Thus, security becomes a challenge for the secure development of SCs, and existing vulnerabilities can result, for example, in financial losses or compromise the integrity of IoT devices, causing risks to the privacy and security of users [11].

To contribute to identifying vulnerabilities in smart contracts, this paper proposes a process that defines a set of steps for verifying smart contracts to detect vulnerabilities using a combination of static and dynamic analysis tools. This process should be applied before the smart contract implementation step.

The main contributions of this paper are listed below:

- It elaborates on research on testing smart contracts to avoid security risks.
- Studies on integrating smart contracts into the IoT to provide more direct authentication without dependence on third parties.
- Describes the smart contract verification process proposed in this paper.

In addition to this Section 1, this paper is organized as follows: Section 2 describes the background. Section 3 describes the related works. Next, Section 4 details the smart contract verification process. The final considerations are presented in Section 5.

2 BACKGROUNG

This section presents concepts about Blockchain, Smart Contracts (SC), SC vulnerabilities, and vulnerability detection tools to facilitate understanding of this paper.

2.1 Blockchain

Blockchain is a distributed database technology that emerged from the creation of Bitcoin by Satoshi Nakamoto in 2008 [9]. This technology has features that allow data to be recorded in a decentralized, encrypted, immutable way with the consensus of network participants using a packet data structure known as a block [2]. These resources are defined below:

*CNPq's Productivity Scholarship in Technological Development and Innovative Extension - DT - Level 1D

- **Decentralization:** has no central coordinating entity, or group, with powers to revert data or change consensus rules.
- **Cryptography:** Blockchain has public and private key encryption. The public key is common to all network participants, while the private key is exclusive to each network member. These keys together unlock the data.
- **Immutability:** data cannot be changed once registered on the network.
- **Consensus:** establishes rules on the consent of network participants. Data registration is only possible with the consent of the majority of participants.

Blockchain is divided into three main types: public, private, and consortium. Bhutta [2] defines these three types of Blockchain:

- **Public:** does not need permission to participate in this type of *Blockchain*. All participants have equal rights to participate in the consensus process, read, edit, and validate.
- **Private:** Unlike public, in private *Blockchain* only authorized participants can join and maintain the network. This type of Blockchain is considered more secure than the public one. However, participants are not anonymous.
- **Consortium:** Also private but is intended for multiple organizations. Only invited and trusted participants are allowed to join and maintain the network.

2.2 Smart Contracts

The Smart Contracts (SC), first defined by [13], represent "a set of promises, specified in digital format, including protocols in which the parties fulfill these promises." In other words, they are self-executing digital contracts established between two or more parties. The clauses are executed through rules defined in the code of a smart contract from the moment the established contractual conditions are met. The main objective of SCs is to enable a safer means of exchanging data between the parties involved in a decentralized way without the need for intermediaries or third-party trust. [1].

2.3 Smart Contract Vulnerabilities

Despite the security benefits SCs provide, there are vulnerabilities specific to SCs. In addition, integrating SCs with the IoT environment creates security challenges due to the decentralized nature and the many devices involved. Some specific vulnerabilities that can arise in smart contracts include:

- **Reentrancy Attacks:** Contracts call other contracts before finishing their execution, allowing a malicious contract to execute code in another contract, potentially changing states unexpectedly.
- **Overflow and Underflow:** Arithmetic operations can result in numbers more significant than the maximum representable (overflow) or smaller than the minimum representable (underflow).
- **Order and Dependency Problems:** The behavior of the contract depends on the order in which the operations are executed or on specific events.
- **Unauthenticated Messages:** The contract accepts unauthenticated messages, allowing falsified data to be entered.
- **Failures in Permission Management:** Inappropriate permissions granted to users or contracts, allowing unauthorized access.
- **Failures in Key Management:** Private keys used to control contracts are not stored or managed correctly.
- **Gas Limit and Out-of-Gas Attacks:** Attacks that exhaust the gas limit available for code execution. In IoT, device operations can be interrupted due to a lack of computing resources.

2.4 Vulnerability Detection Tools

Several vulnerability detection tools for smart contracts help developers identify and fix potential security issues [10]. For this research, we selected some of the most popular tools to take part in the experiments, they are:

- **Oyente**[1]**:** A symbolic analysis tool for Ethereum smart contracts that allows exploration of execution paths and vulnerability detection.
- **Slither**[2]**:** A static analysis tool for Solidity smart contracts that detect known vulnerabilities, security patterns, and style issues.
- **Mythril**[3]**:** is a symbolic analysis tool used to discover vulnerabilities in smart contracts in the Ethereum environment, capable of detecting several known vulnerabilities.
- **Securify**[4]**:** An Ethereum smart contract scanning tool that uses static analysis techniques to find vulnerabilities.

3 RELATED WORKS

The related work in this research focuses on services, processes, and tools developed to help detect vulnerabilities in smart contracts.

