DeFi Attacks — part 2

shymaa arafat

We ended part 1 with this figure from [1] describing several forms of MEV attacks, Fig.1

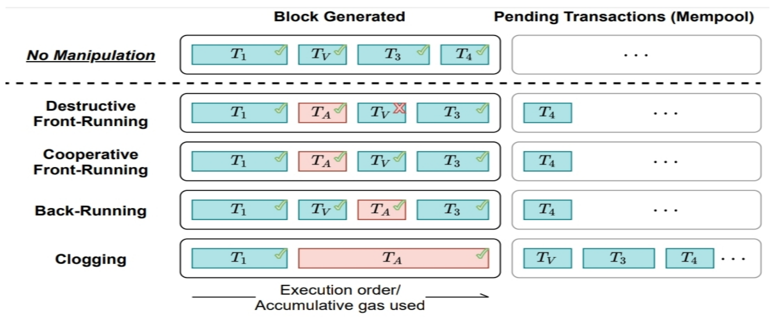

Block Generated
Pending Transactions (Mempool)
No Manipulation
T_1 T_V T_3 T_4
Destructive Front-Running
T_1 T_A T_V T_3 T_4
Cooperative Front-Running
T_1 T_A T_V T_3 T_4
Back-Running
T_1 T_V T_A T_3 T_4
Clogging
T_1 T_A T_V T_3 T_4
Execution order/ Accumulative gas used

Figure 1: Visualization of the four types of adversarial transaction ordering strategies. T_V is the victim and T_A the adversarial transaction. We assume that T_1 to T_4, are included in the next block in their sequence.

Fig.1: different kinds of front/ back running attacks, from [1]

Let's follow it with Fig.2 from [2] describing sandwich attacks first, then start to explain

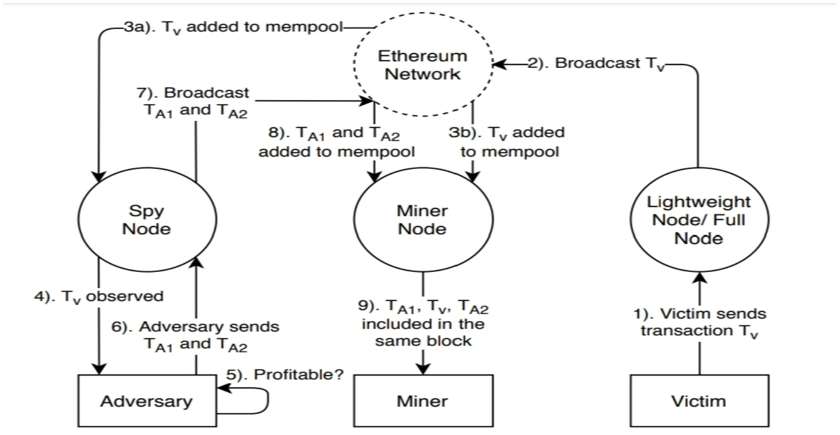

3a). T_v added to mempool
Ethereum Network
2). Broadcast T_v
7). Broadcast T_{A1} and T_{A2}
8). T_{A1} and T_{A2} added to mempool
3b). T_v added to mempool
Spy Node
Miner Node
Lightweight Node/ Full Node
4). T_v observed
6). Adversary sends T_{A1} and T_{A2}
9). T_{A1}, T_v, T_{A2} included in the same block
1). Victim sends transaction T_v
5). Profitable?
Adversary
Miner
Victim

