Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

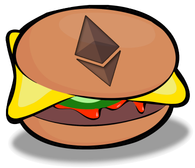
Analyzing and Preventing Sandwich Attacks in Ethereum

Bachelor's Thesis

Patrick Züst

zuestp@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich



Supervisors:
Tejaswi Nadahalli, Ye Wang
Prof. Dr. Roger Wattenhofer

August 2, 2021

Acknowledgements

I would like to thank Tejaswi Nadahalli and Ye Wang for supervising my thesis and for their continuous support throughout the semester. Prof. Dr. Wattenhofer enabled this work by taking a chance on an unproven idea. The first proposal for this thesis was drafted in collaboration with representatives of Bitcoin Suisse AG.

This work was inspired and supported by various individuals of the Ethereum community. Mainly I want to thank Kartik Talwar and Liam Horne, founders of ETH Global, for sharing their insights and their contacts. Many thanks also to the researchers involved in the Flashbots project who are paving the way for advanced MEV studies and provided feedback on this work as well.

Abstract

Ethereum uses a permissionless blockchain to enable applications without central intermediaries. However, the advent of decentralized finance (DeFi) has led to various new attacks which are being launched on a global scale. Bots continuously scan pending transactions and employ different tactics to profitably frontrun them. A common type of frontrunning is the so-called sandwich attack. In this work, we present a large-scale analysis of sandwich attacks for a time period of twelve months. We found that during this time there were at least 480'276 attacks leading to an accumulated profit of 64'217 ETH (189'311'716 USD). We also show that miners have recently begun to play a more active role in these value extractions which drastically changes the patterns we observe for sandwich attacks. Splitting up frontrunnable trades can be a valid mitigation strategy. We show how traders could have saved 30'525 ETH (89'987'700 USD) by releasing multiple smaller swaps instead of one large trade. A public tool to check whether a transaction is susceptible to sandwich attacks and to find a suitable order split was released on `www.DeFi-Sandwi.ch`.

Contents

Chapter 1

Introduction

Ethereum is a permissionless peer-to-peer network based on blockchain technology. Its support of smart contracts has led to the emergence of decentralized finance (DeFi), a set of finance-focused protocols that enable users to take out loans [1], or exchange tokens [2] without any intermediaries involved. As there is no strict legal framework surrounding these services, practices emerged that are prevented in the traditional financial system by regulation, but can happen freely on Ethereum. Specifically, traces of frontrunning can nowadays be found in most blocks that are added to the Ethereum blockchain. This was first described by Daian et al. in 2019 [3]. While DeFi started off as a democratization of finance, it has become a platform with considerable information asymmetries where a small number of players extract large values from unsuspecting victims. In this context, Ethereum is often compared to a *dark forest* [4] where bots continuously scour pending transactions and look for ways to attack them. In recent months, miners have started to act as predators themselves and began actively reordering transactions to extract additional value. These new developments were fueled by a project called *Flashbots* [5] which allows users to submit suggestions for block compositions directly to miners. While researchers are discussing whether miner extractable value (MEV) is boon or bane for the stability of the Ethereum network [6, 3], unsuspicious DeFi users lose millions of dollars every month because their transactions are getting frontrun. This problem will remain even after the Ethereum network switches to a proof of stake consensus mechanism.

Possible cryptography-based defenses and new implementations for DeFi infrastructure to mitigate this issue are being discussed [7], but far away from introduction or even mainstream adoption. Current solutions mostly revolve around private mempools where users pay a fee to submit transactions directly to a miner [8, 9]. This can however cost hundreds of dollars and drastically contravenes the idea of decentralization and trustlessness that Ethereum is based on.

In this work, we focus on so-called sandwich attacks happening on decentralized exchanges. We first analyze the strategies of attackers and show how DeFi users are impacted. We describe why the recent developments require a different analysis approach than chosen in related work, and present results that go far beyond what was known so far. We then show how users can prevent sandwich attacks by splitting vulnerable transactions into multiple trades. This mitigation strategy can be used on the current DeFi infrastructure and does not require any technical knowledge, personal contacts, or trust in miners. We describe the functionalities of `www.DeFi-Sandwi.ch` - a tool that informs traders whether their transaction is susceptible to sandwich attacks and suggests a suitable order split. Finally, we present the insights from a user perception survey we conducted among Ethereum users regarding sandwich attacks.

We hope that our work helps to enlighten the *dark forest* and combat the current information asymmetry in the DeFi world. Our publicly available tool should empower crypto enthusiasts to use existing DeFi services without the risk of falling victim to a frontrunning predator.

Chapter 2

Background

This chapter provides an overview of the important concepts related to our work.

2.1 Automated Market Maker (AMM)

A decentralized exchange (DEX) allows users to trade tokens without the need of a central intermediary. While exchanges in traditional financial markets are typically based on order books, most decentralized exchanges in Ethereum are implemented as automated market makers. These exchanges maintain liquidity pools of two assets and enable users to swap one for the other.

