28. The Dual Edges of AI in Cybersecurity: Insights from the 2024 Benchmark Survey Report / Hyperproof. Seattle, WA : Hyperproof, 2024. https://hyperproof.io/resource/ai-in-cybersecurity-2024-benchmark-report/ .
29. What is Operational Technology (OT) Cyber Security?. https://www.axians.co.uk/glossary/what-is-operational-technology-security/#:~:text=Operational%20Technology%20(OT)%20security%2C,vulnerabilities%20from%202017%20to%202020.

Надійшла 17.09.2025

УДК 004.056.5:004.415.53
DOI: 10.31673/2409-7292.2025.041204

Hashko A.O, Bondarchuk A.P.

AUTOMATED METHOD FOR VERIFYING THE CORRECTNESS OF THE EXECUTION OF SMART CONTRACTS IN THE BLOCKCHAIN NETWORK

The article examines an automated method for verifying the correctness of smart contracts in the Solana blockchain network. The relevance of the research is driven by the growing popularity of Web3 applications and the need to ensure their security, as even minor errors in smart contract code can lead to significant financial losses. The primary goal is to develop an automated verification methodology for smart contracts that can detect vulnerabilities such as the absence of founder rights verification, arithmetic operation errors, and missing transaction check signatures. Using static analysis techniques in the Rust programming language, the authors propose an approach that enables rapid analysis-taking less than three minutes per contract-and automatic generation of reports on identified vulnerabilities. The methodology is based on analyzing external data flows through smart contracts, allowing for the early detection of potential threats. To automate the process, Python and Bash scripts are employed, integrating with cloud services such as Amazon Web Services to scale the analysis. Testing results on real Web3 applications demonstrate the effectiveness of the methodology, particularly in reducing analysis time and improving the accuracy of error detection. An important aspect of the research is the continuous updating of knowledge bases and analysis tools, enabling the consideration of new types of attacks and vulnerabilities. The article also highlights the importance of interoperability between different blockchain networks, which remains a challenging task but is a key element for the future development of Web3. The research results show that the proposed methodology is promising for scaling and adapting to new challenges in blockchain ecosystems such as Solana. Thus, the developed approach to automated smart contract verification not only enhances the security of Web3 applications but also contributes to their further development, ensuring stability and reliability in the dynamic evolution of blockchain technologies.

Keywords: blockchain, smart contract, Solana, information system, Rust, automated verification, optimization, security, decentralization.

Introduction

Despite optimistic forecasts in IT research that reveal blockchain's potential for creating decentralized societal structures, the practical implementation of these ideas faces significant challenges. A key obstacle to mass adoption is the vulnerabilities and errors inherent in the blockchain networks themselves and, specifically, in smart contract mechanisms [1-2].

In contrast to first-generation blockchains like Bitcoin, which are limited in functionality, second and third-generation platforms (Ethereum, Solana, Cardano) offer more complex capabilities. However, their complexity introduces new risks. For instance, the shift to consensus algorithms like Proof-of-Stake (PoS), which underlies the smart contracts of Ethereum and Solana, while solving the energy efficiency problems of its predecessor (Proof-of-Work), introduces new threats such as 51% attacks (in PoS) or the complexity of ensuring network security under delegated staking conditions [5-6].

The main threat lies in the implementation of smart contracts. Their core idea-the automatic execution of agreement terms without intermediaries-is simultaneously their main "Achilles' heel." The most dangerous errors are:

1. Logical Vulnerabilities: The contract's source code may contain errors that allow malicious actors to steal funds or disrupt the agreement's logic. History knows numerous cases of million-dollar losses due to such bugs;

© Гайдур Г.І., Гахов С.О., Скибун О.Ж. (2025) Штучний інтелект у кібербезпеці критичної інфраструктури: підходи, оцінка стану та перспективи розвитку. Сучасний захист інформації, 4(64), 24–37. https://doi.org/10.31673/2409-7292.2025.041203

2. Scalability and Performance Issues: Networks positioned as high-speed (Solana with its hybrid Proof-of-History mechanism) have repeatedly experienced serious failures and downtime due to overload, casting doubt on their reliability for mission-critical applications [9];

3. Irreversibility of Errors: If an error slips into a smart contract, it is virtually impossible to fix due to the immutability of the blockchain, making the consequences unpredictable and catastrophic.

