Security issues in “Something Funky Is Nice” Decentralized Exchange

Siek Ming Jun
U1820369F
School of Computer Science and Enginnering
Nanyang Technological University
Singapore
MSIEK001@e.ntu.edu.sg

Samuel Tan Yong An
U1820796D
School of Computer Science and Enginnering
Nanyang Technological University
Singapore
STAN183@e.ntu.edu.sg

***Abstract*—In a Blockchain, it faces Security, Privacy and Scalability issues. However, in terms of a Decentralized Exchange, Security takes the biggest priority to solve due to its nature. A Decentralized Exchange is a place where users come and conduct trades to swap their token for another token. These tokens have monetary value and thus, the user must trust that the Decentralized Exchange will ensure the safety of their funds. If the Security of the Decentralized Exchange is not enforced, there could be vulnerabilities in the Decentralized Exchange which could be exploited by attackers. This would be devastating to both the users and the exchange as both the user’s fund and their trust in the exchange is lost. Therefore, we must identify the issues related to Security and look for possible solutions that would solve these issues.**

Keywords—Blockchain, Ethereum, Smart Contracts, Tokens, Token Exchange, Decentralized Application, Decentralized Exchange, Blockchain Security

I. Introduction

Bitcoin is the world’s first decentralized public ledger system that was developed by Satoshi Nakamoto in 2009 [1]. It utilizes the idea of a Blockchain, which is a distributed data processing protocol that is able to retain a public distributed ledger in a Peer-to-peer (P2P) network [2]. This ledger is made up of blocks that are used to store transaction data. Multiple blocks are linked together to create the Blockchain. In the Blockchain network, multiple nodes are required to store and maintain the entire copy of the ledger without a central authority. In order to prevent manipulation of ledger, each block will contain the hash value of the previous block and altering the previous blocks will yield a different hash value in the subsequent block. The change in hash value will be flagged out by the network and the nodes will discard this chain as there were signs of manipulation.

A consensus protocol is used to ensure data integrity among the nodes of the blockchain P2P network. In Bitcoin and Ethereum, both utilizes the Proof of Work (PoW) protocol. In Bitcoin’s PoW, each node in the network compete against each other to solve a computationally intensive problem. The problem is to find a nonce value that is able to produce a hash that meets a certain requirement. This requirement has an adjustable difficulty level based on the current speed of PoW. Once a node found the right nonce value, a block is generated, and it will be broadcasted to the entire P2P network. The nodes in the network will check to see if the block is valid before appending it to the end of their chain.

There are other utilities of Blockchain other than a public ledger. Ethereum was created to be a blockchain with a built-in Turing-complete programming language. This language is used to create contracts that can encode arbitrary functions, allowing users to create Decentralized Applications (DApps) [3]. In one of the guest lecture with Vitalik Buterin, he explained that Ethereum is a general purpose Blockchain that is able to handle algorithms expressed in a general-purpose programming language. Instead of establishing a different Blockchain for each use case or applications, a variety of use cases can be accomplished by technologies known as smart contracts. This enables developers to create a range of applications from basic wallet to financial systems like Decentralized Finance. Thus, with smart contracts, we are able to create DApps on the Ethereum Blockchain.

However, there are 3 key issues in Blockchain: Security, Privacy and Scalability.

In Security, Blockchains have faced 51% attacks, selfish mining, front-running attacks and eclipse attack [4]. The main reason for these attacks is due to monetary reasons. Cryptocurrencies have monetary value and attackers are always trying to steal those assets away from the users. This is especially true in the case of Ethereum, where it currently has multiple tokens on the network and all of them have monetary value. With the ability to create DApps, people are now able to utilize these applications but there is always a risk of these DApps being compromised by attackers. If a DApp is compromised, it would mean the loss of assets for the user.

In Privacy, it was found that Bitcoin is not completely anonymous but pseudonymous [5]. As Blockchains like Bitcoin and Ethereum are public, the data on the Blockchain can be viewed by anyone. This is an issue for companies or individual who are not keen on having their sensitive information and business intelligence publicly available on the Blockchain.