2.1.1 Constant Product AMM

The largest decentralized exchange in Ethereum, Uniswap, is built as a constant product AMM [2]: Users are able to swap tokens in the liquidity pool if the product of the two reserves, r_1 and r_2, is the same before and after the swap. Most AMMs also charge a fee for executing trades (currently 0.3% of the input amount on Uniswap). This fee stays in the asset pool and is eventually distributed to the liquidity providers as incentive to provide liquidity. If a user swaps an amount a on Uniswap, the output amount is hence calculated using the formula

$$\frac{a \cdot 0.997 \cdot r_2}{r_1 + 0.997 \cdot a}.$$

Since the reserves of a liquidity pool can change between the signing of a transaction and its execution, the users are not guaranteed to trade at a specific market price. They can therefore choose the maximum relative price increase they are willing to accept. In the Uniswap interface, this slippage rate is specified as a percentage value and then transformed to a minimum output amount. If the received output is smaller than the specified minimum, the swap is reverted. In this case, users keep their tokens but must pay the Ethereum gas fee.

Uniswap is currently the largest decentralized exchange in Ethereum with almost 2bn USD locked in liquidity pools and 1bn USD of daily trade volume on average [10]. Other popular exchanges like SushiSwap and PancakeSwap are forks of Uniswap and the described concepts also apply to them with some small adaptions.

2.2 Frontrunning

In Ethereum, a signed transaction is sent to miners who are incentivized to include it in the blockchain. It is not clear in advance who is going to mine the next block and a transaction is typically broadcasted to the whole network. Nodes store these pending transactions locally in the mempool. The time it takes for a transaction to be included in a block depends on the chosen gas price and the available block space.
Whoever has access to an Ethereum node can inspect transactions in the mempool. If a pending transaction T_V fulfills some criteria, an attacker can broadcast a new transaction T_{A1} with a slightly higher gas price. If a miner orders transactions strictly according to the gas price, the attacker's transaction T_{A1} will be executed before transaction T_V. Frontrunning has been studied in traditional markets for decades [11]. In Ethereum, it was for the first time extensively documented by Daian et al. in 2019 [3]. They observed that bots not only create new transactions based on mempool activity but also engage in gas price bidding wars with each other for preferable block placements.

2.2.1 Miner Extractable Value (MEV)

Several papers have described how frontrunning and backrunning can be used to generate a profit [12, 13, 14, 7, 3]. Examples range from replay attacks over arbitrage to sophisticated attack schemes related to collectibles and ICOs. All profits achieved through frontrunning attacks depend on the transaction order in a block. Since miners have full control over how they arrange transactions, profit through frontrunning is considered miner extractable value (MEV). This term refers to the value that miners can generate by adding new transactions to a block or favorably ordering pending transactions [3]. There is a current discussion between experts whether the existence of MEV is a risk for consensus layer security [3], or fundamental for keeping the Ethereum network decentralized [6].

2.2.2 Sandwich Attacks

As discussed in chapter 2.1.1, a swap on a decentralized exchange can lead to significant boosts of an asset's market price. Attackers continuously monitor the mempool to find a transaction T_V which entails large price differences. They then release a frontrunning transaction T_{A1} to buy the given asset, and a backrunning transaction T_{A2} to sell it for an increased price. The attacker's cost consists of the Ethereum transaction fees (dependent on the gas price), as well as the swap fee (0.3% on Uniswap).

The price increase is limited by the chosen slippage rate of T_V. If the market price increases too much before the victim transaction is executed, its slippage detection will be triggered and the transaction will fail. In this case, the attack will not be profitable. More formally: Let v be the amount of asset t_1 to be swapped in T_V, and m the minimum amount of asset t_2 to be received (determined by the expected output and the slippage tolerance). Let x be the amount of t_1 swapped in T_{A1}. We assume that no other swap took place in the pool since T_V was created and that r_1 and r_2 represent the original liquidity reserves in the pool. The victim transaction will succeed on Uniswap if

$$\frac{v \cdot 0.997 \cdot (r_2 - \frac{x \cdot 0.997 \cdot r_2}{r_1 + 0.997 \cdot x})}{(r_1 + x) + 0.997 \cdot v} \geq m.$$

We solve this for x with equality to find the maximum input amount for the frontrunning transaction T_{A1}, such that T_V does not fail. The result can be written in compact form as

$$maxInput_{A1}(r_1, r_2, v, m) = \frac{5.01505 \cdot 10^{-7} \cdot t}{\sqrt{m}} - 1.0015 r_1 - 0.4985 v$$

with

$$t = \sqrt{9000000 r_1^2 m + 3976036000000 r_1 r_2 v - 5964054000 r_1 m v + 988053892081 m v^2}.$$

The $maxInput_{A_1}$ leads to the largest possible price delta before and after the victim transaction. It is not necessarily the ideal input amount, as the swap fees could potentially exceed the generated revenue if very extreme values for slippage tolerance and reserve amounts are chosen. For all practical purposes, the calculations shown above will however represent the most profitable input amount for T_{A1}.

2.2.3 Flashbots

Miners play a central role in the process of MEV extraction, as the name implies. However, permissionless blockchain technology is built on the fundamental idea of decentralization. Flashbots is an independent organization that wants to create an ecosystem for MEV extraction [5]. Instead of miners having full control over transaction orders in blocks, they propose a system where bots ("searchers") can submit suggestions for transaction orderings ("bundles"). Miners then either choose one of the submitted bundles as transaction order or create their own. Searchers incentivize miners to include their bundle by sharing part of the profit with them. This turns MEV extraction from a central process into an open market where bots continuously search for the most lucrative way to assemble a block with pending mempool transactions. Miners have no obligation to consider Flashbots bundles. They can order transactions by descending gas prices, as it was done for years. But miner revenue has increased by 6.5% on average when the submitted bundles were considered, according to data published by Flashbots [15]. This offers a natural incentive for miners to become part of the network. The project was launched at the beginning of January 2021 and by mid-April 2021, more than 85% of the Ethereum hashrate was using Flashbots [15]. This drastically changes the way MEV is extracted and has a large impact on sandwich attacks, as we illustrate in chapter 3.2.2.