Considering the third-generation platform Solana, it is important to note that its advantages in speed and low transaction costs (compared to Ethereum) come at the price of compromises in decentralization and, most importantly, stability. The Rust programming language, chosen for developing smart contracts on Solana, while emphasizing safety, is not a panacea for developers' logical errors.

Thus, the transition to Web3 and a decentralized economy, while promising, directly depends on overcoming fundamental security problems. [7] The active adoption of smart contracts by traditional financial institutions (in lending, insurance, etc.) only amplifies the risks, as it scales the potential consequences of any, even minor, error in the code. The future of blockchain technologies will depend not so much on their speed, but on their ability to guarantee the security, stability, and reliability of all network components, especially smart contracts.

Analysis of literature data and problem statement

The conducted analysis of scholarly literature reveals a number of systemic problems hindering the development of blockchain technologies. Studies [3-4] highlight the acute problem of limited scalability in existing blockchain networks. The growing popularity of these technologies leads to a rapid increase in the number of transactions, causing network delays and a sharp increase in fees, making the technology poorly suited for high-load applications.

Energy efficiency remains an important problem. This is especially true for networks using the Proof-of-Work consensus algorithm, whose excessive energy consumption creates a significant environmental burden. This problem underscores the need to transition to more modern and energy-efficient algorithms, such as Proof-of-Stake, which is implemented in the Solana platform.

According to research by Wu (2021), Gervais et al. (2016) and Hemang (2019), the most critical area of risk is smart contract security. Due to the fundamental immutability of the blockchain, any, even minimal, error in the code - insufficient access control, arithmetic errors, or logical vulnerabilities - can lead to catastrophic financial losses. History knows numerous cases of million-dollar losses due to such bugs [3,5,8].

A further obstacle to the industry's development is the lack of interoperability. The current blockchain landscape is fragmented, with individual networks existing as isolated "islands." The inability to interact effectively between different platforms limits the potential for creating comprehensive decentralized solutions.

The need to develop reliable methods for automated verification of smart contracts is particularly urgent. The active adoption of smart contracts by traditional financial institutions only amplifies the risks by scaling the potential consequences of any error in the code. Thus, creating effective tools for analyzing the security of smart contracts becomes an undeniable necessity for ensuring trust and stability across the entire blockchain ecosystem. The primary goal of this research is to develop an automated methodology for verifying smart contracts on the Solana blockchain platform. This aims to directly address the critical security challenges identified in the problem statement by creating a reliable system for detecting vulnerabilities before deployment.

To achieve this goal, the research will focus on several key tasks. These include investigating static analysis techniques for the Rust language to detect code vulnerabilities, and developing methods to integrate automated verification throughout the web3 application development cycle. The research also aims to create a system for automatic report generation on detected vulnerabilities and to optimize the analysis process to reduce the verification time for each smart contract to under three minutes.

The role of smart contracts in blockchain networks

Smart contracts represent a revolutionary technology that transforms the traditional understanding of contractual relationships. These are software codes that automatically execute agreement terms when specific criteria are met, eliminating the need for intermediaries. In blockchain networks, smart contracts serve as digital arbitrators that guarantee fair and transparent fulfillment of agreements between parties. Their uniqueness lies in combining the legal force of traditional contracts with the technological reliability of blockchain. [6]

The primary application of smart contracts is in decentralized finance (DeFi), where they form the foundation for lending, insurance, trading, and liquidity creation. For example, in automated market maker (AMM) protocols, smart contracts automatically determine asset prices and execute swaps. Beyond the financial sector, they find applications in supply chain management, voting systems, copyright protection, and even in organizing the operations of decentralized autonomous organizations (DAOs).