In Scalability, Bitcoin processes 7 transactions per second (TSP) and Ethereum processes 10 TSP. Compared to Visa which processes 4,000 TSP, both Bitcoin and Ethereum are very far behind in the transaction race [6]. The reason why the current Blockchain implementations have very small TPS is due to Scalability issues. If a large amount of people suddenly starts to use the network and it goes above a certain threshold, the network will be congested, and it would cause the speed of transactions to decrease.

For our development project, we were tasked to create a DApp on Ethereum. The DApp we chose to create is a

Decentralized Exchange (DEX) called "Something Funky Is Nice DEX". We used React for the front-end and Solidity for the smart contract.

The DEX we developed uses Wrapped Ether (WETH) and it supports WETH/SCSE, WETH/MAE and WETH/EEE trading pairs. In order to trade, users must first swap their Ethereum into WETH and this can be done on the DEX.

Users are able to make Limit and Market Buy/Sell order on each of the trading pairs. Users are also allowed to cancel their orders if it has not been executed. The DEX allows the users to view all the current Buy and Sell order book as well as the user's own Buy and Sell orders.

The DEX follows the principles of 'not your keys, not your crypto'. Nobody holds onto your assets other than yourself and thus, there is no need to deposit the tokens into the DEX. Once there is a buyer/seller for your order, the DEX facilitates the trade on behalf of the user and execute a token swap between the token pair directly from the user's wallet.

For this term paper, we will mainly be discussing the Security issues that are commonly found in DEX as well as the possible solutions to solve those issues.

II. Motivation and Literature Survey

A. Motivation

For a DEX, Security is one of the most important concerns we have during development. This is because for a DEX, it involves the user's money and it is the responsibility of the DEX to ensure that the user's funds are not stolen. Failure to protect the user's fund will result in the loss of user's trust in the DEX, which would cripple the DEX and our business.

The first Security issue we were concern about is front-running attack on a DEX trade. Due to the transparent nature of Blockchain, the order book can be accessed publicly. This makes it easy for front runners to find attractive trades and enter their trade first by increasing their transaction's gas price. If an attacker is able to execute front-running attacks on a DEX, it could cause other legitimate transactions to fail or even cause the loss of funds for users.

The second Security issue we were concern about is the occurrence of overflow and underflow bugs. As a DEX deals with numbers and user's fund, an overflow or underflow occurring during a calculation of a trade would result in the wrong amount being sent to the users.

The last Security issue we were concern about is the hacking of smart contracts. Smart contract is not only used for interpreting business logic and performing computational operations but is also in possession of valuable digital assets. Smart contracts are also open source, which means anyone can have access to the code and look for vulnerabilities that can be exploited. Based on the latest leaderboard, there are currently \$3.79 Billion USD worth of tokens locked up in DEXs [7]. This makes it a valuable target for hackers trying to gain access to these assets by looking for vulnerabilities in the smart contracts.

B. Literature Survey

In this Literature Survey, we will take a look at various studies conducted in the past regarding Security issues as well as news articles about attacks that took place.

In the first study, the authors discussed about the security issues of DEX [8]. One of the issues is regarding DEX Interface attacks and this is focused towards the users. This can be an attack on the web browser wallet, Metamask, or deceptive phishing on DEX. An attack on Metamask includes phishing of user's wallet information and triggering unintended user actions. Deceptive phishing is an attack where the attackers impersonate a legitimate website and steal the users' information. Both of these attacks are conducted on the user and it would result in the loss of assets for the user.

The other issue is regarding front-running attack. The authors conducted experiments on one of the DEX, Oasis, to test the feasibility of front-running attacks and analyze the profitability of the attack. The authors conducted 2 types of front-running attacks, one is done by the user and the other is done by the miner. For the user front-running attack, they concluded that it would be a lucrative scheme only if the order size is big. For the miner front-running attack, the authors were not able to do much due to limited mining power.

In the second study, the authors conducted experiments on an automated DEX, Quipuswap to see whether it was possible to prevent front-running attacks using a certain condition. From the results. it was found that DEXs were prone to front-running attacks, with or without automatic market-making [9]. If the DEX was to limit the execution of the transaction, it would prevent the front-running attack but at the same time, reduces the likelihood of the successful execution of the transaction.