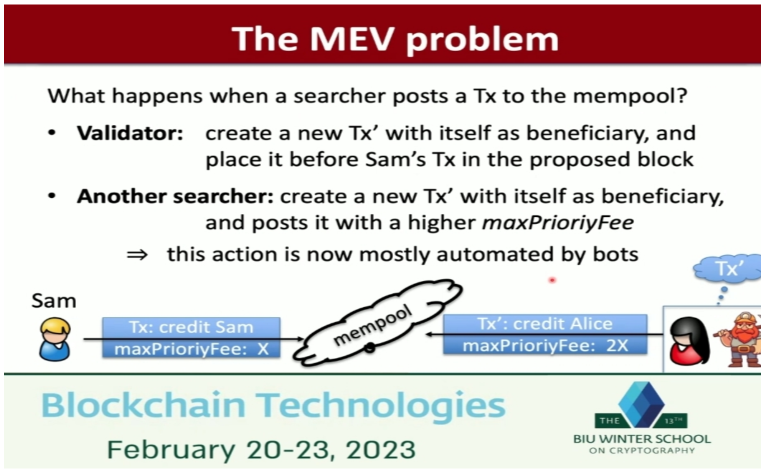
Fig. 2: Sandwich attack system.

Fig2: From [2] describes the steps of a sandwich attack using a spy node like that we decribed in part-1 under Network Layer attacks

In both cases whether it's the miner, an adversary spy node, or an MEV Bot, the idea (as defined by) is to use the information in a profitable way. Let's get back to our example of the victim trying to swap an amount of X to an equivalent amount of Y in an AMM (let's follow the figs V for victim& A for adversary, instead of Alice & Mallory) :

-V could be just a person who needs certain amount of currency Y; in this case A would back-run buying X with its now lower price after T(A) is executed. A may also front-run if deliberately trying to harm V by performing a swap that either makes X cheaper before A sells it, or Y more expensive before A buys it.

-If V is catching an arbitrage opportunity in this DEX; ie, either X is higher than other DEXs or Y is cheaper. Then, A will try to front-run V; ie, put its TX in the pool with higher fee to catch the arbitrage opportunity. Note that arbitrage opportunities usually involves multiple swaps and “V” probably got them through hardworking a Bellman-Ford shortest path graph algorithm or by modeling the system state then applying some heuristics [3,4], and maybe there are more techniques; now “A” will simply steal the result of all this effort by a simple front-run, Fig.3 from [5]. If V is an arbitrage Bot and A is an MEV Bot they will keep raising the fee over each other in a gas auction causing network congestion, higher stale block rate, and higher gas prices for everyone, the so called gas wars.



The MEV problem

What happens when a searcher posts a Tx to the mempool?

- **Validator:** create a new Tx’ with itself as beneficiary, and place it before Sam’s Tx in the proposed block
- **Another searcher:** create a new Tx’ with itself as beneficiary, and posts it with a higher *maxPrioriyFee*

⇒ this action is now mostly automated by bots

Sam
Tx: credit Sam
maxPrioriyFee: X
mempool
Tx’: credit Alice
maxPrioriyFee: 2X
Tx’

Blockchain Technologies
February 20-23, 2023
THE 13TH BIU WINTER SCHOOL ON CRYPTOGRAPHY

Fig.3: how the effort in searching for profitable TXs can be stolen through front-running

I think this attack as just described could also apply to Bitcoin with the RBF in place of gas auctions.

-If V is performing the arbitrage through multiple TXs; in this case A may put his corresponding sequence of copied TXs before, after, and maybe in-between A’s TXs which we call sandwich attacks, but the same argument and methodology still applies.

-The same also applies if V is trying to buy a just liquidated collateral, or even worse V is trying to pay the loan before his collateral gets liquidated, and A tries to precede V; maybe in this case A will try to perform more than just front-running, A may try to delay or time out V’s TX by filling the whole block, or even more than one block with dummy computation; what is called *clogging*.

Clogging:

We can define clogging as a malicious attempt to consume block space to prevent the timely inclusion of other transactions[1]; sometimes called *Supression* or *DoS* due to block stuffing, and is included with code examples in [6] as SWCR-128. Of course the attack is considered profitable only if the paid gas to fill block or more with higher gas fee TXs is less than the expected revenue from preventing or delaying the victimized TX; however, it turned out there are some tricks,[1] referring to [7,8], A can buy time with less gas. By minting gas tokens A can consume block space then later recover a fraction (~50%) of the gas cost he paid by re-freeing the occupied storage later. Incidents of such attacks are frequent in auctions; we mentioned in part-1 the famous example of the Fomo3d game 2018.

