Decentralized Cryptocurrency Exchange Security Analysis

Date: 2018/5/10

Author: Parker Hao, Vincent Chang, Shao Lu, Chenxing Zhang

E-mail: phao@mit.edu, vinchang@mit.edu, lushao@college.harvard.edu, chenxing@mit.edu

Contents

1 Introduction

1.1 The Background

We have witnessed a substantial increase and growth over the past years in cryptocurrency infrastructure, with exchanges being the most popular one. Within the digital currency market, a cryptocurrency exchange is a platform that enables the conversion of fiat money to cryptocurrency and vice-versa as well as the conversion of different cryptocurrency pairs. However, cryptocurrency exchanges are constantly being hacked and just two months ago, 500 million dollars were stolen from Japans second largest cryptocurrency exchange, Coincheck. Security issues on exchanges have been the most critical discussions in the space and one the biggest concerns for cryptocurrency investors.

Despite the attention on centralized cryptocurrency exchanges, we would like to shift the focus to a category of exchanges that have become increasingly popular, decentralized exchanges. A decentralized cryptocurrency exchange is one in which the architecture of the platform has no central controlling server (or bundle of servers). As a result, no third-party escrow intermediary is required to hold the funds of the participants in the exchange transaction.

Given that it is a relatively new and immature technology, we would like to analyze the security risks involved in decentralized exchanges by trying to penetrate with different methodologies, and suggest possible improvements in the systems security.

2 Architecture

2.1 Overview

A crypto currency exchange is mainly composed of few main parts: order placement, order cancellation and order matching. Decentralized exchanges aim to broadcast each part of the transactions to a decentralized network such that no centralized third party server needed in the trading system. This paper will focus on analyzing the architectures of decentralized exchanges built on top of the Ethereum, one of the most popular and wildly used decentralized network.

Ethereum is a project which attempts to build the gen- eralised technology; technology on which all transaction- based state machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto un- explored compute paradigm in the mainstream: a trustful object messaging compute framework.

Ethereum, taken as a whole, can be viewed as a transaction-based state machine: we begin with a gen- esis state and incrementally execute transactions to morph it into some final state. It is this final state which we accept as the canonical version of the world of Ethereum. The state can include such information as account bal- ances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admis- sible. Transactions thus represent a valid arc between two states; the valid part is importantthere exist far more invalid state changes than valid state changes. Invalid state changes might, e.g., be things such as reducing an account balance without an equal and opposite increase elsewhere. A valid state transition is one which comes about through a transaction.

Transactions are collated into blocks; blocks are chained together using a cryptographic hash as a means of refer- ence. Blocks function as a journal, recording a series of transactions together with the previous block and an iden- tifier for the final state (though do not store the final state itselfthat would be far too big). They also punctuate the transaction series with incentives for nodes to mine. This incentivisation takes place as a state-transition function, adding value to a nominated account.

A transaction (formally, T) is a single cryptographically-signed instruction constructed by an actor externally to the scope of Ethereum. While it is assumed that the ultimate external actor will be human in nature, software tools will be used in its construction and dissemination1. There are two types of transactions: those which result in message calls and those which result in the creation of new accounts with associated code (known informally as contract creation).

Mining is the process of dedicating effort (working) to bolster one series of transactions (a block) over any other potential competitor block. It is achieved thanks to a cryptographically secure proof. This scheme is known as a proof-of-work.

2.2 Transaction Execution

For a transaction to be executed, it must meet the following requirements:

- The transaction must be a well-formatted RLP.
- The transaction signature is valid.
- The transaction nonce is valid.
- The gas limit is at least as big as the intrinsic gas required for the transaction.
- The sender has enough Ether to cover the upfront payment, which consists of the gas fee and the value to be sent.

(Note: If the transaction turns out to be invalid due to sender's insufficient balance, the gas fee will not be refunded, because the miner has already started to use his/her computational power to mine this transaction. Thus the gas fee will go to the miner's address no matter a transaction is valid or not.)

The transaction process is made up of multiple phases. It begins with deducting the upfront payment from the sender's balance, and incrementing the sender's nonce by one. Then computations required for the transaction start to be processed. During the entire transaction execution, Ethereum keeps track of the transaction substate, which consists of self-destruct set, log series, the set of touched accounts and refund balance, for immediate information retrieval after the transaction.

- Self-destruct Set: a set of accounts to be discarded after the transaction.
- Log Series: a series of indexable and archived 'checkpoints' in VM code execution that allows third parties, such as decentralized exchange market front-ends like OasisDEX, to easily keep track of contract-calls.
- Touched Accounts: the set of touched accounts are updated in the transaction substate so that empty accounts can be deleted at the end of transaction.
- Refund Balance: The amount of money to be refunded to the sender. Refund counter is initialized to be zero, and gradually increases as contract storage resets to zero. It can only partially offset the gas fee. The sender is only refunded at the end of the transaction.