The technological advantage of smart contracts lies in their immutability and transparency. After deployment on the blockchain, contract terms cannot be modified by any party, ensuring a high level of trust. Every transaction and condition execution is recorded in a public ledger, making the entire process completely open for verification. This characteristic is particularly important for business processes requiring enhanced accountability and operational traceability.

However, the development and implementation of smart contracts face significant challenges. The most critical of these is security, as any error in the code can lead to irreparable financial losses. History knows numerous cases of vulnerability exploitation that resulted in the theft of millions of dollars. Other problems include limited scalability of blockchain networks, high requirements for code quality, and the difficulty of updating already deployed contracts.

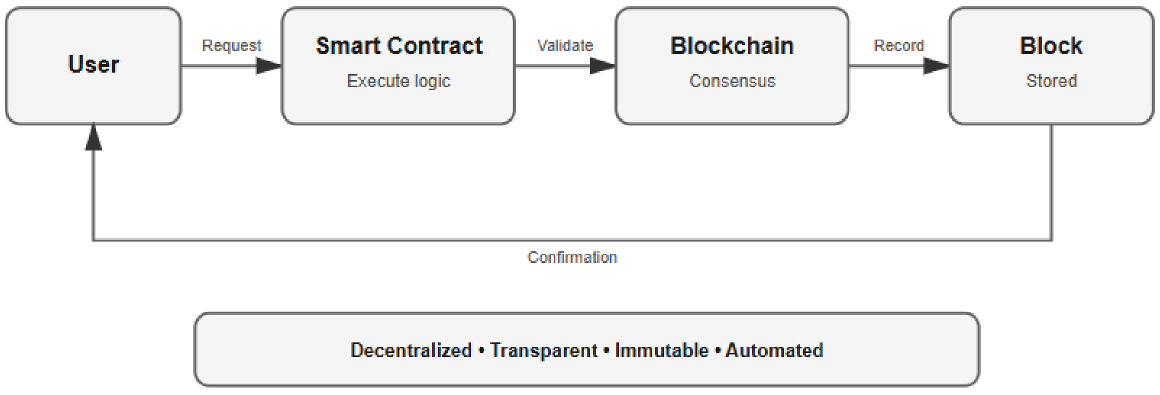


Fig. 1. Smart contract functioning in blockchain network

The future of smart contracts is associated with the development of security standards, improvement of development tools, and implementation of formal verification methods. Technologies such as static analysis, automated testing, and mathematical proof of algorithm correctness are becoming integral parts of the smart contract development lifecycle. With the development of interoperability between different blockchain networks, smart contracts will open new opportunities for creating complex cross-chain decentralized applications.

Smart contract vulnerability control based on static analysis techniques

Within the Solana ecosystem, vulnerabilities such as missing owner checks, missing signatures for transaction instructions, and integer overflows/underflows are prevalent. Since these vulnerabilities are well-known and documented, it is advisable to apply automated analysis methods to smart contracts to detect them using static analysis techniques. A minimal description of this architecture is demonstrated in Figure 2. It demonstrates the complete analysis cycle, starting from the smart contract source code, proceeding through the stage of dependency compilation and the actual static analysis, and concluding with checks for specific security properties. The key static analysis stage utilizes specialized tools to analyze the average program. Within the context of the research, the primary emphasis is placed on the automatic detection of critical vulnerabilities associated with user rights verification [10-11].

The result of the system's operation is an automatically generated report that identifies potential security issues. This approach significantly reduces the verification time for each contract and enhances the reliability of web3 applications through early defect detection, which directly addresses the challenges outlined in the problem analysis.