In [7], a testing service called Fuse is presented, which uses fuzzification techniques to test smart contract codes. The service generally focuses on the *black-box* testing category. The results obtained by the authors show that the service detected Ethereum vulnerabilities with true positive rates of 96-100%. The authors have also made the service available as a tool called ContractFuzzer [4].

The work [3] classifies smart contract vulnerability detection tools. Smart contract detection tools can be divided into static and dynamic detection. To analyze each detection tool, the authors also defined three requirements: type of vulnerability, detection accuracy, and ability to detect the version of the smart contract.

In [6], it uses the metamorphic testing technique to detect vulnerabilities in smart contracts. This testing technique is based on properties and is used to mitigate the oracle problem of software testing. The experiments evaluated 67 smart contracts and used the tools ContractFuzzing, Slither, and Mythril to detect smart contract vulnerabilities. The authors used the following metrics to evaluate their approach: true positives, false negatives, and false positives.

The authors emphasize in their paper [7] the need to improve the vulnerability detection service after detecting only 50% of vulnerabilities when compared to the tool *Oynete*, and only work for offline analysis, making the service inefficient in some instances.

[1]https://github.com/enzymefinance/oyente
[2]https://github.com/crytic/slither
[3]https://github.com/ConsenSys/mythril
[4]https://github.com/eth-sri/securify2

Another work, described in [3], compares different vulnerability detection tools. However, the authors do not present a detailed process that could serve as a guide for other professionals interested in the comparative analysis of these tools.

In [6], the authors adopt an automated testing approach based on Artificial Intelligence, providing a new perspective for detecting vulnerabilities in smart contracts. However, the results of false positives and false negatives are 0. In contrast to related work, this paper introduces a promising process for standardizing smart contract testing.

4 SMART CONTRACT VERIFICATION PROCESS

Figure 1 illustrates the steps of the proposed process, which will be presented and discussed below.

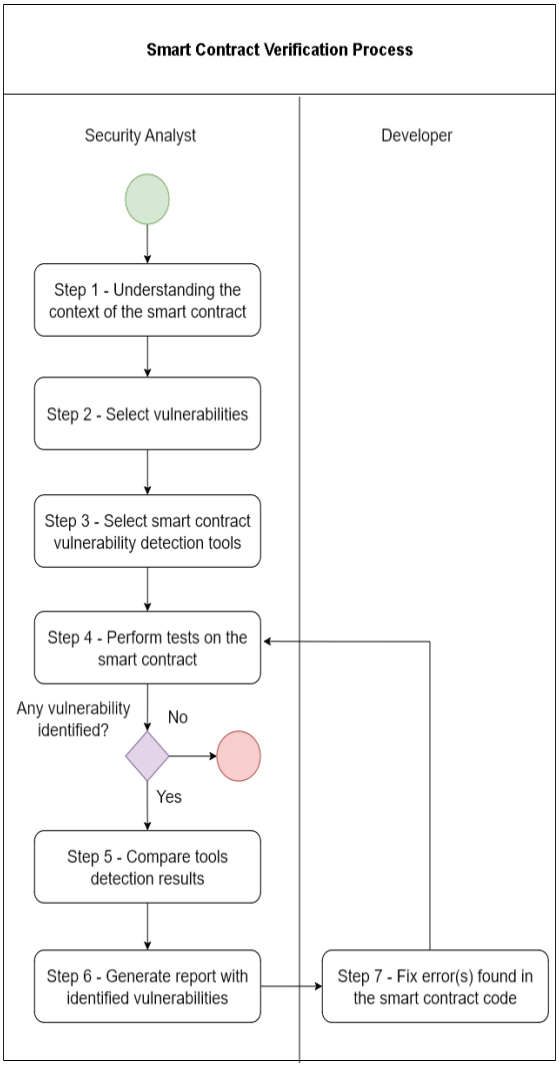

Figure 1: Smart Contract Verification Process.

4.1 Step 1 - Understanding the context of the smart contract:

In the first step of this process, it is essential to understand the smart contract's business logic, functionalities, and interactions with other smart contracts or system components before beginning testing. During this step, the security analyst, the main actor in this process, must understand how the smart contract was programmed, its purpose, and whether it accesses other contracts during its execution on the Blockchain.

4.2 Step 2 - Select vulnerabilities:

In the second step, we seek to identify and select the vulnerabilities in the smart contract, which will be thoroughly analyzed during the experiments. These vulnerabilities can cover several categories, such as those mentioned in the vulnerabilities subsection 2.3. Carefully choosing which vulnerabilities to examine is critical to an effective testing process and identifying potential threats to smart contract security.

4.3 Step 3 - Select smart contract vulnerability detection tools:

In this step, the selection of vulnerability detection tools in smart contracts, such as Oyente, Slither, Mythril, and Securify, is carried out. After selecting the tools, creating an experimentation environment for testing in the subsequent step is essential.