Once all steps and computations are finished, the sender is refunded unused gas and rewarded money for clearing database storage if there is any. Then the miner receives the gas in Ether, and the amount of gas used in this transaction is added to block gas counter for block validation.

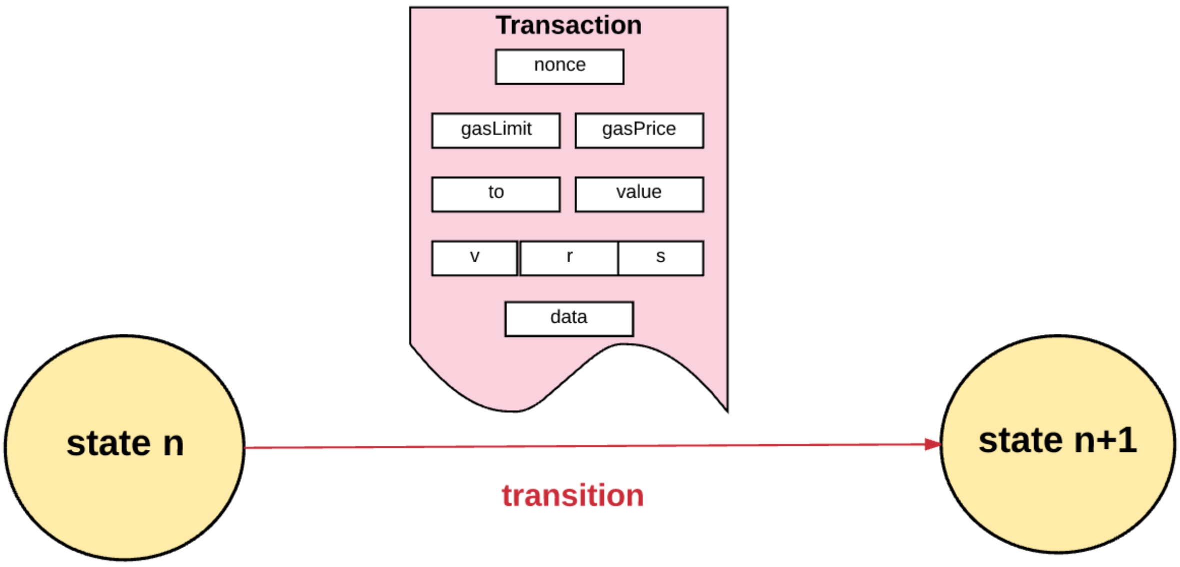


Figure 1: State change after transaction[2]

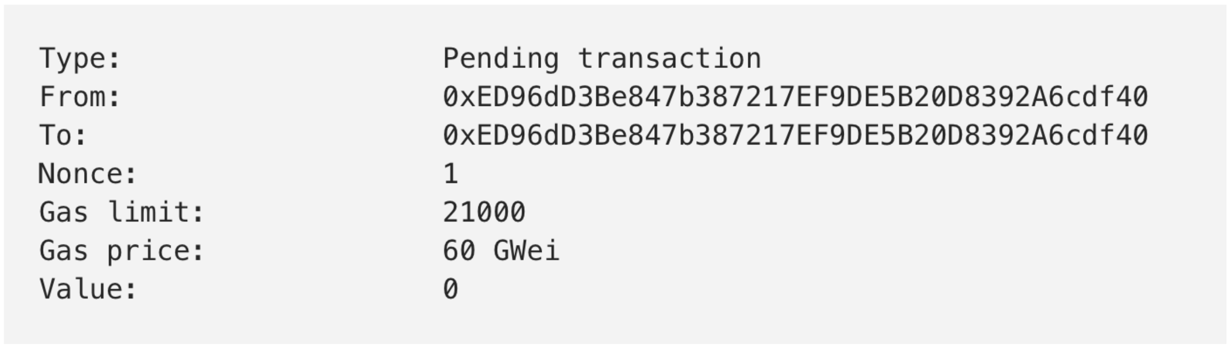
2.3 Order Cancellation

Although transactions on the Ethereum network cannot be cancelled explicitly, they can nonetheless be invalidated if the sender broadcasts another transaction with the same nonce but at a higher gas price, for only one transaction with a certain nonce can be included in the blockchain. In order for the new transaction that the sender uses to overwrite some old transaction to be accepted, the gas price of the new transaction that he/she places needs to be at least 10% higher than the previous gas price.