In the third study, it was found that in 1,311 smart contracts, 12.87% of these contracts were found to have 'Integer Overflow' vulnerabilities which they classify as a high-risk vulnerability [10]. This shows that there is always a possibility for overflow bugs to exist inside the smart contracts.

In 2018, Bancor, a crypto company that provide DEX services was hacked and lost \$23.5 Million worth of cryptocurrency. The attacker exploited a vulnerability in a wallet that was used to upgrade some smart contracts [11].

In April 2020, Bisq, a DEX, was hacked and had more than \$250,000 worth of cryptocurrency stolen. This was due to the attacker exploiting a flaw in the Bisq trade protocol, which allows the attacker to target individual trades in order to steal trading capital [12].

III. Observations and Analysis

A. Front Running Attack in case of DEX

In our DEX, there is a possibility of front-running attacks. This is because the trades made by the users on our DEX can be viewed publicly on the Ethereum network. This would mean that the attacker is able to target our exchange and view the order book to look for attractive trades. If a user were to make a mistake and submitted a wrong trade, the user would have the option to cancel the trade.

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

However, if the attacker was to notice this particular trade, they could execute a front-running attack and send a transaction that has a higher gas price in order for their transaction to be placed higher than the user's cancel transaction. The miners would take the transaction that has the higher gas price and they would reject the user's cancel transaction. In the end, only the user would be the one at the losing end.

In previous studies, the authors were able to execute front-running attacks on live DEXs [8, 9]. As our DEX does not have any protection for this kind of attack, it would mean that there is a possibility that this attack could occur in our DEX.

B. *Overflow & Underflow Bug in case of DEX Contract*

One of the vulnerabilities that an attacker could exploit in the smart contract is the overflow & underflow bug. For this bug, a uint and int variable will wrap around if it goes above its maximum value, or if it goes below its minimum value. This is a huge vulnerabilities for a DEX as it deals with numbers and user's funds.

For example, if the contract uses uint8 and the value of the variable is 255, the maximum value it can hold. Adding even just 1 to it will cause an integer overflow and the value will not be 256 but 0. In terms of a trade, if this was to occur, it would mean that either the buyer or seller will not receive any tokens even though they already sent their tokens to the receiver, which will be devastating for both the user and the DEX.

If there are no safeguards for these overflow bugs, it would have disastrous effect on the DEX as it would mean the wrong calculation of the amount that was sent to the user during a trade. This would result in the loss of user's trust and damage our business reputation. In the previous case study, we have seen that these type of bugs does exist inside current implementations of smart contracts [10] and it is the responsibility of the developers to ensure that these bugs are removed.

C. *Vulnerabilities In Code in case of DEX Contract*

Coding a smart contract is the same as coding any software. This means that smart contracts are susceptible to the same issues faced by any programmers coding a software. One of the issue is the existence of bugs inside the smart contract due to human error. These bugs could be a vulnerability that attackers could exploit and steal funds.

In a previous study, it was found that out of 1,311 smart contracts, only 11.63% of these contracts are bug-free [10] with the rest having some form of low-risk to high-risk vulnerabilities.

This can also be seen by the number of hacks that has taken place since DEXs was created. Due to vulnerabilities inside the smart contracts, 2 DEXs was hacked and had their cryptocurrencies stolen, with one taking place in 2018 and the other taking place just early this year, April 2020 [11, 12]. This means that attackers are always looking for opportunities to steal user's funds and having our code publicly available gives them an advantage.

Our DEX is no exception to this issue as we are all humans. It is impossible for us not to make mistakes, but it is possible for us to prevent this from happening.

IV. Proposed Solutions

A. *Solution to Front Running Attack*

One of the possible solution to front-running attack is to have a hybrid decentralized & centralized exchange [13]. Front-running is only possible due to the public nature of the order book in the Ethereum Blockchain. If we can hide this order book from the eyes of the attacker, it would be impossible for them to front-run the orders.

This hybrid exchange would have 2 layers, an on-chain and off-chain layer. Order books and matching services are stored off-chain and the exchange contract is stored on-chain to ensure trade rules are enforced and user's rights are protected. This not only prevents front-running, but also have the added benefit of eliminating the confirmation waiting time for an order since this is done off-chain. Users would not need to wait until the transaction is confirmed and this will improve the user experience.