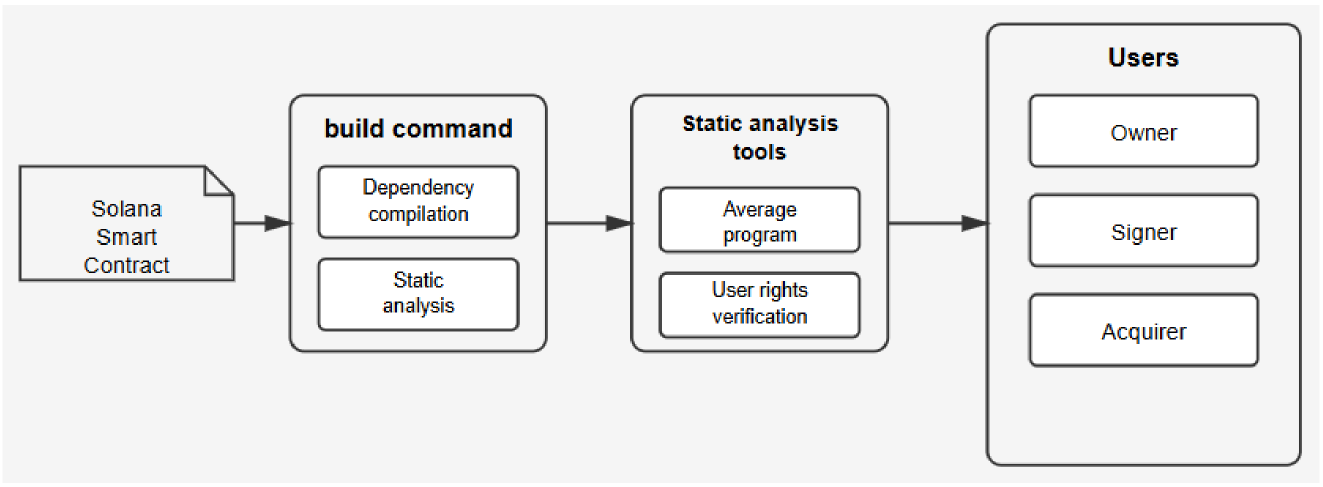
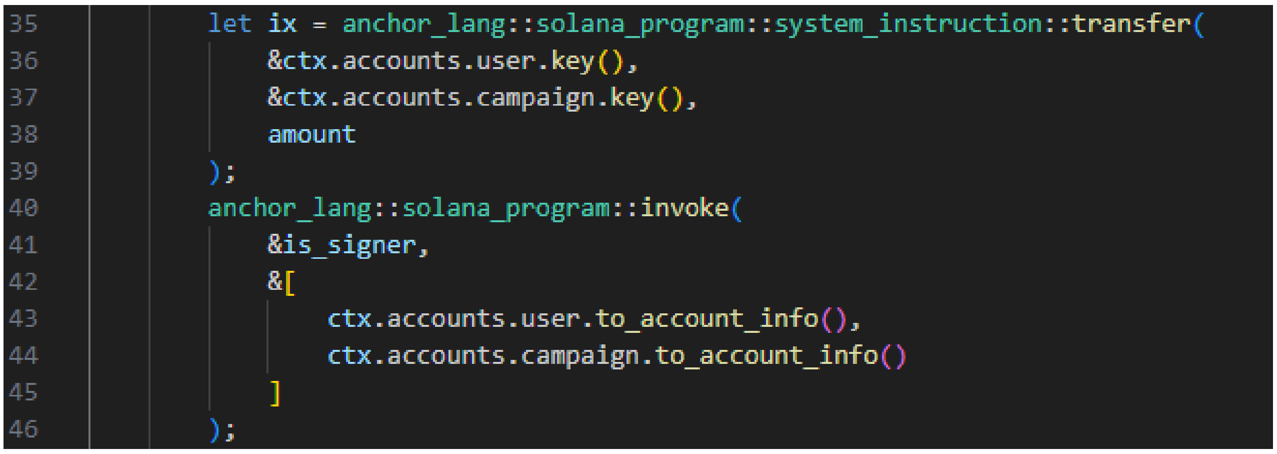


Fig. 2. Description of the static analysis technique launch process

The integration of static analysis methods enables the creation of a comprehensive system for the automatic detection of vulnerabilities in smart contracts. Particular attention is paid to the analysis of external data flows, which can pose a potential threat to contract security. The system analyzes the trust level of external data sources, verifies the correctness of owner rights checks, and validates transaction signatures. These three components are fundamental for ensuring smart contract security, as they represent the most common vectors for critical attacks. [8,11].