Transaction Replay attacks:

Some times the adversary can't detect protifable TXs, if they were encrypted using commit-and-reveal service for example, then A can still run V's as a black box in an emulator and see if it's profitable or not and copy it if profitable. This attack is much easier for miners since they can put their TX on top of the block and thus be sure of the system state at time of running (maybe there are dependant parameters). The authors in [1] included real incidents of replaying the famous bZx attack that we will explain shortly, and *Eminence* exploit¹; Fig.4 is also from [1].

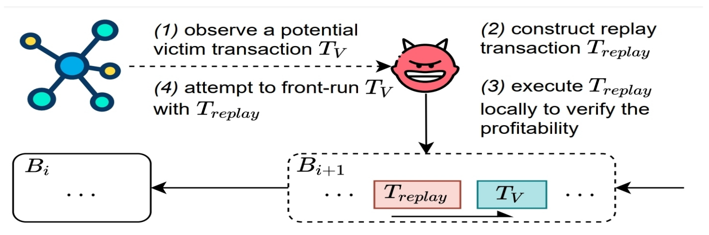

(1) observe a potential victim transaction T_V
(2) construct replay transaction T_{replay}
(4) attempt to front-run T_V with T_{replay}
(3) execute T_{replay} locally to verify the profitability
B_i
B_{i+1}
T_{replay}
T_V

Figure 8: Overview of the transaction replay attack.

Fig.4: Transaction Replay attack from [1]

Privately Minted Transactions/ off-chain agreements:

Generally speaking for all Blockchains, the authors in [1] describe by this the cases when you find low gas fee TXs placed earlier in the block obviously because there have been some kind of private off-chain agreement between the miner and some users; this is considered colluding because it violates the equal opportunity principle, and also doesn't give "V" the chance to submit a higher bid (TX fee) than than the secret value A paid, maybe A is willing to pay more. However, this paper was written before the deployment of EIP-1559 which nearly solved the problem by making miner-tips submitted on-chain along with a fee-cap for maximum total payment [9]. I have no clue if there exist attack incidents on other Blockchains, but the pools mentioned in the paper providing the possibility of such TXs, 1inch & SparkPool, have removed the links [10] I followed and stored 2 years ago and any mentioning of private TXs now refer to encrypted or confidential transactions.

Flashloans & Oracle Manipulation Attacks

This paper [11] about flashloan attacks, the famous 2020 ***bZx*** attack[2], the exploits that caused it, how could it have been optimized to more profit, replay versions imitating it[1],… have been extensively discussed and analyzed everywhere [12,13,14]. The simple problem is that Blockchains are not actually connected to the internet or anything outside the Blockchain; when it needs to know something whether it is the weather condition, a result of a match people are bidding on, or a price of a currency, it does so by sending a query to a trusted source; ie, an oracle. Like any source oracles could be malicious, hence [15] it usually asks more than one source and seeks some kind of majority consensus between them. Incase of bZx kyber asked 2 resources that was both manipulated; maybe at the end it is still could be viewed as a protocol exploit. It’s worth mentioning that there was another ***pump-and-dump*** arbitrage attack by bZx, but I think this one is best classified as market manipulation attack.

Cross-chain & Bridge Attacks

At the beginning most DeFi activities and applications were on the Ethereum Blockchain as the poineer of smart contracts invention, consequently most attacks happened there and most research focused there too. Now there are plenty of EVM like Blockchains that have some protocol differences, hence there are some variations in attacks & mitigations. However, the more complicated problems come from cross-chains activities, whether from adversaries performing their attack as a sequence of steps on multiple chains (instead of just multiple DEXs before for example), or from possible exploits in the applications helping DeFi users interact across chains; there 2 major designs Bridges & Omni chains. I don’t think I can give the reader you a deep dive into this topic, my simple knowledge is:

- There were a statement by Vitalik Buterin in Jan 2022;

"there are fundamental security limits of bridges"

-There were 2 reported attacks [16] on Bridges Feb 2022 and Aug 2021; both teams say the code vulnerabilities were successfully handled[3].