In the example shown below, the sender cancels the previous order by placing another order sending 0 ether to himself/herself. Because the gas price of the new order is much higher than that of the old order, it is more likely to be mined first. Once it's mined, any other transactions that have nonce value of 1 will become invalid. Therefore, the old transaction is successfully cancelled.

```
Type:                   Pending transaction
From:                   0xED96dD3Be847b387217EF9DE5B20D8392A6cdf40
To:                     0xa34C6BCAe6F46ac6470443CCea67d937f6060c7E
Nonce:                  1
Gas limit:              21000
Gas price:              1 GWei
Value:                  0.005 Ether
```

Figure 2: Transaction to be cancelled[3]



```
Type:                  Pending transaction
From:                  0xED96dD3Be847b387217EF9DE5B20D8392A6cdf40
To:                    0xED96dD3Be847b387217EF9DE5B20D8392A6cdf40
Nonce:                 1
Gas limit:             21000
Gas price:             60 GWei
Value:                 0
```

Figure 3: Overwriting transaction[4]

2.4 Order Matching

Order Matching algorithms can generally be divided into 3 categories: 1. FIFO/Time Priority/Price-Time 2. Pro Rata 3. Time Pro Rata. Oasis used Time Pro Rata.

1. **FIFO**: There are algorithms. First one is **Allocation**: Full order allocation offered to the oldest (priority) order at the best bid/offer. Second one is **Priority**: Explicit outright and strategy orders Priority is given to the oldest explicit outright strategy order at the best bid/offer. Implied outright and strategy orders Explicit orders have priority over implied orders. Among implied orders, priority is granted to the order with the oldest constituent parent order.

2. **Pro Rata**: The objective of the pro-rata algorithms is to divide incoming Order between the orders at the same price level, with the volume of the orders allocated to each order being in proportion to the amount of volume they have in the market at that price i.e. its a pure Ratio based algorithm. There are several variations of the pro-rata algorithm used by some specific products.

- vanilla Pro-rata
- Pro-rata with priority order allocation and no volume cap or minimum volume
- Pro-rata with priority order allocation and a volume cap
- Pro-rata with priority order allocation and minimum volume requirement
- Pro-rata with a priority order allocation, a volume cap and minimum volume requirement.

3. **Time Pro Rata**: Fractional allocations are rounded down to the nearest integer for all allocations greater than 1 and rounded up to 1 for all fractional allocations less than 1. For equally-sized fractional allocations, priority is granted to the oldest order. If any volume remains unallocated following this sequence (for instance, as a result of rounding, or when the calculated allocation for an order is constrained by the MIN function), then a further pass of the sequence will occur.

2.5 Gas Fee and Miners

In order to prevent users from abusing the network, all computations on the Ethereum network are charged a certain amount of fees measured in units of gas. Every transaction on the Ethereum network comes with gas fee, which is determined by gas price and gas limit. Gas price is the number of Wei per unit of gas. Gas limit is the maximum amount of gas the sender agrees to spend on executing this transaction. The product of the two represents the maximum amount of Wei the sender is willing to pay. If the sender wants his/her transaction to be mined first, he/she usually sets a high gas price to attract miners to prioritize his/her transaction. Thus, the sender has a trade-off to make between minimizing the gas price, and maximizing the chance of his/her transaction being mined in time.

Besides transaction, an increase in the usage of memory also incurs a gas fee. To minimize the size of Ethereum state database on all nodes, the gas fee for a transaction that clears an entry is not only waived, a qualified refund is also given.

Similar to that in Bitcoin's blockchain, a transaction on the Ethereum network is only considered to be valid if it can pass the validation process, which is conducted by miners. Each miner who chooses to mine this particular transaction provides a mathematical proof to create a block of valid transactions. In order to have his/her block be added to the main blockchain, the miner needs to prove it faster than others.

3 Decentralized Exchange Interface Attacks

Since the concepts of decentralized exchanges only came up in recent years [Best Decentralized Exchanges Which You Can Use To Trade Right Now, CoinSutra, 2018, https://coinsutra.com/best-decentralized-exchanges-dex/], the security analysis works on this topic are still very limited and almost no academic studies can be found from outside sources. Therefore, this paper will focus on examining different possible security issues of decentralized cryptocurrency exchanges from both non-academic sources and our designed experiments.

3.1 Attacks on Metamask - the web browser embedded Ethereum wallet

Metamask is a popular Google Chrome application connecting the web browser and the Ethereum wallet, and it is used by most of the decentralized exchanges. Although Metamask provides a strong and reliable connection between Ethereum wallet and the decentralized exchange, its users still suffer from potential attacks on this additional component.

Common attacks on Metamask include phishing user wallet information and triggering unintended user actions. Metamask information phishing usually happens when users forgot to lock their Metamask accounts. A web interface illustration of Metamask is shown in **Figure 1**. Users need to type in their passwords in order to sign in to their Metamask wallets and see their wallet information (different types and amounts of tokens, as shown in **Figure 2**). However, it is common that users forget to log out of their accounts after visiting decentralized exchanges. In this case, all webpages where the user visit can have the access to the user wallet information and all the associated trading history.