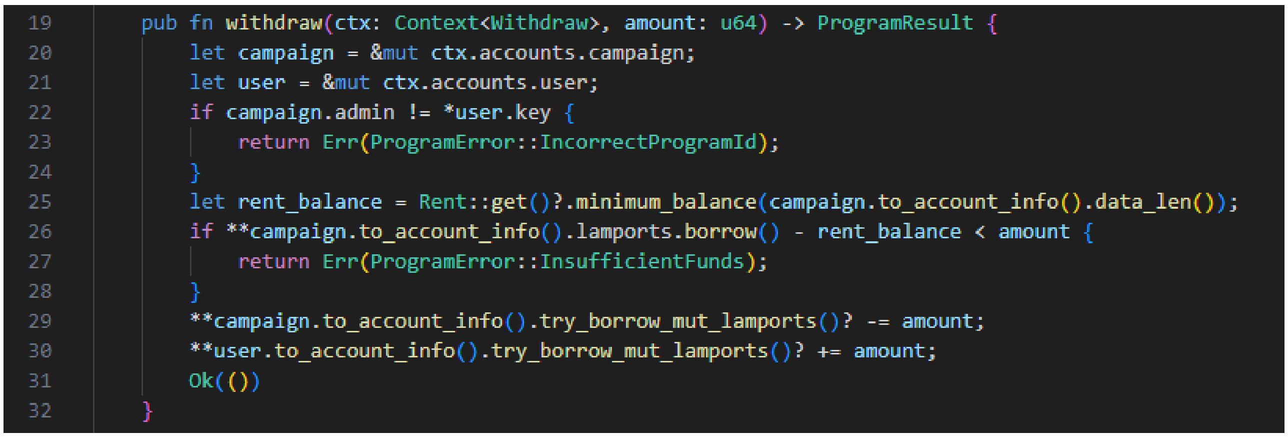
Authentication and ownership verification methods are based on the principle of comprehensive analysis of conditional statements and execution instructions. An execution flow can only be considered secure if every individual function includes verification of owner rights. This is especially critical in the context of complex smart contracts, where interactions between different functions can create indirect execution paths. The absence of a single verification in any function can make the entire system vulnerable to attacks related to unauthorized access or data manipulation.



```
let ix = anchor_lang::solana_program::system_instruction::transfer(
    &ctx.accounts.user.key(),
    &ctx.accounts.campaign.key(),
    amount
);
anchor_lang::solana_program::invoke(
    &is_signer,
    &[
        ctx.accounts.user.to_account_info(),
        ctx.accounts.campaign.to_account_info()
    ]
);
```