Chapter 3

Analyzing Sandwich Attacks

In the following, we describe how we scanned the Ethereum blockchain for sandwich attacks, and talk about the insights we got from analyzing them. Our analysis starts at block number 9'976'964 (May 1, 2020) and ends at block number 12'344'944 (April 30, 2021).

3.1 Methodology

We run our own Ethereum node to get access to the block history. A modified geth client is used to export all transaction receipts where a swap event was triggered by a smart contract of a decentralized exchange. After pre-processing these receipts, we scan our data for two transactions T_{A1} and T_{A2} that comprise a sandwich attack. The presence of a victim transaction T_V is optional. We apply the following heuristics to identify a sandwich attack:

1. T_{A1} and T_{A2} are included in the same block and in this order.
2. T_{A1} and T_{A2} have different transaction hashes.
3. T_{A1} and T_{A2} swap assets in the same liquidity pool, but in opposite directions. The input amount for the swap in T_{A2} is equal to the output amount of the swap in T_{A1}.
4. Every transaction T_{A2} is mapped to exactly one transaction T_{A1}.

A sandwich attack can be successful, even if T_{A1} and T_{A2} are placed in different blocks. However, attackers want T_{A1} and T_{A2} to be included in one block, as additional swaps in the same pool could endanger their profit. Heuristic 1 allows us to find a lower bound of all sandwich attacks.
Heuristic 2 is necessary because two swaps fulfilling the given heuristics could potentially be made in the same transaction. This would not constitute a valid sandwich attack.

Sandwich attacks can still be profitable, if the input amount of T_{A2} differs from the output amount of T_{A1} (and thus violates heuristic 3). However, as this heuristic is our primary way to detect sandwich attacks (see section 3.1.1), we exclude imperfect attacks in our analysis. A rational attacker is incentivized to sell back all tokens in T_{A2} which were received in T_{A1}, as the price for this asset might decrease in the future. Holding small quantities of different tokens is not desirable because moving them will require users to pay a fixed transaction fee.
In rare cases, we were able to find two identical backrunning transactions. To prevent double-counting revenues, we excluded these transactions from our analysis by applying heuristic 4.

3.1.1 Comparision to Related Work

We have seen similar heuristics in work published by Qin et al. [12], as well as Torres and Statee [14]. Our methodology however differs in some fundamental aspects: First, we do not require the presence of a victim transaction T_V. Two transactions fulfilling the criteria above are considered a sandwich attack (although not a profitable one), even if none of the surrounded transactions do any swaps in the same pool. We especially want to include sandwich attacks where a potential victim transaction failed to execute because it was attacked by multiple bots.

Both related publications have some additional requirements for T_{A1} and T_{A2}: In the analysis by Torres and Statee, the liquidity for the two swaps must be provided by the same address, while Qin et al. require that T_{A1} and T_{A2} are either signed by the same account or sent to the same smart contract. We found successful attacks where none of these heuristics apply. This can be traced back to increasingly sophisticated attack mechanisms and a growing network of proxy contracts. We thus match frontrunning and backrunning transactions only if the output amount of the former is equal to the input amount of the latter. In the mentioned related studies, the requirements for similarity were less strict and the two amounts were allowed to differ by 1% and 10% respectively.

Torres and Statee additionally require that the gas price of the victim transaction lies between the gas prices of T_{A1} and T_{A2}. This heuristic does not make sense in today's context, as we see a lot of attacks executed in collaboration with miners. These transactions typically have a a very low gas price (compare section 3.2.2) and do not fulfill the given criteria. While Qin et al. did not set any restrictions for gas prices, they limited their analysis to only 71 different tokens. Our analysis doesn't have any comparable restrictions.

3.2 Empirical Results

In the given period, we analyzed 2'367'980 blocks using the heuristics above. In total, we discovered 480'276 sandwich attacks. Most attack transactions (93.26%) in this dataset were sent to a proxy contract instead of a router offered by the exchange. This allows attackers to check whether the respective liquidity pools still hold the expected amount of tokens before executing the swap. If this is not the case and an attack would not be profitable, the swap can easily be cancelled. In total, we found 964 different proxy contracts that received at least one attack transaction. Attackers appear to switch their proxies frequently, as a contract is only in use for two weeks on average (90'913 blocks). The most active proxy contract (0x0000..0084) processed 51'475 of the attack transactions we discovered (5.36%). For the sandwich attacks we investigated, the destination address of the frontrunning and the backrunning transaction differed in 2.95% of the cases. Overall, the sandwich attacks we discovered made use of 5387 different ERC-20 tokens. In 94.77% of attacks, ETH served as input token for the frontrunning as well as the victim transaction. In 3.55% of attacks, the input token for the frontrunning transaction was one of four well-known stable coins (Tether, USDC, BUSD, or DAI).