4.4 Step 4 - Perform tests on the smart contract:

In this step, creating an experimentation environment to test the smart contract is necessary. The experimentation environment will include installing the tools selected in the previous step. Next, the selected tools will analyze the smart contract to identify the vulnerabilities. If no vulnerabilities are identified, the process ends; otherwise, the process moves on to compare the results of the tools step.

4.5 Step 5 - Compare tools detection results:

After testing in step 4 is completed, the detection results from each tool will be analyzed and compared to generate a final report. This report will be transmitted to developers to implement possible corrections to the contract.

4.6 Step 6 - Generate report with identified vulnerabilities:

This step involves preparing a report containing all the specifications of the tests carried out on the smart contract and the vulnerabilities identified.

4.7 Step 7 - Fix error(s) found in the smart contract code:

Step 7 involves the developers correcting the contract code once the vulnerability has been identified. If the corrections are not implemented, the process must remain at this step until the developers make the necessary corrections. Once the corrections have been completed, it is necessary to return to step 4, where the tests are repeated, ensuring that the smart contract has no vulnerabilities.

5 FINAL REMARKS

The SC verification process is in the development step. A study has been carried out on vulnerability detection tools in smart contracts. The work is currently in the development and testing steps of the proposed process. The following steps of this work are to implement an SC for the authentication of IoT devices and create the experimentation environment for the tools that are part of the proposed process. Then, follow the process steps and obtain concrete results by detecting vulnerabilities.

REFERENCES

[1] Jauberth Abijaude, Fabíola Greve, and Péricles Sobreira. 2021. *Blockchain e Contratos Inteligentes para Aplicações em IoT, Uma Abordagem Prática.* 149–197. https://doi.org/10.5753/sbc.6757.3.4

[2] Muhammad Nasir Mumtaz Bhutta, Amir A. Khwaja, Adnan Nadeem, Hafiz Farooq Ahmad, Muhammad Khurram Khan, Moataz A. Hanif, Houbing Song, Majed Alshamari, and Yue Cao. 2021. A Survey on Blockchain Technology: Evolution, Architecture and Security. *IEEE Access* 9 (2021), 61048–61073. https://doi.org/10.1109/ACCESS.2021.3072849

[3] Daojing He, Rui Wu, Xinji Li, Sammy Chan, and Mohsen Guizani. 2023. Detection of Vulnerabilities of Blockchain Smart Contracts. *IEEE Internet of Things Journal* 10, 14 (2023), 12178–12185. https://doi.org/10.1109/JIOT.2023.3241544

[4] Bo Jiang, Ye Liu, and W. K. Chan. 2018. ContractFuzzer: Fuzzing Smart Contracts for Vulnerability Detection *(ASE '18)*. Association for Computing Machinery, New York, NY, USA, 259–269. https://doi.org/10.1145/3238147.3238177

[5] Habib Ullah Khan, Mourade Azrour, Jamal Mabrouki, Azidine Guezzaz, and Ambrina Kanwal. 2021. Internet of Things Security: Challenges and Key Issues. (2021). https://doi.org/10.1155/2021/5533843

[6] Jiahao Li. 2023. Metamorphic Testing for Smart Contract Vulnerabilities Detection. arXiv:2303.03179 [cs.SE]

[7] Xiupei Mei, Imran Ashraf, Bo Jiang, and W.K. Chan. 2019. A Fuzz Testing Service for Assuring Smart Contracts. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 544–545. https://doi.org/10.1109/QRS-C.2019.00116

[8] Roberto Minerva, Abyi Biru, and Domenico Rotondi. 2015. Towards a definition of the Internet of Things (IoT). *IEEE Internet Initiative* 1, 1 (2015), 1–86.

[9] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[10] Fabricio Ismael Leyes Ontivero. 2023. Um estudo comparativo de ferramentas de auditoria em contratos inteligentes. https://repositorio.ufu.br/handle/123456789/37299

[11] Kai Peng, Meijun Li, Haojun Huang, Chen Wang, Shaohua Wan, and Kim-Kwang Raymond Choo. 2021. Security Challenges and Opportunities for Smart Contracts in Internet of Things: A Survey. *IEEE Internet of Things Journal* 8, 15 (2021), 12004–12020. https://doi.org/10.1109/JIOT.2021.3074544

[12] Shivam Saxena, Bharat Bhushan, and Mohd Abdul Ahad. 2021. Blockchain based solutions to secure IoT: Background, integration trends and a way forward. *Journal of Network and Computer Applications* 181 (2021), 103050. https://doi.org/10.1016/j.jnca.2021.103050

[13] Nick Szabo. 1997. Formalizing and Securing Relationships on Public Networks. *First Monday* 2, 9 (Sep. 1997). https://doi.org/10.5210/fm.v2i9.548

[14] Shivani Wadhwa, Shalli Rani, Kavita, Sahil Verma, Jana Shafi, and Marcin Wozniak. 2022. Energy Efficient Consensus Approach of Blockchain for IoT Networks with Edge Computing. *Sensors* 22, 10 (2022). https://doi.org/10.3390/s22103733