솔라나 스마트 계약의 취약점 연구

바게리 마흐부베[1], 박성환[2], 권동현[3*]
[1]부산대학교 정보컴퓨터공학과 학부생
[2]부산대학교 정보융합공학과 박사과정
[3]부산대학교 정보컴퓨터공학과 교수

mahyool@pusan.ac.kr, starjara@pusan.ac.kr, kwondh@pusan.ac.kr

Study of Vulnerabilities in Solana Smart Contracts

Mahboubeh Bagheri[1], Seong-Hwan Park[2], Dong-Hyeon Kwon[3*]
[1]Dept. of Computer Science and Engineering, Pusan National University
[2]Dept. of Information Convergence Engineering, Pusan National University
[3]Dept. of Computer Science and Engineering, Pusan National University

Abstract

Solana, a high-performance blockchain platform, is known for its fast transaction speeds and low operational costs, making it a popular choice for decentralized applications. However, its architecture introduces unique security vulnerabilities in smart contracts. This paper presents an analysis of six key vulnerabilities in Solana smart contracts—missing ownership checks, missing signer checks, arithmetic overflow/underflow, cross-program invocation (CPI) vulnerabilities, account confusion, and missing key checks. We further evaluate how automated verification tools like VRust and fuzzing techniques detect these vulnerabilities. Through case studies of widely-used Solana programs like Mango Markets and the Solana Program Library (SPL), we illustrate the effectiveness of these tools in real-world scenarios.

1. Introduction

Solana has emerged as a strong competitor to Ethereum due to its low transaction costs and high throughput, with the ability to process thousands of transactions per second (TPS) [1]. However, the platform's architecture introduces specific security vulnerabilities in its smart contracts. Identifying and addressing these vulnerabilities is crucial to the long-term security of the Solana ecosystem. This paper provides an in-depth analysis of six critical vulnerabilities and assesses how automated verification tools such as VRust and fuzzing techniques can detect these vulnerabilities [2]. We also incorporate real-world case studies to validate the effectiveness of these tools.

2. Common Vulnerabilities in Solana Smart Contracts

Solana's smart contracts face unique security challenges due to its architectural choices. The following are six prevalent vulnerabilities found in Solana smart contracts:

2.1 Missing Ownership Checks

In Solana, accounts are governed by a program ID that manages data access. If smart contracts fail to verify whether an account is owned by the intended program, unauthorized access and data manipulation may occur [3].

Detection: VRust has shown effectiveness in identifying missing ownership checks through static analysis, ensuring that contracts verify account ownership during development.

2.2 Missing Signer Checks

Signer checks validate that the account executing sensitive operations, such as fund transfers, is authorized to do so. When signer checks are omitted, unauthorized entities may carry out critical operations [3].

Detection: VRust is highly effective in detecting missing signer checks due to its static nature, but **FuzzDelSol** is better at catching complex scenarios involving signer authorization through stress testing.

2.3 Cross-Program Invocation (CPI) Vulnerabilities

Solana supports cross-program invocations (CPI), allowing one program to call another. If validation is not properly implemented, malicious programs may be invoked via Program Derived Addresses (PDAs), which can lead to significant exploits [1].

Detection: FuzzDelSol excels in uncovering CPI-related vulnerabilities, especially under conditions where static analysis may not detect dynamic behaviors.

2.4 Arithmetic Overflow/Underflow

Unchecked arithmetic operations, particularly in financial applications, can result in overflow or underflow. Although Rust's debug mode detects these issues, in release mode, safety checks are often disabled for performance, leaving room for exploitation [2].

Detection: VRust catches arithmetic issues in debug mode, while **FuzzDelSol** can stress-test the system to reveal overflows/underflows that occur during runtime.

2.5 Account Confusion

This occurs when smart contracts fail to differentiate between user-provided accounts and trusted program accounts. Attackers may supply malicious accounts that are mistakenly treated as trusted [1].

Detection: FuzzDelSol is more effective in detecting account confusion vulnerabilities, particularly when random data triggers unexpected program behaviors.

2.6 Missing Key Checks

When a program fails to verify that a specific account is the intended one, security breaches like the Wormhole attack, which resulted in a $320 million loss, can occur [4].

Detection: VRust is suitable for detecting missing key checks by verifying contract logic, but **FuzzDelSol** can also catch this issue dynamically under extreme input conditions.