3.2.1 Profitability

To make statements about the profitability of attacks, we focus on transactions where at least one of the two involved tokens is ETH (which is the case for 96.28% of attacks). This allows us to calculate gas prices directly and sum up the accumulated profit in Ether. We use the reserves in the liquidity pools before the attack to find the conversion rate between different tokens. The accumulated profit over time can be seen in 3.1. The profit started increasing rapidly in July 2020. This coincides with the total number of sandwich attacks we are observing. Although the number of attacks stayed high through fall 2020, the profitability of the individual attacks decreased, most probably because of increased competition. The surge of profit at the beginning of 2021 could be connected to attackers collaborating with miners which is explained in more detail in section 3.2.2. In total, attackers earned 64'217 ETH (189'311'716 USD) in the given timespan. This also includes unprofitable attacks which constitute 18.14% of all attacks. If unprofitable attacks were excluded, the profit would amass to 73'337 ETH (216'197'476 USD).

To estimate the effectiveness of the attacks, we investigated the subset of profitable attacks where there was exactly one victim transaction, and where this victim transaction was sent to the router of either Uniswap V2 or Sushiswap. For this selection (226'905 attacks), it was possible to compute the slippage tolerance set by users. These calculations are based on the assumption that the

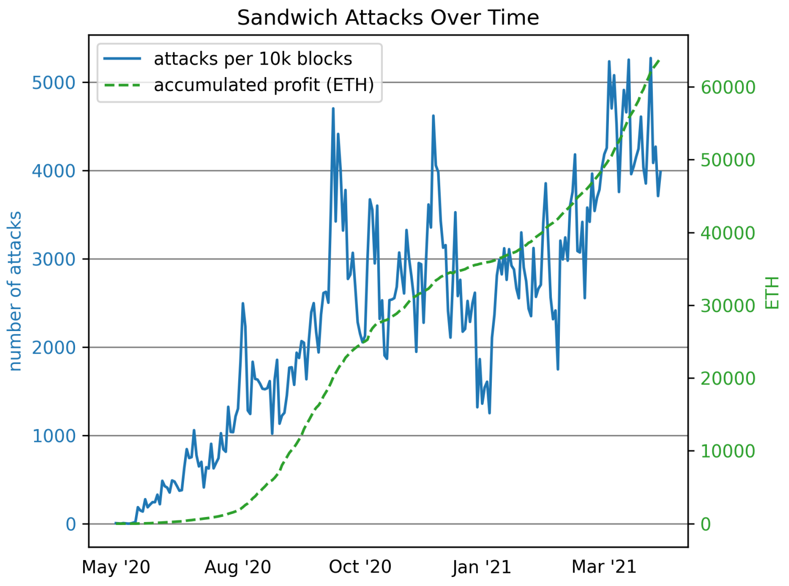


Figure 3.1: The number of sandwich attacks we observed and the profit attackers accumulate over time.

reserves in the liquidity pools were the same when the transaction was created and before the first attack transaction was executed. Figure 3.2 shows how the selected slippage rates are distributed. Especially the share of transactions with a slippage tolerance of more than 10% is notable. The official Uniswap V2 interface suggests values between 0.1% and 1%. Our analysis showed that attackers almost always achieve the maximum possible profit, i.e. they choose an ideal input amount for the buy transaction and push the price to its limit. The minimum output and the actual output of the victim transaction differed by less than 1% on average. This also includes all cases where an attacker could not achieve the maximum profit due to a lack of funds.

The most profitable attack with a single victim transaction occurred on Feb 17, 2021 on Uniswap V2: In this transaction (0xea1f..7a1c), a trader swapped 304 ETH for Union Protocol Governance Token (UNN). The allowed price slippage was set to a staggering 13%. A known sandwich bot released the respective frontrunning (0x87b5..66d6) and backrunning (0xf493..4054) transactions. They used 140 Ether in the frontrunning transaction and received 179.17 Ether from the backrunning transaction. The bot thus netted a profit of 39.17 ETH (100'626 USD) in a single attack.
The ten most unprofitable attacks all happened on Dec 19, 2020, and seem to stem from a misconfiguration of an attacker. They lost at least 219 ETH (645'612 USD) in less than 90 minutes. In the most unprofitable attack, the supposed vic-

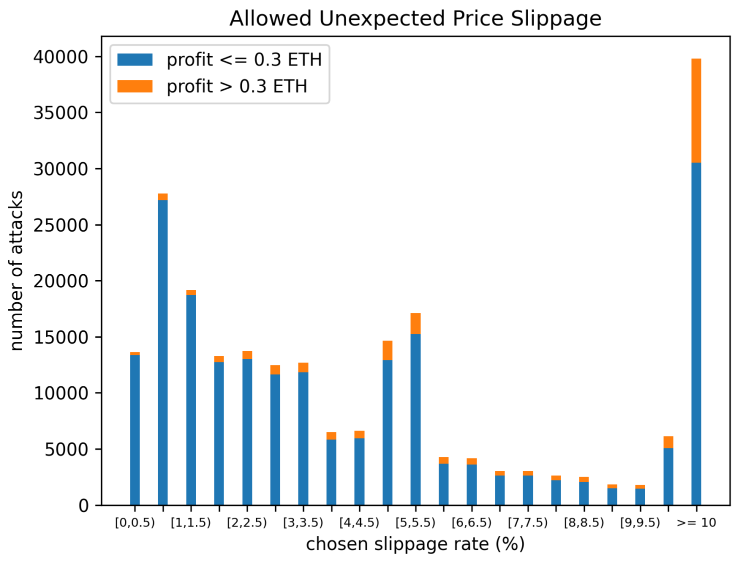


Figure 3.2: Distribution of the chosen slippage rates for all profitable attacks where the victim was sent to a router contract.