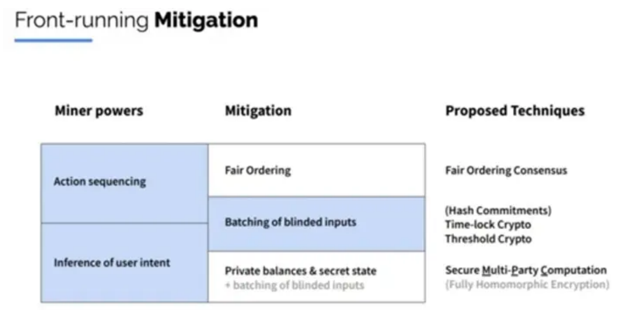
-The ***a16z*** crypto group and Uniswap have different opinions you could say about the security risks of Bridges Vs Omni chains [17].

I guess this is enough overview of the widely known attack types, I refer the interested reader to the valuable SoK paper [14], the 10 authors put a lot of effort to consolidate all known attack kinds, they also analyzed and compared all the available academic literature as opposed to statistics and data analysis of reported attack incidents

Mitigations

There have been some contraverserial arguments about MEV in the crypto community; some talks about accepting MEV and maybe smoothing it, and there are many MEV Bots out there in the loud like Ethermine for example [18]. In any case, the research and the code development goes hand in hand for both cases; here's an overview of most well known mitigations[4].

-We first mentioned Commit-then-reveal in part-1, however it introduces some problems [19]; the delay for encryption/decryption and for having a limited choice of miners accepting your TX, all this in such a time critical competition (don't forget you're originally doing this to not be front-run!), also how to put a time out for the TX. Then there have been proposed different ordering techniques [20,21,22,5], Fig.5 from [19] summarizes them.



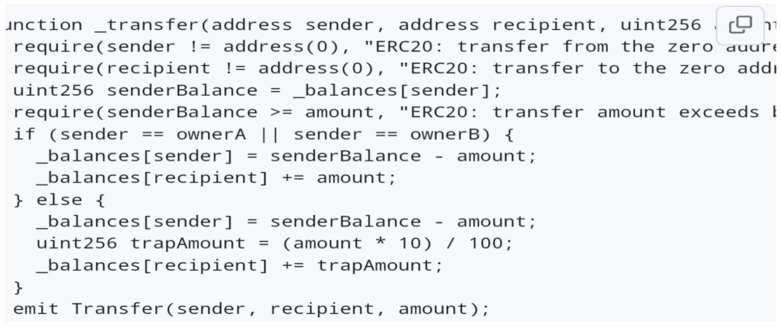
Front-running **Mitigation**

Miner powers	Mitigation	Proposed Techniques
Action sequencing	Fair Ordering	Fair Ordering Consensus
	Batching of blinded inputs	(Hash Commitments) Time-lock Crypto Threshold Crypto
Inference of user intent	Private balances & secret state + batching of blinded inputs	Secure Multi-Party Computation (Fully Homomorphic Encryption)

Fig.5 front running mitigations from [18]

-Heuristics and AI techniques is an important approach that have been used to identify behavior patterns for many kinds of attacks not just MEV; infact most Rug-Pull protections can be classified as so. For MEV, [1,2,14] include heuristics for every kind of attacks; example heuristics used in [1] to identify sandwich MEV attacks is that either the same user address sends transactions TA 1 & TA 2, or two different user addresses send TA 1 & TA 2 to the same smart contract, also that the amount of asset sold in TA 2 must be within 90% ~110% of the amount bought in TA 1. Practically identifying heuristics are used by Bots in both directions in an ongoing battle; when the team in [23] formalized profit as an optimization problem and compared calculated optimal profit to that produced by platforms promising users optimal revenue, they found profits differ by up to 418.14 ETH (~517K$) in some cases. Those suboptimal trades were sometimes immediately followed by ***back-running TXs*** (heavily tied to specific miners) ***extracting the missed profit***, in short, they hypothesize ***those miners used insider information to gain profit*** (maybe the original platforms deliberately did not optimize the profit as promised colluding with those miners).