If decentralization is our priority, another possible solution is to change our current implementation of the DEX. This new implementation would allow users to find counterparties to trade privately instead of relying on the order book to facilitate trades [14].

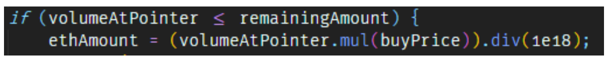
An indexer aggregates the 'intent to trade' from both makers and takers and only open up a channel of communication between them once they are matched. Once they are satisfied with the price, the trade will be executed. Due to the private nature of this implementation, it would be impossible for attackers to conduct a front-running attack.

Currently our DEX does not have any prevention for front-running attacks but if we were to implement one, we would choose the second solution as we want to keep our exchange decentralized.

B. *Solution to Overflow & Underflow Bug*

The solution for these bugs is to ensure that every arithmetic operations are checked for overflow and underflow before continuing the execution of the function. This can be done manually by the programmer or by utilizing the SafeMath library which helps us conduct the overflow and underflow check for every arithmetic operations.

As this is one of the key concerns of our DEX, we have replaced all arithmetic operations with the helper functions from the SafeMath library. This ensures that our DEX will never have overflow or underflow bugs and our trade will always be executed correctly.



```
if (volumeAtPointer ≤ remainingAmount) {
    ethAmount = (volumeAtPointer.mul(buyPrice)).div(1e18);
```

Fig. 1. Example of SafeMath Implementation in DEX.

In Fig. 1, it shows a snippet of the Buy Limit Order function of our DEX. Inside this snippet, we are calculating the total WETH amount that would be required for this buy order trade by multiplying the total volume inside a particular sell order by the buying price. We then divide the result by 1e18 in order to get the correct WETH amount as we are using wei inside the smart contract.

For both multiplying and division, we are using the SafeMath helper functions to ensure that overflow does not occur. If it does, an exception would occur, and the transaction will be reverted. This will ensure that each buy and sell order is correctly executed and users will not receive the wrong number of tokens during a trade.

C. Solution to Vulnerabilities In Code

To ensure that there are no vulnerabilities in our DEX contract, we must conduct auditing of our smart contract. Auditing must not be done internally as there could an oversight on our part. Thus, we would have to employ a third-party audit firm to conduct a security audit.

One of the most reputable audit firm is OpenZeppelin and they are well known for their security audits for famous projects. Some of the projects they have conducted security audits for includes Augur, Brave and even Solidity Compiler [15]. As such, this would be one of the audit firms that we could use for our smart contract auditing.

To be on the safe side, it would be better to conduct multiple audits with various audit firms. This would ensure that our smart contract is free of bugs which would prevent any vulnerabilities from being exploited.

As this is a module assignment, we do not have the capital to hire experts to audit our code and thus, we were only able to manually audit it ourselves.

V. Conclusion

In conclusion, this term paper is solely focused on the Security issues of our development project, Something Funky is Nice DEX, a DEX which is a DApp on the Ethereum network. For a DEX, Security is one of the highest priority as it deals with the user's money and it is the responsibility of the DEX to ensure the safety of the user's funds. Failure to ensure Security will result in us losing the user's trust which will be devastating to the DEX and our business. Thus, we have decided to focus on the Security issues of our development project and not on Privacy and Scalability.

In this term paper, we have analyzed 3 different types of vulnerabilities that could be exploited on our DEX by attackers in order to gain monetary benefits.

The first vulnerability is front-running attacks on our DEX. Based on previous studies conducted, the authors were able to successfully execute front-running attacks on DEXs that are live on the Ethereum network. This means that it would be possible to conduct the same kind of attack on our DEX.

The second vulnerability is Overflow and Underflow bugs inside the smart contract. These type of bugs are the most devastating bugs for a DEX as it deals with numbers and user's funds. An overflow or underflow occurring during the calculation of a trade will result in the wrong amount being sent to the user. If this occurs, we would lose the user's trust and our DEX's reputation will be tainted. Based on previous study conducted, we have seen that overflow bug does exist in current implementations of smart contracts due to oversight of the developers.