The other common type of attacks is to trigger users to do unintended actions by displaying fake requests. As shown in **Figure 3**, if the confirmation requests come from a malicious server, the users can get triggered to do unintended actions. As the Metamask notification

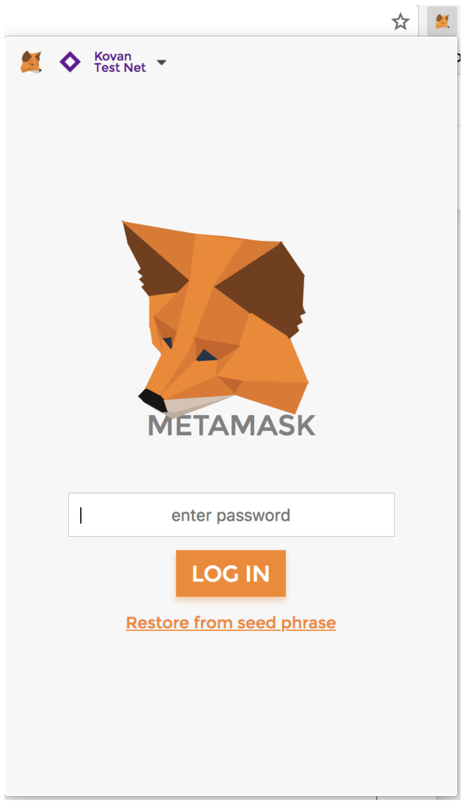


Figure 4: Metamask Login Interface

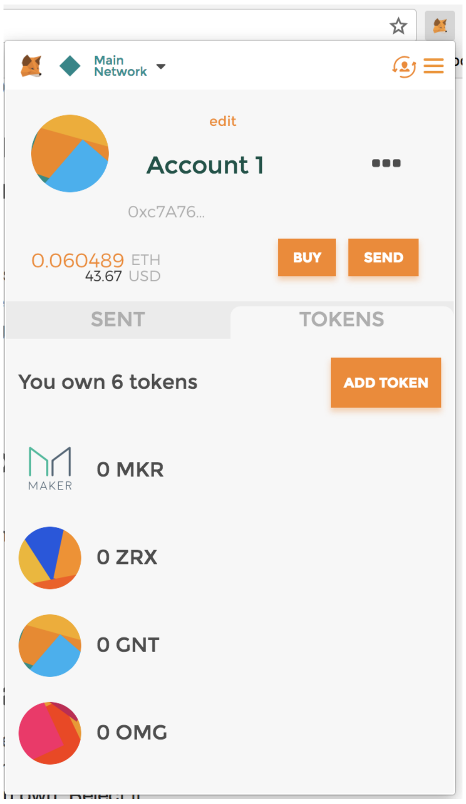
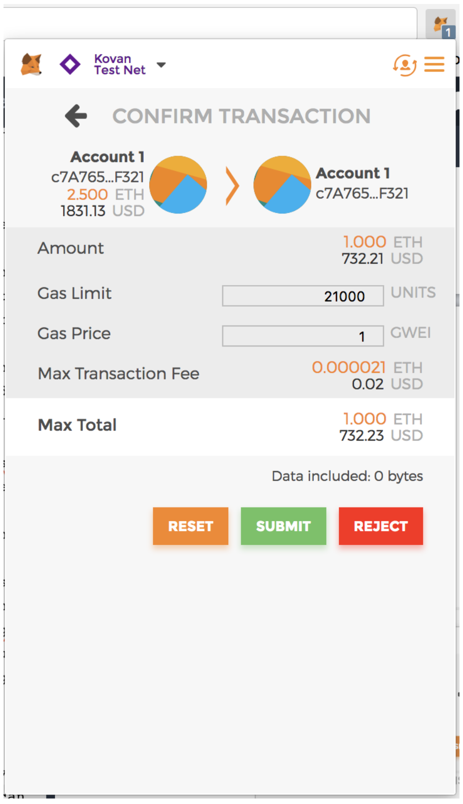


Figure 5: Metamask Wallet Interface



Account 1
c7A765...F321
2.500 ETH
1831.13 USD

Account 1
c7A765...F321

Amount	1.000 ETH 732.21 USD
Gas Limit	21000 UNITS
Gas Price	1 GWEI
Max Transaction Fee	0.000021 ETH 0.02 USD
Max Total	1.000 ETH 732.23 USD

Data included: 0 bytes

Figure 6: Metamask Confirmation Notification

is composed by common Javascript interfaces, it is hard for users to distinguish fake notifications from the real ones.

