Security issues in Dutch Auction

Chen Yiting
U2022489K
SCSE
Nanyang Technological University
Singapore
CHEN1421@e.ntu.edu.sg

Zhao Yu
Student's Matriculation Number
SCSE
Nanyang Technological University
Singapore
Email address of Student

Zou Zeren
U2022422H
SCSE
Nanyang Technological University
Singapore
ZZOU002@e.ntu.edu.sg

***Abstract*—This study delves into key security issues in the solidity dutch auction system submitted for CE/CZ4153 Blockchain Technology course, with focused discussions on code quality, front-running attacks, and possible timestamp manipulation. Firstly, the paper discuss some vulnerabilities found in the code implementation by utilizing Slither for a thorough examination of smart contracts. Secondly, the paper underscores the absence of privacy provisions in auction contracts, which may lead to the exposure of private data and bidder information. Lastly, it looks into potential weak spots with block timestamps in the auction implementation, highlighting how such manipulation could impact the fairness of auctions. The intent of this research is to improve the auction system developed by addressing these security concerns and suggesting solutions. The findings hold significance for creating more secure, efficient, and dependable dutch auction smart contracts.**

***Index Terms*—blockchain, Ethereum, smart contracts, tokens, dutch auction, privacy.**

I. Introduction

A. Blockchian Technologies

Blockchain technologies, essential for secure distributed computations, are gaining traction due to their wide range of applications [1]. Ethereum, a popular blockchain platform, supports not just cryptocurrency but also other applications like games and financial services through its decentralized Ethereum Virtual Machine (EVM) and smart contracts [2]. Three fundamental concerns are critical in the field of blockchain: security, scalability, and privacy.

- **Security** properties of blockchain stems from developments in both cryptography and chain architecture. Several intrinsic security features, including consistency, resistance to tampering, resistance to Distributed Denial-of-Service (DDoS) attacks, pseudonymity, and resistance to double-spending attacks, are ensured by the design of the blockchain [3]. However, extra high-level security features are needed with the advancement of smart contracts, such as overflow protection and code injection resistance.
- **Privacy** concerns, particularly in applications that manage sensitive data, stem from blockchain's transparent nature. To prevent user data from being misused or exposed, it can be difficult to strike a balance between transparency and confidentiality.
- **Scalability** refers to the ability of a blockchain network to effectively manage high transaction volumes. With poor throughput, significant transaction delay, and massive consumption of energy, both Bitcoin and Ethereum are experiencing scalability problems [4].

Term Paper submitted for CE/CZ4153 Blockchain Technology, NTU.

B. Dutch Auction

In an auction, a seller offers products or services for sale, and potential buyers submit their bids based on the price they are willing to pay. Numerous types of auctions have been developed over time. In a Dutch auction, the seller sets the starting price and gradually reduces it until a bidder agrees to the going rate [5].

C. Project Focus and Development

Our development project's main goal was to create a dutch auction system based on blockchain technology, taking advantage of the immutability and decentralisation of blockchain technology to guarantee an open and equitable bidding process. In order to support different kinds of auctions and preserve the integrity and dependability of the bidding process, we created a number of smart contracts.

D. Paper Focus and Contribution

This paper focuses on blockchain **security** issues in our auction system, discussing front-running attack issues in smart contracts, potential timestamp manipulation, and overall code quality and safety. Despite the importance of privacy and scalability, enhancing the security framework is essential. The study addresses these issues, provides solutions, and contributes to the stability and reliability of our dutch auction system, setting roads for future improvements.

II. Motivation and Literature Survey

A. Motivation

Security is the paramount concern for our blockchain-based, decentralized auction system. This emphasis is due to the financial implications of the system, the immutable nature of smart contracts, and lessons learned from compromised blockchain projects.