tim swapped DFF for ETH. The attacker however swapped ETH for DFF in the frontrunning transaction (0x8ffb..2e02) and DFF for ETH in the backrunning transaction (0xa4c6..c192). This inversion of tokens led to a degraded market price and a loss of 23.67 ETH (69'779 USD).
Unprofitable attacks can often be traced back to an unfavorable transaction ordering or to a failed victim transactions. The latter can happen if multiple bots attack a victim transaction which pushes the price over the accepted slippage tolerance. In 37.62% of unprofitable attacks, there was no successful swap in the respective pool between T_{A1} and T_{A2}. There have also been several attacks on sandwich bots that extract value from them by adjusting the transfer behavior of tokens [16, 17]

3.2.2 Active Reordering by Miners

Miners have full control over the transaction order in a given block. They can thus decide to exclude transactions, or even launch attacks themselves. If a rational miner does not actively reorder transactions, the order is given by the descending gas prices. If miners include private transactions, they can choose an arbitrary gas price for these transactions. Qin et al. monitored the mempool and found several examples of miners trying to disguise private transactions by setting a realistic gas price [12]. However, typically miners choose a gas price of 1 Gwei or less for private transactions.

We consider a sandwich attack to be in collaboration with a miner if the gas prices of the frontrunning and backrunning transactions are both at most 1 Gwei. There was a surge of such attacks since the beginning of 2021. This timing coincides with the release of the Flashbots project (see chapter 2.2.3). Instead of broadcasting transactions publicly and hoping that they will be included in the right order, attackers can use this tool to get more control over the block in which their attack should take place.

Table 3.1 shows the observed changes ever since miners started actively reordering transactions to extract additional value. If attackers find a susceptible victim

Property	Nov	Dec	Jan	Feb	Mar	Apr
Total Attacks	52K	60K	48K	51K	76K	84K
Gas Price $\leq$ 1 Gwei	0%	0%	5%	5%	6%	36%
Average Distance T_{A1}, T_{A2}	39.6	37.9	33.7	33.5	31.8	13.9
One Victim Transaction	83%	84%	86%	87%	90%	97%
Profitable Attacks	78%	76%	67%	80%	84%	92%

Table 3.1: Implications of active reordering by miners

transaction, they typically include T_{A1}, T_V, and T_{A2} in three contiguous block positions. This leads to a relative increase of attacks where there is exactly one swap transaction between T_{A1} and T_{A2}. It also leads to an increase in profitability.

It is to be expected that in the future most – if not all – sandwich attacks are executed in collaboration with miners. With a system such as *Flashbots* in place, the block positions of publicly broadcast attack transactions become somewhat arbitrary. Attackers even face the risk of miners purposefully ordering their publicly broadcasted transactions unfavorably. Considering the rapid increase in DeFi activity and the associated surge of MEV opportunities, miners are incentivized to actively reorder the transactions in their blocks. Daian [6] even argues that this process is fundamental to keep the Ethereum network secure in the future.

3.2.3 Unused Opportunities For Sandwich Attacks

Whether a sandwich attack generates positive revenue depends on the slippage tolerance and the input amount of the victim transaction, as well as the sizes of the respective liquidity pools. The details for this calculation were highlighted in chapter 2.2.2. To determine the hit rate of attackers, we investigated all swaps that occurred in the given timeframe and checked whether they could have been sandwich attacked. As we need to determine the slippage tolerance set by users, we again focus on transactions that were sent directly to the Uniswap V2 or the Sushiswap router. There were 17'644'672 such swaps in the given time frame.

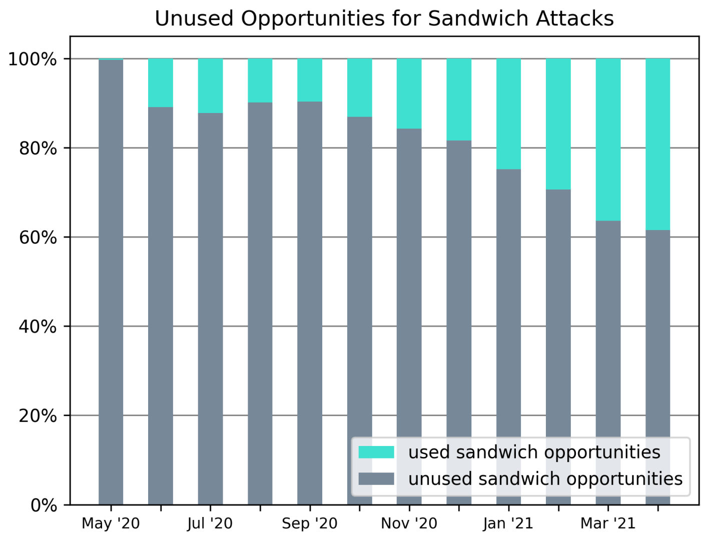


Figure 3.3: Ratio of profitable sandwich opportunities that were used by attackers, considering only transactions sent to a router contract with ETH as input token.

