THE UNIVERSITY of EDINBURGH
informatics

Front-Running Attacks in DeFi Network

Presenter: Yu SHEN

Outline

- What is Front-Running
- A Taxonomy of Front-Running Attacks in DeFi Network
- Preventing, Detecting or Mitigating Front-Running Attacks

Front-Running

- In front-running, a person sees a concrete transaction that is set to execute and reacts to it before it actually gets executed.
- Compare with insider trading & arbitrage:
  - Insider trading: the person has access to more general privileged information that might predict future transactions but is not reacting at the actual pending trades.
  - Arbitrage (legal): the person reacts after the trade is executed, or information is made public, and profits from being the fastest to react.

Traditional Front-Running

- Front-running originates on the Chicago Board Options Exchange (CBoE)*.

* J. W. Markham. Front-running-insider trading under the commodity exchange act. Cath. UL Rev., 38:69, 1988.

Traditional Front-Running

- The Securities Exchange Commission (SEC) in 1977 defined it as:

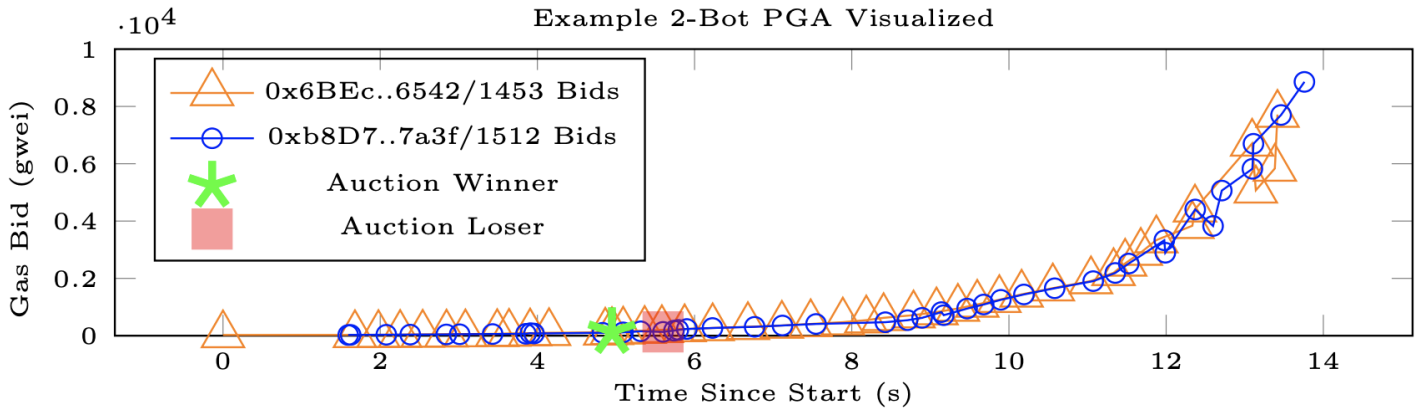
  > The practice of effecting an options transaction based upon non-public information regarding an impending block transaction* in the underlying stock, in order to obtain a profit when the options market adjusts to the price at which the block trades.

- The first front-running policies applied only to certain option markets. In 2002, the rule was expanded to cover all security futures.
- In 2012, it was expanded further with the new amendment, FINRA Rule 5270, to cover trading in options, derivatives, or other financial instruments overlying a security with only a few exceptions.

* A block in the stock market is a large number of shares, 10 000 or more, to sell which will heavily change the price.

Front-Running in DeFi

- Adversary can use Price Gas Auction (PGA) to front-run her transaction.

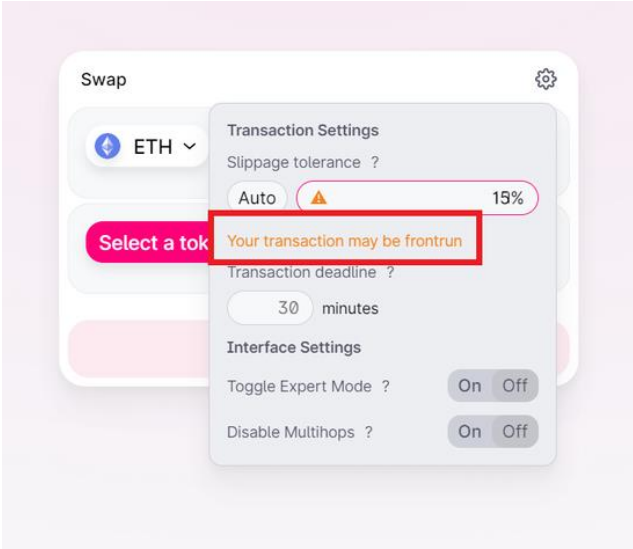


- Miners enjoy the full power to choose the order of the transactions in her mining block.