Firstly, the system's financial transactions, managed by smart contracts, require high security standards to prevent loss

or unfair advantages. Secondly, in a decentralized environment, trust lies in the code, making its security vital for user confidence and system integrity [6]. Blockchain's immutability and transparency further magnify security concerns. Any flaws become permanent vulnerabilities if not addressed beforehand, necessitating rigorous security measures during development. Lastly, past blockchain projects reveal that security is often overlooked, leading to breaches and system failures [7].

Therefore, despite the importance of privacy and scalability, security is our top priority. This aligns with the needs of our users, stakeholders, and best practices within the blockchain community.

B. Literature Survey

Various blockchain solutions have been proposed in recent studies to improve the effectiveness and security of e-auction systems. These studies investigate various facets of blockchain technology, ranging from enhancing the auction procedure to guaranteeing transaction security.

Blockchain Auction for Secure Communications: Khan et al. [8] propose a blockchain-based distributive auction system designed for relay-assisted secure communications. Their system eliminates the need for a central authority by decentralising the auction process through the use of blockchain. By automating transaction rules, smart contract implementation improves the security and stability of the system. Additionally, the study discusses possible weaknesses in distributed systems and suggests defences against different kinds of malicious behaviours.

Blockchian Auction in UAE:A blockchain-based e-auction system designed specifically for the United Arab Emirates (UAE) market is presented by Qusa et al [9]. Their research focuses on leveraging blockchain technology to establish an open and impenetrable bidding environment. The technology uses smart contracts to automate auction procedures, eliminating the need for middlemen. The authors talk about how this system can deal with issues that frequently arise in online auctions, like bid rigging and privacy concerns.

Enhanced Tree-Structured E-Auction: An enhanced blockchain tree structure-based e-auction system is proposed by Sarfaraz et al. [10] to make the processes of evaluating bids and choosing winners more efficient. By reducing the complexity of bid management, the tree-structured approach facilitates the safe and transparent handling of a large number of bids. The study demonstrates how blockchain technology can simplify e-auction procedures while maintaining security and fairness.

Safe Sealed-bid Bid Procedure: Zhang and colleagues [11] created a blockchain-based secure sealed-bid auction system. Their method makes use of blockchain's transparency and immutability to guarantee the validity of auction bids. By enforcing stringent guidelines for bid submission and opening, the system uses smart contracts to prevent bid leakage and manipulation. The authors do point out the difficulties in ensuring that bidders correctly interact with the contract functions and the complexity of implementing smart contracts in such a system.

III. Observations and Analysis

A. Reentrancy Issue in Auction._finalization Function of Dutch Auction Application

The `Auction._finalization` function in the Dutch Auction contract appears to possess a reentrancy vulnerability. This is evidenced by external calls which may be exploited to re-enter the contract before its execution completion. Reentrancy is feasible as the `_processPurchase` function initiates an external call to the token contract. This grants control to potentially untrusted contracts, which might then re-invoke the Auction contract prior to the conclusion of the original call. If leveraged, this vulnerability could enable an attacker to deplete funds from the contract or to disrupt the auction's logical processes, such as impeding the finalization process and equitable distribution of auctioned items. A similar reentrancy attack was witnessed with KyberSwap, a decentralized exchange, where an exploit of a reentrancy flaw in a mint function led to a loss of approximately 47 million dollars. This incident accentuates the criticality of securing smart contract designs and the severe repercussions that vulnerabilities can impose on DeFi platforms. [12]

B. Dangerous Strict Equalities in DutchAuction._preValidateFinalization()

The use of strict equality checks in `DutchAuction._preValidateFinalization()` can lead to logic misinterpretation. The code employs `==` to compare `remainingSupply()` with 0. This could be problematic if `remainingSupply()` does not exactly equal zero due to rounding or minor calculation errors. Such issues might arise from the discrete nature of blockchain states. Improper failure of this check could prevent the auction from finalizing correctly, leading to funds being locked in the contract. A similar DoS vulnerability, known as 'Gridlock,' exploited strict equality in Edgeware's smart contracts. This case highlights the dangers of strict equalities and the importance of defensive programming in Solidity. Slither's dangerous-strict-equality detector is designed to catch such vulnerabilities [13].