3. Automated Verification Techniques for Detecting Vulnerabilities

Automated tools play a vital role in ensuring the security of Solana smart contracts. We focus on two prominent techniques **static analysis (VRust)** [3] and **fuzzing (FuzzDelSol)** and their effectiveness in detecting the vulnerabilities listed above.

3.1 Static Analysis in Rust (VRust)

VRust performs static analysis on Solana programs, using Rust's Mid-level Intermediate Representation (MIR) to preemptively catch errors during development [3]. VRust is particularly strong at detecting missing ownership checks, missing signer checks, and arithmetic overflows, but it may struggle with dynamic vulnerabilities like account confusion or CPI issues.

3.2 Fuzzing Techniques

Fuzzing introduces random data into smart contracts to trigger unexpected behaviors. Tools like FuzzDelSol stress-test contracts by supplying unpredictable inputs, making them highly effective in detecting runtime issues such as CPI vulnerabilities, account confusion, and arithmetic overflow/underflow [1]. This method is especially useful for dynamic vulnerability detection, which is not always possible with static analysis.

4. Case Studies and Results

We examine two real-world case studies to evaluate the effectiveness of the above-mentioned automated tools in detecting vulnerabilities in Solana smart contracts.

4.1 Solana Program Library (SPL)

The Solana Program Library (SPL) is a collection of programs frequently used in the ecosystem. **VRust** was used to analyze SPL for missing ownership and signer checks, successfully detecting these vulnerabilities in early versions of the program. However, fuzzing with **FuzzDelSol** revealed additional vulnerabilities related to CPIs and arithmetic overflow, which static analysis tools missed [3]. This highlights the complementary nature of these tools, with VRust excelling at static issues and FuzzDelSol uncovering dynamic vulnerabilities.

4.2 Mango Markets and Metaplex

Mango Markets and Metaplex are custom programs that were analyzed for vulnerabilities such as missing signer checks and unchecked arithmetic operations. **VRust** successfully detected missing signer checks, while **FuzzDelSol** uncovered additional issues with arithmetic overflow/underflow that were not caught by static analysis. Moreover, the fuzzing approach revealed potential CPI vulnerabilities that could be exploited under specific conditions, further illustrating the importance of combining static and dynamic analysis tools [1].

5. Conclusion

The security of Solana smart contracts is crucial for the platform's continued growth [4]. Our analysis shows that while tools like **VRust** are effective at detecting static vulnerabilities like missing ownership checks and signer checks, dynamic analysis techniques like **FuzzDelSol** are essential for uncovering runtime vulnerabilities such as CPIs and account confusion. The combined use of these tools provides a robust defense against the wide range of vulnerabilities present in Solana smart contracts. Future work should focus on enhancing these tools to address emerging security threats as the Solana ecosystem continues to evolve [5].

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Special R&D Zone Development Project (R&D) - Development of R&D Innovation Valley support program (2023-DD-RD-0152), supervised by the Innovation Foundation.

References

[1] Sven Smolka, Jens-Rene Giesen, Pascal Winkler, Oussama Draissi, Lucas Davi, Ghassan Karame, Klaus Pohl. "Fuzz on the Beach: Fuzzing Solana Smart Contracts." Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), Copenhagen, Denmark, 2023, pp. 1197-1211.

[2] Tien N. Tavu. Automated Verification Techniques for Solana Smart Contracts. Undergraduate Research Scholars Program, Texas A&M University, 2022.

[3] Siwei Cui, Gang Zhao, Yifei Gao, Tien Tavu, Jeff Huang. "VRust: Automated Vulnerability Detection for Solana Smart Contracts." Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), Los Angeles, USA, 2022, pp. 639-652.

[4] Sébastien Andreina, Tobias Cloosters, Lucas Davi, Jens-Rene Giesen, Marco Gutfleisch, Ghassan Karame, Alena Naiakshina, Houda Naji. "Defying the Odds: Solana's Unexpected Resilience in Spite of the Security Challenges Faced by Developers." arXiv preprint, arXiv:2406.05231 [cs.CR], 2024.

[5] Fang, J., & Qu, H. "VeriOover: A Verifier for Detecting Integer Overflow by Loop Abstraction." Proceedings of the 13th International Workshop on Computer Science and Engineering (WCSE 2023), Ocean University of China, Qingdao, China, 2023.