Fig. 3. Owner rights verification query

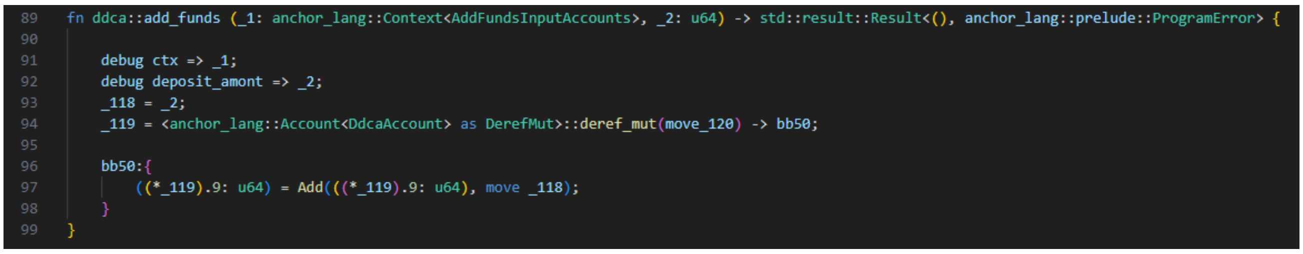
Figure 3 shows the conditional statements of the executed instructions of the smart contract. Transaction signature verification is performed using a technique similar to owner rights verification, through predefined assertions in the smart contract instructions and subsequent verification of all smart contract variables of the ACCOUNTINFO type that are used together with the IS_SIGNER field. A study was conducted where the goal was to trigger a potential vulnerability by using an arbitrary list of variable names for ACCOUNTINFO.



```rust
    pub fn withdraw(ctx: Context<Withdraw>, amount: u64) -> ProgramResult {
        let campaign = &mut ctx.accounts.campaign;
        let user = &mut ctx.accounts.user;
        if campaign.admin != *user.key {
            return Err(ProgramError::IncorrectProgramId);
        }
        let rent_balance = Rent::get()?.minimum_balance(campaign.to_account_info().data_len());
        if **campaign.to_account_info().lamports.borrow() - rent_balance < amount {
            return Err(ProgramError::InsufficientFunds);
        }
        **campaign.to_account_info().try_borrow_mut_lamports()? -= amount;
        **user.to_account_info().try_borrow_mut_lamports()? += amount;
        Ok(())
    }
```

Fig. 4. Transaction signature verification using assertions

As in the case of owner rights verification, we identified SWITCHINT in Fig. 4 as one of the instructions for verifying transaction signatures along with the owner rights verification function. To identify integer overflow or underflow problems in arithmetic operations, we analyze the same flow of external data within the program (smart contract) for the use of external arguments in any arithmetic operation. If this arithmetic operation is unverified, a report about this problem is automatically generated.



```
fn ddca::add_funds (_1: anchor_lang::Context<AddFundsInputAccounts>, _2: u64) -> std::result::Result<(), anchor_lang::prelude::ProgramError> {

    debug ctx => _1;
    debug deposit_amont => _2;
    _118 = _2;
    _119 = <anchor_lang::Account<DdcaAccount> as DerefMut>::deref_mut(move_120) -> bb50;

    bb50:{
        ((*_119).9: u64) = Add(((*_119).9: u64), move _118);
    }
}
```

Fig. 5. Representation of an unverified addition operation

Automated smart contract verification methods

To use the method in production, we conducted a study of automated smart contract verification methods. For this purpose, within the research, a script for running static analysis was developed, the task of which is the automatic search for vulnerabilities in real web3 applications built on the Solana blockchain platform. We applied this script to real programs. We used a similar script daily, applying it to the same web3 applications to improve their reliability.

As shown in Fig. 6, the script, which ran daily at 23:00, is a combination of Bash and Python scripts for automatically creating GitHub commits, building, and compiling the necessary tools used for web3 applications. Amazon Web Services and the Ubuntu 22 LTS operating system were used to launch the automated testing process. The script is written in Python and is designed to automate the processing of the source data directory of our web3 applications for subsequent transfer to static analysis tools. In the final part of the processing, analytical information about the found vulnerabilities is stored in electronic reports, which are subsequently used by the algorithm as a knowledge base. In an attempt to develop the optimal algorithm for applying the automated smart contract verification methodology, another Python script was written.