C. Low-Level Calls in Auction Contract

The Auction contract implements low-level calls such as `.call()`, `.delegatecall()`, and `.staticcall()` for interactions with external contracts. These calls, while providing control over gas and exception handling, do not automatically throw exceptions on failure, potentially leading to security vulnerabilities like reentrancy attacks. An unchecked return value from a low-level call can result in loss of funds or an invalid contract state. Solidity's `extcodesize` is recommended to verify contract existence before such calls [14].

D. Timestamp Dependence in TimedAuction Contract Functions

`TimedAuction` contract functions such as `isOpen`, `hasClosed`, and the constructor, depend on `block.timestamp` for timing actions. Timestamp manipulation by miners can impact the auction outcome, leading to unfair practices or time-based attacks. Notable instances include Feathercoin's exploitation through timestamp manipulation, affecting its market value, and the GovernMental Ponzi scheme, manipulated to benefit a miner, resulting in the accumulation of nearly 1100 ETH. These incidents underscore the risks associated with timestamp dependence in smart contract design [15].

IV. Proposed Solutions

In this section, propose potential solutions to address the issues that you found in your analysis earlier. These solutions may be inspired from the lectures, invited talks, related works, or any other instance of similar development projects.

A. Solution to Reentrancy Issue in Auction._finalization Function

1) **Implementation of Reentrancy Guards:** Introducing a reentrancy guard, such as a state variable that tracks whether the function is already being executed, can prevent recursive calls. This guard would be checked and set at the beginning of the `_finalization` function.
2) **Use of Checks-Effects-Interactions Pattern:** Restructuring the function to first make all state changes (checks and effects) and then perform external calls (interactions) can mitigate reentrancy risks. This would involve modifying the `_processPurchase` function to ensure that all critical state changes occur before any external calls are made.

B. Solution to Dangerous Strict Equalities in DutchAuction._preValidateFinalization()

1) **Implementation of Range-Based Checks:** Replace strict equality checks with range-based checks. For instance, instead of `remainingSupply() == 0`, use a condition that checks if `remainingSupply()` is within an acceptable margin of error. This approach accounts for minor discrepancies due to rounding or calculation errors.
2) **Comprehensive Testing and Validation:** Implement thorough testing procedures to simulate edge cases and ensure that the modified checks behave as expected under various scenarios, including those that would have previously caused strict equality checks to fail.

C. Solution to Low-Level Calls in Auction Contract

1) **Enhanced Return Value Checking:** Ensure that all low-level calls such as `.call()`, `.delegatecall()`, and `.staticcall()` include robust checks for their return values. Implement a fail-safe mechanism that reverts the transaction if the call returns false, thereby preventing the continuation of execution under failure conditions.
2) **Use of High-Level Abstractions:** Where possible, replace low-level calls with high-level contract interactions that inherently handle exceptions. This reduces the direct use of low-level calls and leverages the safety features of Solidity's high-level constructs.
3) **Contract Existence Verification:** Apply Solidity's `extcodesize` to check for the existence of the contract at the address being called. This ensures that calls are made to valid contracts and mitigates the risk of interacting with non-existent contracts.

D. Solution to Timestamp Dependence in TimedAuction Contract Functions

1) **Block Number Utilization:** Replace `block.timestamp` with block numbers for timing actions. Since block numbers are less prone to manipulation, they can provide a more reliable measure for contract events timing.
2) **Time-Window Mechanisms:** Implement a time-window approach where actions are not dependent on a single block timestamp but on a range of blocks, reducing the potential impact of timestamp manipulation.
3) **External Time Oracle:** Utilize a trusted external time oracle to provide a more secure and reliable time reference for the contract. This approach helps to reduce dependency on the blockchain's internal timekeeping mechanisms.