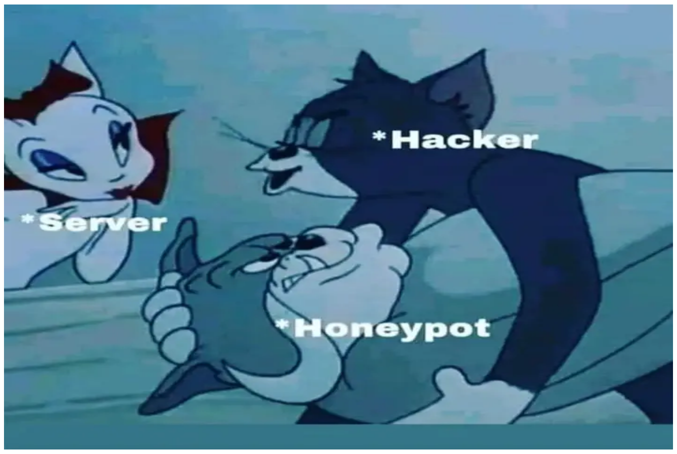
-An interesting case of deceiving Ethermine MEV Bot to lose 250K$ what was called ***Salmonella Bad Sandwich*** [24]; the trader code returns only 10% if someone else is transacting his smart contract. Hacker News site [25] gives more sophisticated analysis discussing how this contract could steal others money; maybe this could be classified under practical smart contract security, Fig.5; the approach is also similar to the idea of authentication enabled contracts (revert execution, with consumed gas lost of course, if invoked with an unauthorized address) described in [1] as a mitigation for replay attacks.



```
unction _transfer(address sender, address recipient, uint256 
 require(sender != address(0), "ERC20: transfer from the zero addre
 require(recipient != address(0), "ERC20: transfer to the zero addr
 uint256 senderBalance = _balances[sender];
 require(senderBalance >= amount, "ERC20: transfer amount exceeds 
 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);
```

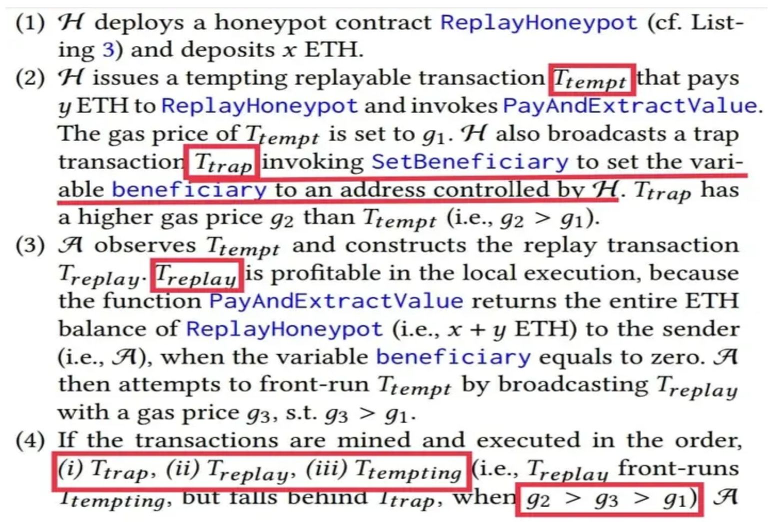
Fig.5: the poisonous transfer function of the Salmonella contract from (https://github.com/Defi-Cartel/salmonella)

-Another trick that is used to protect from MEVs is Honeypots; an old long used cyber security term for deceiving an attacker into a trap.



A fun picture describing Honeypots as used in cyber security

[1] suggest the use of Honeypots as a protection from replay attacks, like the poisoned sandwich we have just explained, the adversary is trapped into replaying a TX that when executed by him is less profitable than the gas costs he had to spend, Fig6.



(1) $\mathcal{H}$ deploys a honeypot contract ReplayHoneypot (cf. Listing 3) and deposits x ETH.