THE UNIVERSITY of EDINBURGH
informatics

Front-Running in DeFi

- Some exchanges may warn their users of being frontrun, though few attention paid to this risk.
- Early research* (2019) reports rampant adversarial manipulation of transactions in the Ethereum network by bots extracting upwards of USD 6M in revenue from unsophisticated users.
- Nowadays, front-running constitutes a large percentage of MEV (maximal extractable value) which yields USD 700+M since Jan 1, 2020**.



* Philip Daian et al. “Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability”. In: IEEE S&P. 2020, pp. 585–602.
** https://explore.flashbots.net/

Types of Front-Running*

1. Displacement Attack
   - It is not important to the adversary for victim's function call to run after the adversary runs her function. Victim's transactions can be orphaned or run with no meaningful effect.
2. Insertion Attack
   - After the adversary runs her function, the state of the contract is changed, and she needs the victim's original function to run on this modified state.
3. Suppression Attack
   - After the adversary runs her function, she tries to delay the victims from running their functions. After the delay, she is indifferent to whether their functions run or not.

* Eskandari S., Moosavi S., Clark J. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. Financial Cryptography and Data Security 2019.

Displacement Attack on Ethereum Name Service (ENS)

- Instead of allowing new .eth domain names to be purchased directly, ENS domains are put up for a sealed bid auction which seals the domain name in a bid (not the bid amount).
- All bidders (winners and losers) must send a transaction to reveal their bids for a specific domain or sacrifice their bid amount . If two bidders bid the same price, the first to reveal wins it.
- Most implementations are user friendly but less confidential:
  - Leaks the hash of the domain and the initial bid amount in the auction.
  - Original names can be guessed from the hashes (e.g., rainbow tables, used in ENS Twitter bot).
  - People can even bid on domains even though they do not know what they are because of speculation on its value.
- Front-running attacks are seen in regular (non-blockchain) domain registration.

Suppression Attack on Gambling DApps

- Fomo3D: a game style not based on random outcomes.
  - The aim is to be the last person to have purchased a ticket when a timer goes to zero in a scenario where anyone can buy a ticket and each purchase increases the timer by 30 seconds.
- On August 22, 2018, the first round of the game ended with the winner collecting 10,469 Ether*.
- Winning strategy:
  - The winner appears to have started by deploying many high gas consumption DApps unrelated to the game. When the timer of the game reached about 3 minutes, the winner bought 1 ticket and then sent multiple high gasPrice transactions to her own DApps.
  - These transactions congested the network and bribed miners to prioritize them ahead of any new ticket purchases in Fomo3D (Gas Auction).

* https://etherscan.io/tx/0xe08a519c03cb0aed0e04b33104112d65fa1d3a48cd3aeab65f047b2abce9d508

THE UNIVERSITY of EDINBURGH
informatics

Insertion Attack on DEX Users

- An example of a “sandwiching” attack.

Date	Type	Price USD	Price ETH	Amount MANA	Total ETH
2021-03-14 11:16:31	sell	$1.1469328	0.00060687	22,816.616	13.846757
2021-03-14 11:16:31	buy	$1.1595113	0.00061353	11,496.844	7.0536272
2021-03-14 11:16:31	buy	$1.1422316	0.00060438	22,816.616	13.79

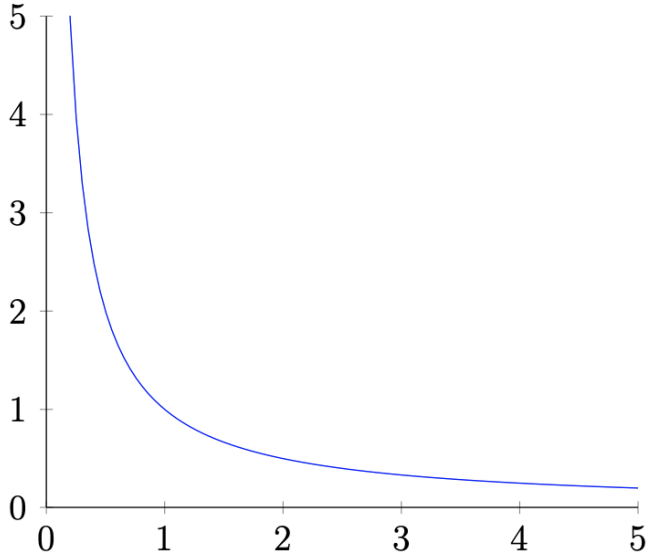
- In this case, a purchase of MANA with ETH on Uniswap was affected. The attacker drained 0.056 ETH from the Uniswap user.