The third vulnerability is vulnerabilities in smart contracts. We have seen multiple cases of smart contracts being exploited of their vulnerabilities because of flaws in the smart contracts. This resulted in the loss of user's funds which was a substantial amount. The user's trust in the DEX was also lost in the process.

To prevent attackers from exploiting these vulnerabilities, we came up with solutions that would solve these vulnerabilities.

The solution to the first vulnerability would be to either change our DEX into a hybrid exchange or to change to a private implementation of our system. Both of these would make it impossible for front-running attacks to occur due to the private nature of the order book or implementation.

The solution to the second vulnerability is to use the SafeMath library's helper functions for every arithmetic operations in our smart contracts. This will ensure that overflow and underflow will not occur during the calculation of a trade and if it does, an exception would occur and revert the transaction.

The solution to the third vulnerability is to conduct security audits on our smart contracts by employing reputable audit firms like OpenZeppelin. Conducting multiple audits by various audit firms will ensure that our smart contract would be free from exploits.

Out of these solutions, we have implemented only SafeMath into our smart contract. The reason we did not implement the other solutions is due to budget and time constraints.

References

[1] S. Nakamoto,"Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] X. Yang, Y. Chen and X. Chen, "Effective Scheme against 51% Attack on Proof-of-Work Blockchain with History Weighted Information," *2019 IEEE International Conference on Blockchain (Blockchain),* Atlanta, GA, USA, 2019, pp. 261-265, doi: 10.1109/Blockchain.2019.00041.

[3] V. Buterin, "Ethereum Whitepaper," ethereum.org, 2013. [Online]. Available: https://ethereum.org/en/whitepaper/. [Accessed: 19-Nov-2020].

[4] CZ4153 Module 4 Lecture 1,2,3

[5] CZ4153 Module 4 Lecture 4

[6] CZ4153 Module 4 Lecture 7, 8, 9

[7] "DeFi Pulse: The DeFi Leaderboard: Stats, Charts and Guides," DeFi. [Online]. Available: https://defipulse.com/. [Accessed: 19-Nov-2020].

[8] P. Hao, V. Chang, S. Lu, C. Zhang, "Decentralized Cryptocurrency Exchange Security Analysis" Journal, 2018

[9] A. Sobol, "Frontrunning on Automated Decentralized Exchange in Proof Of Stake Environment", IACR Cryptol. ePrint Arch., 2020, pp. 1026

[10] T. Min and W. Cai, "A Security Case Study for Blockchain Games," 2019 IEEE Games, Entertainment, Media Conference (GEM), New Haven, CT, USA, 2019, pp. 1-8, doi: 10.1109/GEM.2019.8811555.

[11] J. Russell, "The crypto world's latest hack sees Bancor lose $23.5M," TechCrunch, 10-Jul-2018. [Online]. Available: https://techcrunch.com/2018/07/10/bancor-loses-23-5m/. [Accessed: 19-Nov-2020].

[12] P. Baker, "Hacker Exploits Flaw in Decentralized Bitcoin Exchange Bisq to Steal $250K," CoinDesk, 08-Apr-2020. [Online]. Available: https://www.coindesk.com/hacker-exploits-flaw-in-decentralized-exchange-bisq-to-steal-250k. [Accessed: 19-Nov-2020].

[13] Y. Hsieh, C. Hsueh and J. Wu, "The Exchange Center: A Case Study of Hybrid Decentralized and Centralized Applications in Blockchain," 2018 1st IEEE International Conference on Hot Information-Centric

Networking (HotICN), Shenzhen, 2018, pp. 232-233, doi: 10.1109/HOTICN.2018.8606010

[14] Totle, “Front Running and its Effect on Decentralized Exchanges,” Medium, 12-Sep-2019. [Online]. Available: https://medium.com/totle/front-running-and-its-effect-on-decentralized-exchanges-e463ca4474db. [Accessed: 19-Nov-2020].

[15] “Security Audits,” OpenZeppelin. [Online]. Available: https://openzeppelin.com/security-audits/. [Accessed: 20-Nov-2020].