Most bots however only execute attacks where ETH serves as input token, so we focus our analysis on the 9'003'759 swaps where this was the case. The cost of an attack depends on the gas price and the gas used. While miners do not have to pay for gas if they execute an attack themselves, they forgo income they would have received by including other transactions. In the following, we consider a swap a missed sandwich attack opportunity, if the potential revenue is larger than the gas cost it would take to attack it.
In total we found 3'612'343 swaps with ETH as input token that could have been profitably attacked. Figure 3.3 shows how the share of unused sandwich opportunities declined over time. A similar trend can be seen when only considering swaps that would have returned more than 1 ETH in revenue. The largest missed opportunity for a sandwich attack was transaction (0x71c9..979b) where 724 ETH (2'134'352 USD) could have been extracted with a successful sandwich attack.

There are several reasons why opportunities for sandwich attacks remain unused: If a frontrunnable transaction only spends a few seconds in the mempool, bots do not have enough time to generate and transmit the respective attack transactions. Some frontrunnable transactions are never broadcasted publicly and instead created by the miners or sent to them using a private channel [5, 8]. Opportunities for sandwich attacks can also remain unused because of unfavorable transaction orderings or a lack of funds on the side of the attacker. Most notably, attackers today almost exclusively focus on swaps with ETH as input token. We found that 52.35% of attackable swaps have a different token as input. An attacker who does not hold the respective token would need to do at least one more swap which would reduce the profitability of the attack.

Chapter 4

Order Splits as Mitigation Strategy

A sandwich attack is only possible if the difference in market price before and after the swap is large enough to compensate for transaction and exchange fees. One basic mitigation strategy is hence to split up an order into multiple smaller parts. These split orders each lead to a smaller change in price. This approach has been explored in traditional financial markets for multiple decades [18]. In this chapter, we examine the application of order splitting in the context of decentralized exchanges and show how the strategy can prevent large losses for traders.

4.1 Theory

In an ideal order split, the input amount and slippage tolerance of a swap must be chosen such that there is no way to generate a profitable sandwich attack. We assume that only transactions of one trader and one attacker are being broadcast. Additionally, we assume that, if a large swap is split into multiple sequential transactions, each of these is included in the blockchain before the next one is being broadcast. The number of split orders must be kept as low as possible – not only because of the constant transaction fee which needs to be paid for every transaction, but also because the exchange fee stays in the pool after every swap and changes the market price for the worse.

We now illustrate how to find the ideal input amounts for the split orders. The following calculations are based on the notions we introduced in chapter 2.2.2. There we illustrated how the ideal input of an attacker, $maxInput_{A_1}$, is calculated based on the input and minimum output amount of a victim transaction, v and m, as well as the sizes of the liquidity pools, r_1 and r_2. The reserves and the accepted slippage rate are now considered given. The minimum output amount m thus solely depends on v and will be written as $minOutput_v(v)$. If the attacker executes and ideal attack, the output amounts of the two attack transactions for victim input v are calculated as follows:

$$output_{A_1}(v) = \frac{maxInput_{A_1}(v) \cdot 0.997 \cdot r_2}{r_1 + 0.997 \cdot maxInput_{A_1}(v)}$$

$$output_{A_2}(v) = \frac{output_{A_1}(v) \cdot 0.997 \cdot (r_1 + maxInput_{A_1}(v) + v)}{(r_2 - output_{A_1}(v) - minOutput_v(v)) + 0.997 \cdot output_{A_1}(v)}$$

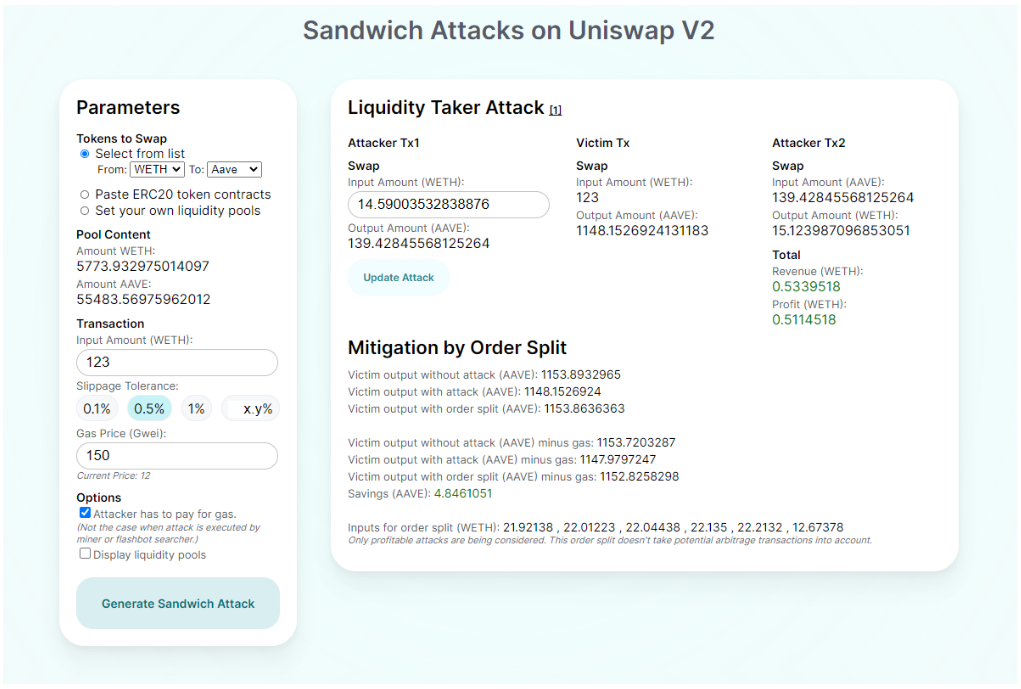
A sandwich attack can only be profitable if the inequality below holds. As all parts of the formula only depend on parameter v, a ternary search can be used to solve it with equality for v.