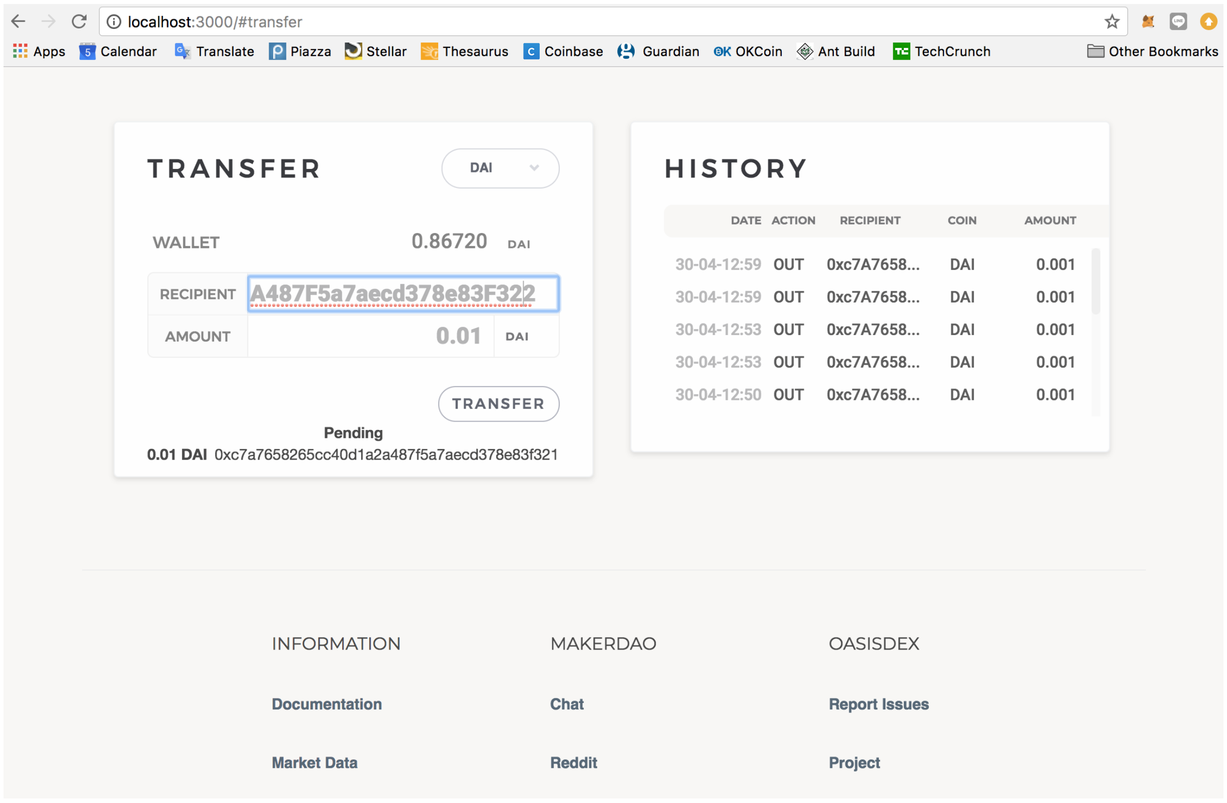
3.2 Deceptive Phishing on Decentralized Exchange

Deceptive phishing refers to any attack where the attackers impersonate a legitimate website and steal users' information or provide them wrong instructions. The attack is commonly combined with instant message or email to trigger users to visit the compromised website.

For decentralized exchanges, attackers can also set up phishing websites with similar user interface and web domain to deceive users. From an implementation standpoint, this type of attack can be simpler because the code sources of the decentralized exchanges are open to public and the attackers can clone the repository and make changes to it.

This paper explores the experiment of a deceptive phishing on the decentralized exchange Oasis. Since the decentralized exchange codes are open-source and open to the public, it is relatively simple to learn the data flow within the decentralized exchange and design phishing attacks based on the code source.

Our approach of phishing is to trick users to send their tokens to unintended third parties. This can be accomplished by changing the transfer event function in the source codes as in **Listing 1**. After the code changes, the decentralized exchange users will unintentionally send all their tokens to the attacker and the hack is hard to detect from the user interface.



TRANSFER DAI

WALLET 0.86720 DAI

RECIPIENT A487F5a7aecd378e83F322

AMOUNT 0.01 DAI

TRANSFER

Pending

0.01 DAI 0xc7a7658265cc40d1a2a487f5a7aecd378e83f321

HISTORY

DATE	ACTION	RECIPIENT	COIN	AMOUNT
30-04-12:59	OUT	0xc7A7658...	DAI	0.001
30-04-12:59	OUT	0xc7A7658...	DAI	0.001
30-04-12:53	OUT	0xc7A7658...	DAI	0.001
30-04-12:53	OUT	0xc7A7658...	DAI	0.001
30-04-12:50	OUT	0xc7A7658...	DAI	0.001

INFORMATION	MAKERDAO	OASISDEX
Documentation	Chat	Report Issues
Market Data	Reddit	Project

Figure 7: Oasis Decentralized Exchange Deceptive Phishing

As shown in **Figure 4**, no matter what an user inputs as a recipient of the token transfer event, the token will be sent to the hacker's preset address.