(2) $\mathcal{H}$ issues a tempting replayable transaction T_{tempt} that pays y ETH to ReplayHoneypot and invokes PayAndExtractValue. The gas price of T_{tempt} is set to g_1. $\mathcal{H}$ also broadcasts a trap transaction T_{trap} invoking SetBeneficiary to set the variable beneficiary to an address controlled by $\mathcal{H}$. T_{trap} has a higher gas price g_2 than T_{tempt} (i.e., $g_2 > g_1$).

(3) $\mathcal{A}$ observes T_{tempt} and constructs the replay transaction T_{replay}. T_{replay} is profitable in the local execution, because the function PayAndExtractValue returns the entire ETH balance of ReplayHoneypot (i.e., $x + y$ ETH) to the sender (i.e., $\mathcal{A}$), when the variable beneficiary equals to zero. $\mathcal{A}$ then attempts to front-run T_{tempt} by broadcasting T_{replay} with a gas price g_3, s.t. $g_3 > g_1$.

(4) If the transactions are mined and executed in the order, (i) T_{trap}, (ii) T_{replay}, (iii) $T_{tempting}$ (i.e., T_{replay} front-runs $T_{tempting}$, but falls behind T_{trap}, when $g_2 > g_3 > g_1$) $\mathcal{A}$

Fig.6: The main steps of a deploying Honeypot protection from replay attacks (an annotated screenshot from [1] Appindex)

Footnotes

1-The Eminence exploit was a kind of pump-and-dump attack that happened in the last days of Sep 2020, some of the drained money was returned (https://www.coindesk.com/tech/2020/10/01/defi-degens-hit-hard-by-eminence-exploit-will-be-partially-compensated/). Another interesting example of a newer flashloan attack, last May, of 7.5m$ on the Jimbo protocol exploiting a vulnerability in their shift fn is explained here: https://www.coindesk.com/tech/2023/05/31/jimbos-protocol-to-work-with-us-homeland-security-to-help-recover-75m-from-flash-loan-exploit/, & https://medium.com/p/45c7e2c9c00

2-Another interesting example of a newer flashloan attack, last May, of 7.5m$ on the Jimbo protocol exploiting a vulnerability in their slippage control is explained here (https://www.bleepingcomputer.com/news/security/flash-loan-attack-on-jimbos-protocol-steals-over-75-million/) which found this the most explainatory. Other sources include: (https://www.theregister.com/2023/05/30/jimbos_protocol_defi_attack/), (https://www.coindesk.com/tech/2023/05/31/jimbos-protocol-to-work-with-us-homeland-security-to-help-recover-75m-from-flash-loan-exploit/); where I first heard about it: (https://medium.com/p/45c7e2c9c00)

3-A more sophisticated link that contains code exploit details for the interested reader is (https://neptunemutual.com/blog/decoding-omni-bridges-call-data-replay-exploit/); another paper that I didn't know about at the time of first writing is (https://rdi.berkeley.edu/zkp/zkBridge/zkBridge.html)

4-There were a mention in [5] for using specific hardware-based component that enables secure execution of a pre-defned code (enclave) in an isolated environment; ie, to provide a *Trusted*

Execution Environment ***TEE***, ie, to order TXs without any MEV, but we consider this approach beyond the scope of this article. The interested reader may check (https://decentralizedthoughts.github.io/2023-04-09-blockchainsplustees-day1-summary/, https://writings.flashbots.net/block-building-inside-sgx)

References

[1] Arthur Gervais, Liyi Zhou, et al, “Quantifying Blockchain Extractable Value: How dark is the forest?”, Jan2021.

https://arxiv.org/abs/2101.05511

[2] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais,

"High-Frequency Trading on

Decentralized On-Chain Exchanges", 29Sep2020, arXiv:2009.14021v1

[3] Robert Andrew Martin, "Graph algorithms and currency arbitrage" parts 1&2, and GitHub code, Mar 2019; https://reasonabledeviations.com/2019/03/02/currency-arbitrage-graphs/, https://github.com/robertmartin8/CryptoGraphArb/tree/master

last accessed 24/6/2023.