V. Conclusion

In this paper, we focused on the issue of security within a blockchain-based Dutch auction system, a critical aspect given the financial stakes and the immutable nature of blockchain technology. We identified and analyzed several vulnerabilities such as reentrancy risks, issues with strict equalities, low-level call vulnerabilities, and dependencies on block timestamps. These vulnerabilities were not only highlighted but also addressed through comprehensive solutions including the implementation of reentrancy guards, range-based checks, enhanced validation of low-level calls, and the use of block numbers and external oracles for time-dependent functions.

The overall contribution of this paper lies in its detailed analysis of specific security challenges within a blockchain auction system and the provision of targeted, practical solutions. These solutions are informed by best practices in smart contract security and lessons learned from previous incidents in the blockchain space. Our research provides valuable insights for improving the security of the Dutch auction system developed for the CE/CZ4153 Blockchain Technology course at Nanyang Technological University. Additionally, the methodologies and solutions proposed in this paper can serve as a reference for the broader blockchain community, aiding in the development of more secure, reliable, and efficient blockchain-based systems.

REFERENCES

[1] M. Javaid, A. Haleem, R. Pratap Singh, S. Khan, and R. Suman, "Blockchain technology applications for Industry 4.0: A literature-based review," *Blockchain: Research and Applications*, vol. 2, p. 100027, Dec. 2021.

[2] P. P. Pothavarjula and B. Sirisha, "An Investigation of Decentralized Ledger Applications Using Ethereum in a Blockchain Network," in *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 524–529, Mar. 2022.

[3] R. Zhang, R. Xue, and L. Liu, "Security and Privacy on Blockchain," *ACM Computing Surveys*, vol. 52, pp. 1–34, May 2020.

[4] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to Scalability of Blockchain: A Survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[5] M. Bennett, R. Mullard, M. T. P. Adam, M. Steyvers, S. Brown, and A. Eidels, "Going, going, gone: Competitive decision-making in Dutch auctions," *Cognitive Research: Principles and Implications*, vol. 5, p. 62, Nov. 2020.

[6] M. Zachariadis, G. Hileman, and S. V. Scott, "Governance and control in distributed ledgers: Understanding the challenges facing blockchain technology in financial services," *Information and Organization*, vol. 29, pp. 105–117, June 2019.

[7] S. Hwang and S. Ryu, "Gap between theory and practice: An empirical study of security patches in solidity," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, (Seoul South Korea), pp. 542–553, ACM, June 2020.

[8] A. S. Khan, Y. Rahulamathavan, B. Basutli, G. Zheng, B. Assadhan, and S. Lambotharan, "Blockchain-Based Distributive Auction for Relay-Assisted Secure Communications," *IEEE Access*, vol. 7, pp. 95555–95568, 2019.

[9] H. Qusa, J. Tarazi, and V. Akre, "Secure E-Auction System Using Blockchain: UAE Case Study," in *2020 Advances in Science and Engineering Technology International Conferences (ASET)*, (Dubai, United Arab Emirates), pp. 1–5, IEEE, Feb. 2020.

[10] A. Sarfaraz, R. K. Chakrabortty, and D. L. Essam, "A tree structure-based improved blockchain framework for a secure online bidding system," *Computers & Security*, vol. 102, p. 102147, Mar. 2021.

[11] M. Zhang, M. Yang, and G. Shen, "SSBAS-FA: A secure sealed-bid e-auction scheme with fair arbitration based on time-released blockchain," *Journal of Systems Architecture*, vol. 129, p. 102619, Aug. 2022.

[12] Hacken, "Kyberswap hack explained," 2023. Accessed: 2023-11-29.

[13] Trail of Bits, "Avoiding smart contract "gridlock" with slither," 2019. Accessed: 2023-11-29.

[14] K. Zipfel, "Unsafe low-level call - smart contract vulnerabilities," 2023. Accessed: 2023-11-29.

[15] Neptune Mutual, "Understanding block timestamp manipulation," 2023. Accessed: 2023-11-29.