Listing 1: Transfer Event Code Changes

```
transfer(event) {
    function process(transaction) {
−       let recipient = transaction.recipient();
+       MY_ADDRESS = '0xc7A****************F321';
+       let recipient = MY_ADDRESS.toLowerCase();
        if (!(/^0x/.test(recipient))) {
            recipient = '0x'.concat(recipient);
        }
}
```

4 Trading Engine Attack

4.1 Front Running

Front-running is an investing strategy that predicts the impact of upcoming trades on the price of a security. For ordinary securities, it is illegal for traders to practice front-running using prior trading information about their own or the others, which is punished by the Securities and Exchange Commission. It can also be interpreted as a variant of "insider trading".

A example would be the HSBC foreign exchange market front-running scandal. In December 2011, one HSBC client was looking to exchange about $3.5 billion for British pounds, a fairly large transaction. However, two bank employees bought British pounds before placing the clients order, expecting that the large transaction would raise the price of the pound and they can thus profit with illegal market-moving information.

For centralized cryptocurrency exchanges, front-running is not uncommon. Hackers have previously stolen bitcoins and then profited by shorting the relevant cryptocurrencies given that their anticipated hacking events will drive down the prices of the related cryptocurrencies.

Please refer to figure 1 for the general process of front-running. Here we will also use an example. If Eve places a sell order for one token at 100 dollars, suddenly a big player, Alice, comes in with a larger buy order at 120 dollars for each token. If Eve scans the ethereum network quickly enough, she is able to see the larger buy order's broadcast before their order match and the actual tokens are verified and transferred on the network. Given that Eve definitely prefers selling her token while receiving 120 dollars rather than 100 dollars, she will cancel her order immediately upon observing the large buy order. If she pays a higher gas fee, her cancellation will be more likely to be verified by the miners before the order matching engine executes Eve's sell order. She can subsequently place a sell order again, but this time at 120 dollars for her token.

4.2 Front Running Experiments and Results

We have conducted dozens of manual experiments on the decentralized exchange OASIS and on the trading pair W-ETH/DAI. (Using OASIS's API can save tons of time and efforts to automate trading, however, to truly test on a decentralized trading and its metrics, we chose to not use the centralized API endpoints) In our experiments, We mainly test the delay and success rate of canceling orders upon receiving additional information about future updates to the order-book as illustrated above.

Because of the limited budget of the project, we choose to use the Ethereum Kovan test-net instead of the main net. However, due to the crowded traffic on the main net, test-net typically runs faster and thus generally decreases the time it takes to place or cancel an order, which will be covered subsequently.

The experiment consists of three stages, first, we test the time delay for canceling orders in correlation with the gas fee on the Kovan test net. Second, by choosing different pairs of the gas fee, we can approximate the delay time. At each different delay time period, we test the success rate of canceling orders using those pairs of the gas fee to induce front-running and simulate fake gains in profits. Lastly, we evaluate whether the additional gas fees is worth the chance for front-running in the main net trading environment with real capital.

Below is the chart for testing different gas fees and their corresponding delay time for canceling orders. The data generally follows a decreasing convex curve.

From the graph, intuitively it makes sense because the higher gas fee that traders pay, the