$$output_{A_2}(v) - maxInput_{A_1}(v) - transactionFees \geq 0$$

The transaction fees are calculated as a product of the gas price and the gas used. The latter depends on the number of executed computations during an attack and hence also on the sophistication of the proxy contracts. If the attack is executed by an independent bot, the gas price is usually very similar to the one of the victim transaction. If the attack is however launched in collaboration with a miner, the gas price can be set to 0. To hedge against all kinds of attacks, it makes sense to disregard transaction fees entirely when solving the equation for v.
The result, $maxInput_V$, constitutes the largest possible input amount, such that the swap cannot be exploited by a sandwich attack. The input amount of the first split order, v_1, is thus $max(v, maxInput_V)$. If $v_1 \leq v$, the same calculations is done again with updated reserve amounts r_1' and r_2'. This process needs to be repeated until $v_1 + v_2 + ...v_n = v$. With a simulation of these n split orders, it is then possible to determine whether the decreased token output and the increased transaction cost can compensate for the potential losses induced by a sandwich attack.

4.2 Backtesting

We tested whether order splitting is a valid mitigation strategy by checking if it could have prevented the sandwich attacks we discovered. To establish a lower bound it was assumed that attackers only pay exchange fees and no transaction fees. The slippage rate for the split orders was set equal to the original swap. In total, we considered 226'905 profitable sandwich attacks where the victim transaction was sent directly to the router of an exchange. The analysis showed that 160'347 (70.67%) of the given attacks could have been prevented if a suitable order split had been used. In these transactions alone, traders lost 42'504 ETH (125'301'792 USD). Applying the order split strategy would have saved them 30'525 ETH (89'987'700 USD). This is enough of a reason to make the order split technology available for the community.



Sandwich Attacks on Uniswap V2

Parameters

Tokens to Swap
● Select from list
From: WETH To: Aave
○ Paste ERC20 token contracts
○ Set your own liquidity pools

Pool Content
Amount WETH:
5773.932975014097
Amount AAVE:
55483.56975962012

Transaction
Input Amount (WETH):
123
Slippage Tolerance:
0.1% [✓ 0.5%] 1% x.y%
Gas Price (Gwei):
150
Current Price: 12

Options
✓ Attacker has to pay for gas.
(Not the case when attack is executed by miner or flashbot searcher.)
☐ Display liquidity pools

Generate Sandwich Attack

Liquidity Taker Attack [1]

Attacker Tx1	Victim Tx	Attacker Tx2
Swap	Swap	Swap
Input Amount (WETH): 14.59003532838876	Input Amount (WETH): 123	Input Amount (AAVE): 139.42845568125264
Output Amount (AAVE): 139.42845568125264	Output Amount (AAVE): 1148.1526924131183	Output Amount (WETH): 15.123987096853051
Update Attack		Total
		Revenue (WETH): 0.5339518
		Profit (WETH): 0.5114518

Mitigation by Order Split

Victim output without attack (AAVE): 1153.8932965
Victim output with attack (AAVE): 1148.1526924
Victim output with order split (AAVE): 1153.8636363

Victim output without attack (AAVE) minus gas: 1153.7203287
Victim output with attack (AAVE) minus gas: 1147.9797247
Victim output with order split (AAVE) minus gas: 1152.8258298
Savings (AAVE): 4.8461051

Inputs for order split (WETH): 21.92138 , 22.01223 , 22.04438 , 22.135 , 22.2132 , 12.67378
Only profitable attacks are being considered. This order split doesn't take potential arbitrage transactions into account.

Figure 4.1: Screenshot of www.DeFi-Sandwi.ch

4.3 Implementing a Web Interface

We built a tool, publicly available on www.DeFi-Sandwi.ch, that automatically checks whether a trade on Uniswap V2 can potentially be sandwich attacked. If this is the case, a suitable order split is suggested.

4.3.1 Functionalities

Users can choose the assets to be swapped from a predefined list or insert smart contract addresses of any ERC-20 tokens. If the respective trading pair exists on Uniswap, the contents of the liquidity pools are displayed. Users can also specify their own token pools; a functionality that allows for experimentation independent of the current Ethereum state. For their transaction, users need to specify the desired input amount, the slippage tolerance, and the gas price. According to the notions described in chapter 2.2.2, the tool generates an ideal sandwich attack and shows the potential profit of the attacker, as well as the losses of a trader. If a profitable sandwich attack is possible, a suitable order split is suggested, according to the calculations in 4.1. The respective savings are also calculated.

4.3.2 Architecture

The tool is built with Javascript and HTML/CSS. Almost all logic is implemented client-side which allows users to inspect the code and hence has an educative component. The backend is implemented using Express.js and mainly serves as a gateway to the Ethereum network. The respective connection is established by using the web3.js and Infura APIs. In the backend we also monitor how the tool is being utilized.

4.3.3 User Monitoring

We combine our own monitoring data with Google Analytics to better understand the usage of our tool. The daily activity varies drastically, and while we saw up to 100 unique users on some days, there were no users at all on other days. Most visitors are from the United States, Turkey, Switzerland, Australia, and China. Our data suggests that the tool is indeed used to verify that specific swaps cannot be attacked. We have however not found any traces of these trades or the suggested order splits in the actual blockchain.