[4] Liyi Zhou, Kaihua Qin, Antoine Cully,

Benjamin Livshits, and Arthur Gervais,

"On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols", Mar 2021; https://arxiv.org/abs/2103.02228, last accessed 24/6/2023.

[5] Dan Boneh & Valeria Nikolaenko, “MEV & Fair Ordering" , Blockchain Technologies, BIU winter school on Cryptography, Feb 2023; https://youtu.be/T1bD7_OTD1o

[6]”DoS With Block Gas Limit";https://swcregistry.io/docs/SWC-128, last accessed 24/6/2023.

[7] Gastoken.io., Gas Tokens, https://gastoken.io/; cross reference.

[8] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction

ledger". Ethereum project yellow paper, 151(2014):1–32, 2014; cross reference.

[9] Tim Roughgarden, "An Economic Analysis of EIP-1559" Q&A with Vitalik Buterin, Silicon Valley Ethereum, Dec 2020; https://youtu.be/ndNyx-Oj9Wk

[10] https://help.1inch.io/en/articles/4695716-what-are-private-transactions-and-how-they-work,

last accessed 17/3/2021 (removed now)

[11] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais,"Attacking the DeFi EcoSystems with Flashloans for Fun and Profit", Fc21, Ifca; https://fc21.ifca.ai/papers/158.pdf; https://youtu.be/E_I-RDc2TE4; Open Blockchain Workshop OBWS , https://youtu.be/D96W0CC8TDI; TPBC 2021, https://youtu.be/nd0npdNQTyI

[12]Arthur Gervais, "DeFi Flashloan Attacks", Berkeley DeFi MOOC 2021, lec:13.6, https://youtu.be/NifGKCiiX3E

[13] Chainlink, "Price Manipulation Attacks from First Principle", Fall 2021 Hakathon; https://www.youtube.com/live/nrnC2Z-ZZow

[14] Liyi Zhou et al, "SoK: Decentralized Finance (DeFi) Attacks", arXiv:2208.13035v3, April 2023; https://youtu.be/85V2amqUOp0

[15] Ari Juels, "Oracles", Berkeley DeFi MOOC 2022, lec:8; https://youtu.be/vFcW18ZpPZ4

[16]https://cointelegraph.com/news/wormhole-token-bridge-loses-321m-in-largest-hack-so-far-in-2022, last accessed 25/6/2023.

[17] https://cointelegraph.com/news/a16z-votes-against-proposal-to-deploy-uniswap-v3-on-bnb-chain, last accessed 10/6/2023.

[18] William Foxley, "Ethermine Adds Front-Running Software to Help Miners Offset EIP 1559 Revenue Losses",

https://www.coindesk.com/markets/2021/03/17/ethermine-adds-front-running-software-to-help-miners-offset-eip-1559-revenue-losses/, last accessed 24/6/2023

[19] Carsten Baum et al , "SoK: Mitigation of Front-running in Decentralized Finance", DeFi22; https://youtu.be/sTztAajadfI

[20] Dahlia Malkhi, "Maximal Extractable Value (MEV) Protection on a DAG", CESC 2022; https://youtu.be/Wi7yK-4A_i0

[21] Ari Juels, "Fair Transaction Ordering: why and how", TSE Lab; https://youtu.be/KIO3pHLD0hA

[22] Massimo Bartolleti et al, "Maximizing Extractable Value from Automated Market Makers", FC22, ifca; https://youtu.be/rAxIuAP_6mw

[23] Aviv Yaish, Maya Dotan, Kaihua Qin, Aviv Zohar, Arthur Gervais, "Suboptimality in DeFi"; https://www.researchgate.net/publication/371409361_Suboptimality_in_DeFi

[24] William Foxley, "Bad Sandwich: DeFi Trader 'Poisons' Front-Running Miners for $250K Profit"; https://www.coindesk.com/tech/2021/03/22/bad-sandwich-defi-trader-poisons-front-

running-miners-for-250k-profit/, last accessed 24/6/2023.

[25] Hackers News, "Wrecking sandwich traders for fun and profit";https://news.ycombinator.com/item?id=26514624, last accessed 25/6/2023