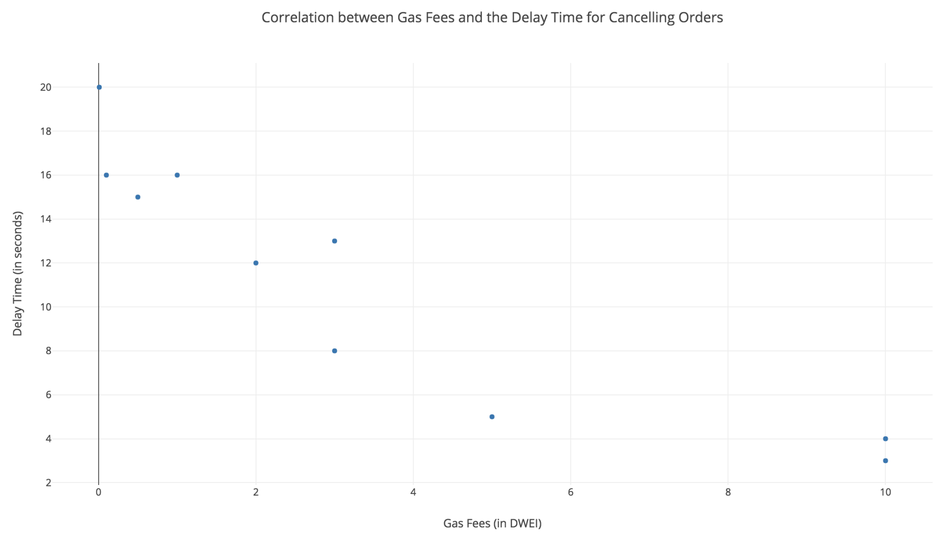


Figure 8: Correlation between Gas Fees and Delay Time in Executing Cancel Orders

faster that their transactions will be verified by the miners. However, when the gas fee keeps increasing from a threshold, the time delay doesn't decrease that much because it is mostly limited by the actual broadcast time. In other words, additional incentives beyond the threshold does not yield extra benefits.

In the second experiment, we choose 1 second and 5 seconds for the delay time. Note that this delay time is different than the delay time for cancelling orders, which is incurred by the miners. This delay time means the time lag from when one places an order, to when the counter party decides to cancel his order to front run and gain more profits. For the 1 second delay, the counter party almost succeeds at every time while paying a gas fee ratio at 10:1. For the 5 second delay, the counter party succeeds with a rate of approximately 24% while paying a gas fee ratio at 10:1. Using Etherscan, market participants are able to know about the upcoming transactions within 5-10 seconds. However, Etherscan is a free service and if people choose to monitor their own ethereum network in real time, that time delay of observing transactions can be reduced further. Thus, in the real main network, front running by cancelling orders can happen fairly frequently and with a non-negligible success rate, greatly impacting the fairness of the market.

Lastly, we compared with the profits that was generated through cancelling orders with the gas fees. Surprisingly, these two amounts are equal, meaning that we either gained or lost assets on the Kovan test net. However, the average order size that was used in the experiment is on the magnitude of 0.01 - 0.05 ethereum, which is significantly smaller than the average size of a buy/sell order on the main net. Thus, front running on decentralized cryptocurrency exchanges is still a lucrative scheme, but one that violates the fairness of the market and the security of decentralized cryptocurrency exchanges under weak regulations.

4.3 Miner Front Running

Miner front running is a special case of front-running, where the party with informational advantage is actually also the miner for the transaction. Miner front-running worsens the