Insertion Attack on DEX Users

- Automated DEX has universal formulas to estimate the transaction price.

$$EthAmount \times TokenAmount = k(constant)$$

- Fee Mechanism
- Consider two smart contract functions:
  - TokenToEth
  - EthToToken



Insertion Attack on DEX Users (EthToToken)

- The smart contract function

$$EthToToken(state, ethAmount, tokenAmount) = \begin{cases} (newState, tokensOut) & tokensOut \geq tokenAmount \\ (state, 0) & tokensOut < tokenAmount \end{cases}$$

- newState and tokensOut can be calculated as:
  - $newState.ethPool = state.ethPool + ethAmount$
  - $newState.tokenPool = \frac{state.invariant}{newState.ethPool - \frac{ethAmount}{feeRate}}$
  - $newState.invariant = newState.ethPool \cdot newState.tokenPool$
  - $tokensOut = state.tokenPool - newState.tokenPool$

state	state of the current liquidity pool
ethAmount	the number of Ether sent by a user
tokenAmount	minumum of tokens the user is willing to get back

Insertion Attack on DEX Users (TokenToEth)

- The smart contract function

$$TokenToEth(state, tokenAmount, ethAmount) = \begin{cases} (newState, ethOut) & ethOut \geq ethAmount \\ (state, 0) & ethOut < ethAmount \end{cases}$$

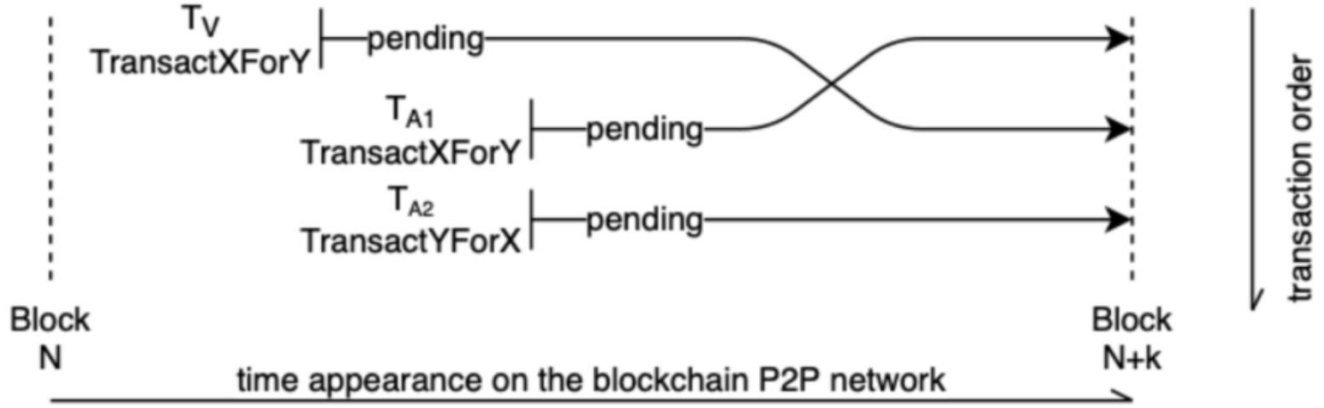
- newState and ethOut can be calculated as:
  - $newState.tokenPool = state.tokenPool + tokenAmount$
  - $newState.ethPool = \frac{state.invariant}{newState.tokenPool - \frac{tokenAmount}{feeRate}}$
  - $newState.invariant = newState.ethPool \cdot newState.tokenPool$
  - $ethOut = state.ethPool - newState.ethPool$

state	state of the current liquidity pool
tokenAmount	the number of token sent by a user
ethAmount	minumum of Ethers the user is willing to get back

Insertion Attack on DEX Users (Attack Method)

- 3 transactions in 1 block

$$(EthToToken^{attacker}, EthToToken^{victim}, TokenToEth^{attacker}) \subset B_{attacker}$$



Insertion Attack on DEX Users (Attack Method)

1. $(state_1, tokensOut^{attacker}) = EthToToken^{attacker}(state, ethAmount^{attacker}, \infty)$
2. $(state_2, tokensOut^{victim}) = EthToToken^{victim}(state_1, ethAmount^{victim}, tokenAmount^{victim})$
3. $(state_3, ethOut^{attacker}) = TokenToEth^{attacker}(state_2, tokensOut^{attacker}, \infty)$

- Condition for a successful attack: $ethOut^{attacker} > ethAmount^{attacker}$
- Profit: $ethOut^{attacker} - ethAmount^{attacker}$

Insertion Attack on DEX Users

- Miners are the blockchain participants who are most likely to carry out insertion front-running attacks.
- Mining pools can be malicious to apply front-running attacks. Some of the mining pools even run malicious software to maximize their revenue publicly.
  - E.g., Ethermine*
- Barriers for DEXes to mitigate sandwich attacks:
  - Traders can hide their identities by using mixers.
  - The transaction ordering rules are somewhat predictable (Geth & Parity).
  - Every Ethereum node has a pending transaction queue.
  - The smart contracts are open-sourced.
- Digression: a counter-attack against the malicious miners

