Frontrunning

No Author Given

Concordia University

Abstract.

1 Introductory Remarks

1.1 What is frontrunning?

1.2 Miners and their power

2 Related Work

3 Comparison Framework

Solutions:
Criteria:

4 Frontrunning Attacks

4.1 Financial Markets

4.2 Non-financial Applications

Applications, Ghazal, register domains before the user, (domain squatting?)
Other applications (look at Dapps or other blockchain use cases)
Arbitrage (buy before the order, sell to the original order) (other scenarios)
Maker griefing (attack on the system?s reputation itself)
Etherdelta case: Fill the order when cancelling transaction is sent. What would be the profitable scenario here?

5 Implications

5.1 Historical evidence of such attacks:

There has been many incidents of frontrunning in real world blockchains specially where it facilitates monetary gain. Here we will analyze the evidence of these incidents:

Status ICO ICO, Initial Coin Offering, is one of the blockchain applications, specially blockchains such as Ethereum with smart contract capability. The common practice is to deploy a smart contract on the blockchain indicating the details of the ICO such as the trade ratio, when it starts and ends, and more details on how it will be capped. In June 2017, Status.im started their ICO and within 3 hours they reached the dynamic ceiling in place that triggered the end of the ICO, summing in 300,000 ether in funds, estimated at more than 200 million dollars at the time of their ICO. [**?**]. The idea behind Dynamic Ceilings is to make it more costly for larger contributors, in the form of transaction fees, which have to split their contributions to different addresses, with minimal impact to smaller contributors [**?**]. On the time of the ICO there were reports of Ethereum network being unusable and transactions were not confirming. Further study showed that some mining pools (: define mining pool) could have been manipulating the network for their own profit.

Bancor is an Ethereum-based application that allows users to exchange their tokens without any counterparty risk. This protocol aims to solve the cryptocurrency liquidity issue by introducing *Smart Tokens* [**?**]– ERC20 compatible tokens with a built-in liquidity mechanism that are always available to users. Smart Tokens can be bought and sold through the users smart contract at an automatically calculated price which displays supply and demand. Doing so, Bancor protocol provides continuous liquidity for digital assets without relying on an orderbook as there is no requirement to match sellers and buyers.
Front-running Bancor Recently, researchers have shown that Bancor is vulnerable to frontrunning attacks. Implemented on the Ethereum blockchain, when Bancor transactions are broadcast to the network, they sit in a pending transaction pool known as *mempool* waiting for the miners to mine them. Since Bancor handles all the trades and exchanges on the Ethereum blockchain (unlike other existing decentralized exchanges), these transactions are all visible to the public for 16s (the average Ethereum blocktime) before being included within a block. This leaves this blockchain-based decentralized exchange vulnerable to the blockchain race condition attack as attackers are given enough time to front-run other transactions and gain favourable profits [**?**]. Bancor frontrunning attacks can occur in two different ways:

- **Miner Frontrunning.** As mentioned, Bancor protocol uses an algorithm that automatically calculates the price of digital assets to provide better market liquidity. In the Bancor model, essentially buy orders increase the price of the tokens while sell order decrease it. Since the Blockchain miners are the only parties who can decide on the order of transactions within a block, they can easily intercept and reorder the pending transactions sitting in the mempool and profit from a guaranteed price-rise. For example, a miner learns about the pending *buy* transaction of 1000 Ether, based on the Bancor design, if this transaction goes through, it causes the price of Ether to increase. So the dishonest miner can step in front of this transaction and place his own buy order ahead of it. So he would simply create his *buy* of

1000 Ether and include it within a block and now he mines the previous *buy* transaction of 1000 Ether. Doing so, he would receive a better rate than other Bancor user, can sell the tokens he has received and gain a price advantage at the expense of others. Similarly, a dishonest miner can sell hi tokens in front, if he sees a pending *sell* transaction.

- **Non-miner Frontrunning.** Researchers have also shown that a regular non-miner user can also front-run Bancor. In order for the Bancor transactions to be executed on the Ethereum Virtual Machine (EVM), users have to pay for the computations in small amount of Ether called *gas* [**?**]. The price that users pay for transactions (a.k.a. transaction fees) can increase or decrease how quickly they are executed and included within the blocks by the miners. This is because the Ethereum miners consume resources to process the transactions and so receive the transaction fees after creating the blocks. Once seeing two identical transactions with different transaction fees, profit maximizers miners are free to mine select the transaction which offers the highest fee. Therefore, any regular non-miner users who run a full-node Ethereum client can modify the order of pending transactions by paying a more amount of gas *i.e.,* by monitoring the network, upon seeing a pending *buy* transaction which will further increase the price of the asset, a font-runner user can pay a higher gas price and send his transaction a head of that. By doing so, he achieves a better rate from any other Bancor users.

New attacks:
Ghazal

6 Mitigations

Commit / reveal methods

- Send the hash first, send the actual data after (sealed-bid auctions)
- Proof of burn methods (generate random addresses with no private keys)

Submarines

- The method described in HD can be optimized and more functional with the new changes to ethereum and solidity
- Submarines 2.0: There can be a new way of doing this (new WRT hackingditributed article), can predict smart contract addresses using the nonce and data of the contract (forwarder/refunder), so funds can be sent to these addresses and then contracts can be deployed
  - Pros: new addresses, no indication of the order (Anonymity-set)
  - Cons: DDoS? Requires 3 transactions for each order? ¿¿ 1 send the funds 2 Deploy the contract 3 call the function
- Other methods? Consensus protocol based solutions?

7 Concluding Remarks