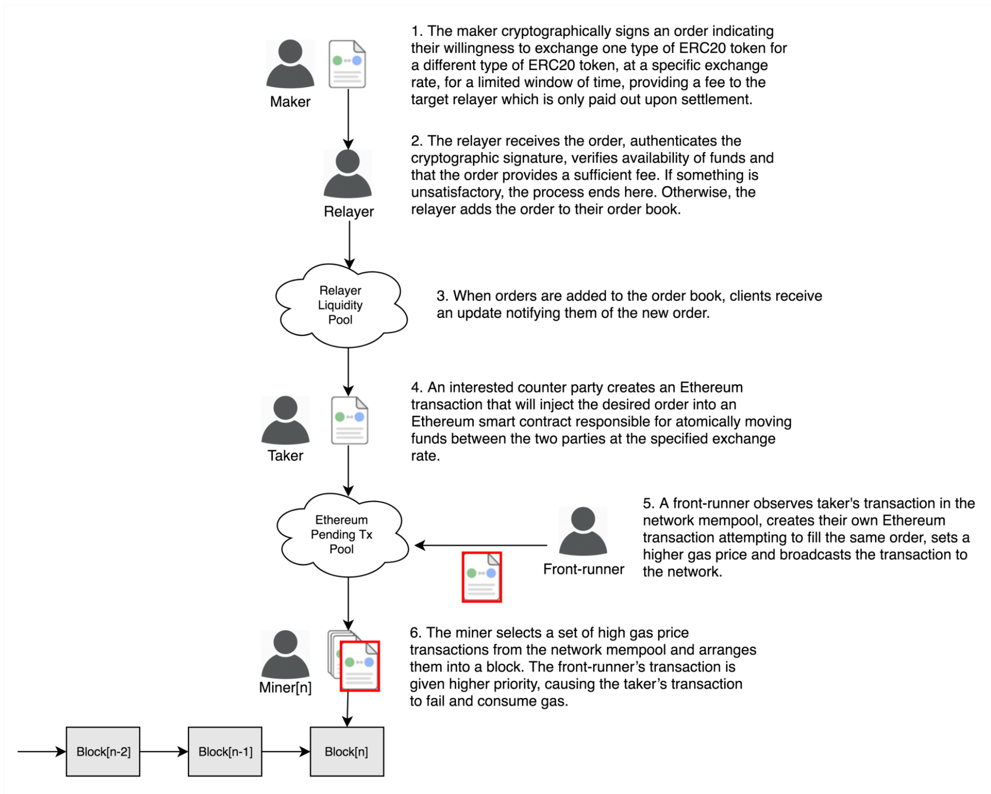

Maker
1. The maker cryptographically signs an order indicating their willingness to exchange one type of ERC20 token for a different type of ERC20 token, at a specific exchange rate, for a limited window of time, providing a fee to the target relayer which is only paid out upon settlement.
Relayer
2. The relayer receives the order, authenticates the cryptographic signature, verifies availability of funds and that the order provides a sufficient fee. If something is unsatisfactory, the process ends here. Otherwise, the relayer adds the order to their order book.
Relayer Liquidity Pool
3. When orders are added to the order book, clients receive an update notifying them of the new order.
Taker
4. An interested counter party creates an Ethereum transaction that will inject the desired order into an Ethereum smart contract responsible for atomically moving funds between the two parties at the specified exchange rate.
Ethereum Pending Tx Pool
Front-runner
5. A front-runner observes taker's transaction in the network mempool, creates their own Ethereum transaction attempting to fill the same order, sets a higher gas price and broadcasts the transaction to the network.
Miner[n]
6. The miner selects a set of high gas price transactions from the network mempool and arranges them into a block. The front-runner's transaction is given higher priority, causing the taker's transaction to fail and consume gas.
Block[n-2]
Block[n-1]
Block[n]

Figure 9: General Process of Front Running for ERC20 Tokens [1]

implications of the transaction because miners will be able to simply ignore the cancellation order and enforce the transaction with the gas fee which is paid to the miner itself as well. However, due to limited mining power, we couldn't test the consequences of miner front-running comprehensively.

4.4 Liquidity Limitations

Liquidity refers to the extent to which a market allows assets to be bought and sold at stable prices. Lower liquidity tends to result in a more volatile market (especially when large orders are placed), and it causes prices to change more drastically; whereas higher liquidity creates a less volatile market, and prices do not fluctuate as significantly.

Major cryptocurrencies typically have higher liquidity. However, cryptocurrencies traded on decentralized exchanges have much lower liquidity given the slower trade matching engine and the transfer of actual tokens on the ethereum network.

The following is a table listing the top five decentralized cryptocurrencies ranked by market capacity as of May 13th, 2018. The volume and market capacity percentage for bitcoin and ethereum typically ranges from 5% to 10%, while the popular decentralized cryptocurrencies have a significantly smaller volume that is traded on a decentralized basis. (EOS and TRX have much higher percentages because most of their volume come from centralized exchanges)

Token Symbol	Market Capacity	24h Volume	Percentage
EOS	$12,511,849,091	$1,680,910,000	13.434545%
TRX	$4,921,640,645	$582,420,000	11.8338587%
VEN	$2,460,801,736	$30,133,000	1.224519617%
ICX	$1,570,819,454	$43,325,000	2.758114555%
BNB	$1,531,779,800	$39,374,500	2.570506544%

5 Conclusion

Overall, decentralized cryptocurrency exchanges and cryptocurrencies are still in their infancy and relatively immature. There are common attacks such as phishing, which can cause significant losses for the decentralized concept of trading. In addition, the trading engine itself is also subject to various attacks that cannot be dealt with conventional solutions, such as the fundamental issues with front-running. Thus, more innovations are needed for decentralized cryptocurrency exchanges to offer truly reliable trading.

6 Contribution

Vincent Chang wrote the architecture overview and developed the Metamask wallet and the decentralized exchange interface attacks.

Parker Hao focused on the trading engine attacks, especially the front running attacks. He managed the experiments, collected the results, and analyzed them under different scenarios. He also looked into miner front running and the liquidity limitations of decentralized cryptocurrencies and wrote the section of trading engine attack.

Lu Shao attempted DDoS attack, and wrote transaction execution, order cancellation, and gas fee and miners under the architecture section.

Chenxing Zhang researched architecture of ethereum and order matching algorithms under the architecture section.

References

[1] Front-running, Griefing and the Perils of Virtual Settlement (Part 1): `https://blog.0xproject.com/front-running-griefing-and-the-perils-of-virtual-settlement`

[2] How does Ethereum work, anyway?: `https://medium.com/@preethikasireddy/how-does-ethereum-w`

[3] Releasing Stuck Ethereum Transactions: `https://medium.com/@jgm.orinoco/releasing-stuck-ether`

[4] Releasing Stuck Ethereum Transactions: `https://medium.com/@jgm.orinoco/releasing-stuck-ether`

Ethereum: A secure decentralised generalised transaction ledger. Gavin Wood. BYZANTIUM VERSION f72032b - 2018-05-04. cryptopapers.net Ethereum Project Yellow Paper.