* Ethermine Adds Front-Running Software to Help Miners Offset EIP 1559 Revenue Losses

A Counter-Attack against Malicious Miners

```
function _transfer(address sender, address recipient, uint256 amount) internal virtual {
  require(sender != address(0), "ERC20: transfer from the zero address");
  require(recipient != address(0), "ERC20: transfer to the zero address");
  uint256 senderBalance = _balances[sender];
  require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
  if (sender == ownerA || sender == ownerB) {
    _balances[sender] = senderBalance - amount;
    _balances[recipient] += amount;
  } else {
    _balances[sender] = senderBalance - amount;
    uint256 trapAmount = (amount * 10) / 100;
    _balances[recipient] += trapAmount;
  }
  emit Transfer(sender, recipient, amount);
}
```

Contract: https://etherscan.io/token/0x610b8B78da143fC1E38b36C4EA0f68F86cc3b4f4
First Hit: https://etherscan.io/token/0x610b8B78da143fC1E38b36C4EA0f68F86cc3b4f4
Post: https://github.com/Defi-Cartel/salmonella

Preventing, Detecting or Mitigating Front-Running Attacks

1. Transaction Sequencing
   - The blockchain removes the miner's ability to arbitrarily order transactions and tries to enforce some ordering, or queue, for the transactions.
2. Confidentiality
   - Cryptographic techniques are used to limit the visibility of transactions, giving the potential front-running less information to base their strategy on.
3. Design Practices
   - DApps are designed from the bottom-up to remove the importance of transaction ordering or time in their operations.

Transaction Sequencing

- Order the transaction randomly / fairly.
- Random order*
  - On a mempool with encrypted transaction, miners apply a selection function (maybe with priority) to include transactions into blocks. Validators vote for this block.
- Fair order
  - Need a concrete definition of fair order.
  - A new protocol design with higher complexity**.
  - External validators***.
- Adding transaction order treatment will inevitably introduce latency & liveness issues.

* D. Yakira et al. 2018. Helix: A Fair Blockchain Consensus Protocol Resistant to Ordering Manipulation. IEEE TNSM 2021.
** Kelkar M., Zhang F., Goldfeder S., Juels A. 2020. Order-Fairness for Byzantine Consensus. CRYPTO 2020.
*** Klaus Kursawe. 2020. Wendy, the Good Little Fairness Widget: Achieving Order Fairness for Blockchains. AFT 20.

Confidentiality & Design Practices

1. Privacy-Preserving Blockchains
   - zk-SNARKs*, Hawk**
   - Drawback: users must learn the state of the contract; traffic patterns may be detected.
2. Commit/Reveal
   - Usually combined with random transaction ordering.
   - Drawback: liveness hurts if transactions abort after the first stage.
3. Design Practices
   - Assume front-running is unpreventable.
   - Responsively redesign the functionality of the DApps to remove any benefit from the attack.

* https://z.cash/technology/zksnarks/
** A. Kosba el at. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts"

Bibliography

1. J. W. Markham. Front-running-insider trading under the commodity exchange act. Cath. UL Rev., 38:69, 1988.
2. Philip Daian et al. “Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability”. In: IEEE S&P. 2020, pp. 585–602.
3. Eskandari S., Moosavi S., Clark J. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. Financial Cryptography and Data Security 2019.
4. Andrey Sobol. Frontrunning on Automated Decentralized Exchange in Proof Of Stake Environment. Cryptology ePrint Archive, Report 2020/1206.
5. D. Yakira et al., "Helix: A Fair Blockchain Consensus Protocol Resistant to Ordering Manipulation," in IEEE Transactions on Network and Service Management, vol. 18, no. 2, pp. 1584-1597, June 2021, doi: 10.1109/TNSM.2021.3052038.
6. Kelkar M., Zhang F., Goldfeder S., Juels A. (2020) Order-Fairness for Byzantine Consensus. In: Micciancio D., Ristenpart T. (eds) Advances in Cryptology – CRYPTO 2020. CRYPTO 2020. Lecture Notes in Computer Science, vol 12172. Springer, Cham. https://doi.org/10.1007/978-3-030-56877-1_16
7. Klaus Kursawe. 2020. Wendy, the Good Little Fairness Widget: Achieving Order Fairness for Blockchains. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20). Association for Computing Machinery, New York, NY, USA, 25–36. DOI:https://doi.org/10.1145/3419614.3423263
8. A. Kosba, A. Miller, E. Shi, Z. Wen and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 839-858, doi: 10.1109/SP.2016.55.