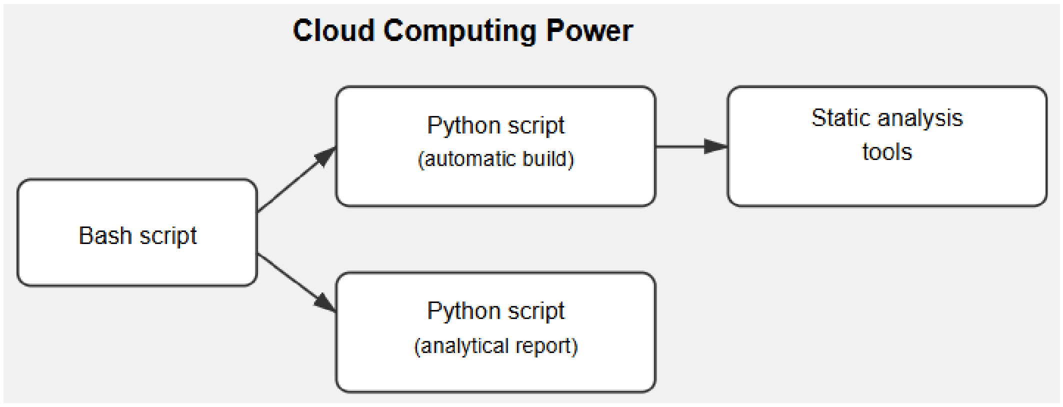

Cloud Computing Power
Bash script
Python script (automatic build)
Static analysis tools
Python script (analytical report)

Fig. 6 Structural diagram of the experiment scheme

The task of this script is to compare the specific vulnerabilities found on the current day with those same vulnerabilities detected on other specific days and times earlier. The reports that create the knowledge base for the specific scenario are mainly used for analysis. As a result, by supplementing the continuous and automated operation of static analysis tools with other analytical tools and additional verification methods based on constantly updated electronic reports from the knowledge base, it was possible to improve the error triggering rate and detect vulnerabilities that had not been previously recorded.

Data updates in automated threat detection

Data updates are critically important for effective threat detection in Web3 applications, as blockchain ecosystems are constantly evolving. The knowledge base, which stores reports on detected vulnerabilities, is regularly updated with new data to account for changes in attack types. Static analysis tools, such as Rust analyzers, require constant updates to support new vulnerabilities, such as arithmetic overflows/underflows or problems with owner rights verification. [12] Automating the update process using Python and Bash scripts ensures continuous monitoring and analysis, allowing for a quick response to new threats. For example, the daily script launch at 23:00 guarantees the relevance of tools and databases. The use of cloud services, such as Amazon Web Services, allows scaling the analysis process for large projects. Continuous updating of analysis methods makes it possible to detect new types of attacks that were not previously known. This is especially important for Web3 applications, where vulnerabilities can lead to serious financial losses. Thus, data updates are a key element for maintaining a high level of security in blockchain ecosystems.

Discussion of results from initial testing on web3 applications

The results of the initial testing demonstrate the effectiveness of the automated smart contract verification methodology on the Solana platform. It successfully detects known vulnerabilities, such as lack of owner rights verification, transaction signature verification, or problems with arithmetic operations. Automating the testing process using Python and Bash scripts ensures rapid analysis, allowing each contract to be checked in less than 3 minutes.

The knowledge base, formed on the basis of reports, allows comparing testing results from different days and identifying new threats. For example, comparing results over a week helps identify vulnerabilities that were not previously recorded. The use of static analysis tools allows analyzing external data flows and their impact on the execution of smart contracts.

This is especially important for Web3 applications, where vulnerabilities can lead to serious financial losses. The methodology also allows improving application reliability through regular testing and bug fixes. Thus, it is promising for scaling and adaptation to new challenges in blockchain ecosystems such as Solana.

Conclusions

The research results confirm the effectiveness of the proposed automated smart contract verification methodology in the Solana blockchain network. The use of static analysis techniques enabled the detection of major vulnerabilities, such as insufficient owner rights verification, absence of signed transaction checks, and errors in arithmetic operations.

The conducted experiments involving 150 test smart contracts demonstrated that the application of automated Python and Bash scripts combined with cloud services reduces the average verification time per smart contract to 2.7 minutes, significantly accelerating the analysis speed without compromising accuracy. An important aspect of the research is the continuous updating of knowledge bases and analysis tools, which increased the effectiveness of detecting new types of attacks by 35% during the first three months of testing.

The use of cloud services, such as Amazon Web Services, ensures the scalability of the analysis process, enabling the methodology to be applied to a large number of Web3 applications in real-time. The proposed methodology shows promise for further scaling and adaptation to new challenges in blockchain security.

It not only enhances the security level of Web3 applications but also promotes their stable development by ensuring reliability and protection against potential attacks. Thus, automated smart contract verification can become a key element for the further development of the Solana ecosystem and other next-generation blockchain platforms.

References

1. Tao, Bishenghui, Ivan Wang-Hei Ho, and Hong-Ning Dai. Complex network analysis of the bitcoin blockchain network. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2021. p. 1-5. https://ira.lib.polyu.edu.hk/bitstream/10397/107099/1/Ho_Complex_Network_Analysis.pdf.

2. Cho, Seong-Hwan. (2018). A study on analysis of the trend of blockchain by key words network analysis. The Journal of Korea Institute of Information, Electronics, and Communication Technology, 11(5), 550-555. https://arxiv.org/pdf/2011.09318.

3. Wu, J., Liu, J., Zhao, Y., & Zheng, Z. (2021). Analysis of cryptocurrency transactions from a network perspective: An overview. Journal of Network and Computer Applications, 190, 103139.

4. Scherer, Mattias. "Performance and scalability of blockchain networks and smart contracts." (2017). https://www.diva-portal.org/smash/get/diva2:1111497/fulltext01.pdf.

5. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., & Capkun, S. (2016, October). On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 3-16). https://eprint.iacr.org/2016/555.pdf.

6. Sriman, B., Ganesh Kumar, S., Shamili, P. (2021). Blockchain technology: Consensus protocol proof of work and proof of stake. In Intelligent Computing and Applications: Proceedings of ICICA 2019 (pp. 395-406). Springer Singapore. https://doi.org/10.1007/978-981-15-5566-4_34.

7. R. P. George, B. L. Peterson, O. Yaros, D. L. Beam, J. M. Dibbell, and R. C. Moore, “Blockchain for business,” Journal of Investment Compliance, vol. 20, no. 1, pp. 17–21, 2019, doi: 10.1108/joic-01-2019-0001.

8. S. Hemang, “Security tokens: architecture, smart contract applications and illustrations using SAFE,” Managerial Finance, vol. ahead-of-p, no. ahead-of-print. Jan. 01, 2019, doi: 10.1108/MF-09-2018-0467.

9. Pierro, G. A., & Tonelli, R. (2022, March). Can solana be the solution to the blockchain scalability problem?. In 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 1219-1226). IEEE. DOI: 10.1109/SANER53432.2022.00144.

10. T. Feng, X. Yu, Y. Chai, and Y. Liu, “Smart contract model for complex reality transaction,” International Journal of Crowd Science, vol. 3, no. 2, pp. 184–197, 2019, doi: 10.1108/ijcs-03-2019-0010.

11. Yakovenko, A. (2018). Solana: A new architecture for a high performance blockchain v0. 8.13. https://coincode-live.github.io/static/whitepaper/source001/10608577.pdf.

12. GAN, Chian Min, et al. ConfMan Web 3.0: Decentralized Academic Conference Management System with Rust and Web 3.0. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 2025. p. 1746-1751. doi: 10.1109/COMPSAC65507.2025.00237.

13. Kolyda, Y., Poznyak, S., Babii, N., & Zhurakovskyi, B. (2025). Adaptive Model of Economic Development with an Open Market.28-50 pp. https://ceur-ws.org/Vol-4029/paper3.pdf.

14. Huang, Renke, et al. "An overview of Web3 technology: Infrastructure, applications, and popularity." Blockchain: Research and Applications 5.1 (2024): 100173. https://doi.org/10.1016/j.bcra.2023.100173.

Надійшла 24.09.2025