4.3.4 User Feedback

We gathered feedback for the tool by posting it on several social networks and in discussion groups, as well as through our survey surrounding user perception (see chapter 5). All responses were positive and most suggestions for improvements revolved around additional features or design changes. It also confirmed that there was no product with similar functionalities available. Based on this feedback, we already made changes to the tool which include a simplification of design, and the inclusion of additional explanations.

Chapter 5

User Perception of Sandwich Attacks

To fully comprehend sandwich attacks, it is not enough to analyze block receipts; the human factor plays an important role as well. We thus initiated a survey to understand how users perceive these kinds of attacks and what mitigation strategies are used. Despite considerable effort, it was difficult to find traders who were aware of sandwich attacks and willing to fill out a survey. This again illustrates the large information gap in this area. We thus decided to focus the survey on experts of the Ethereum community. It is an indicator for future research in that domain and an opportunity to set this work and the developed tool into perspective.

With a comprehension question at the start, it was verified that the participants had a correct understanding of how sandwich attacks worked. In total, there were valid responses by 13 people. Six of them self-identified as researchers, four as developers, and three as investors.
Most participants (77%) have been personally sandwich-attacked or know somebody who has been attacked. This is, however, not how they became aware of the topic: All participants learned about it by reading research publications or posts on Twitter. One person also mentioned using Sandwiched.wtf [19] to check whether transactions from their account have been attacked.
While almost all participants agree that DeFi users can be negatively impacted by sandwich attacks, a clear minority considers the attacks immoral (see Figure 5.1). Two people even brought up the point that sandwich bots should not be considered attackers. One participant wrote: “The use of the word ‘attack’ frames the discussion in a way that leaves no room for differing opinions, and in my opinion shows a lack of understanding about the ethics of markets.”

The participants were asked to estimate different data points which we collected in this study. For the number of sandwich attacks per day, the answers vary across the whole spectrum of options without any clear favorites: While some believe that less than 500 attacks occur per day, others suspected it was more

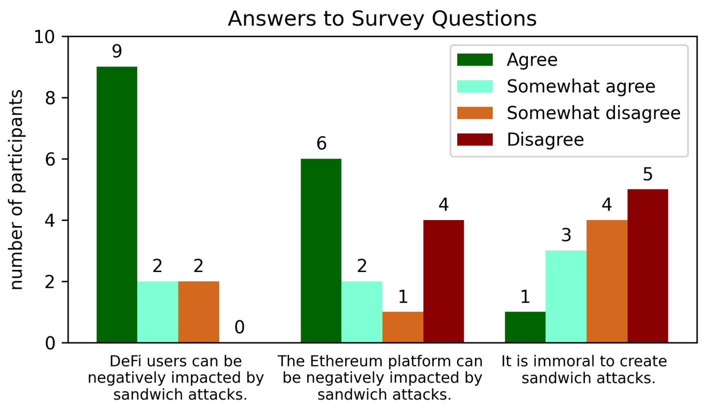


Figure 5.1: Responses given by survey participants about statements concerning the impact of sandwich attacks.

than 50'000. In April 2021, there were on average 2'803 sandwich attacks per day.
The guesses also differ for the average daily profit, which was 430 ETH in April 2021. The responses of the participants are again evenly distributed across all options from 50 ETH to 10'000 ETH. The probability that a profitable victim transaction is attacked was estimated to be at least 60% by all participants but one. This is an overestimation compared to the numbers we present in chapter 3.2.3, even if only swaps with ETH as input token are considered. However, we do not know the ratio of transactions that were included by miners without being broadcasted publicly, so it is not possible to have a definitive answers for this question.

When the participants were asked about how they had tried to prevent sandwich attacks, more than half of them mentioned consciously setting a low slippage rate. Two people said they had used private transaction pools to submit their trades. We also asked them about mitigation strategies they were aware of but had not tried: Six people mentioned submitting transactions via Flashbots. The technical barrier for this seems to be high though. One person mentioned mistX [20], a decentralized exchange that runs entirely on the Flashbots network, as a possible way to prevent sandwich attacks without having to deal with technical details.

Overall, the survey shows how even experts are unsure about the actual impact and frequency of sandwich attacks. The most effective mitigation strategies present high technical barriers, and most participants agree that the attacks can negatively impact DeFi users and the Ethereum network.

Chapter 6

Conclusion and Future Work

We used block analysis to investigate sandwich attacks on decentralized exchanges in Ethereum. Compared to related work, we apply heuristics better suited for the current DeFi environment and present more recent and extensive data. To our knowledge, it is the first time that the strategy of attackers and the impact of active reordering by miners is systematically documented. We established order splitting as a mitigation strategy in a theoretical framework, verified its effectiveness using backtesting, and created a public tool that suggests ideal order splits for given trades. Our user study showed the large information gaps surrounding sandwich attacks and confirmed the need for continued research in that area.

There are various ways our data analysis can be expanded. The Ethereum platform and the DeFi ecosystem are undergoing rapid changes. Extending the block analysis for a longer time period will reveal how sandwich attacks are influenced by different factors, like Ethereum's switch to proof of stake, the release of Uniswap V3, or the continuous proliferation of the Flashbots project. The preliminary results of this study were met with a lot of interest when shared privately with Ethereum experts, so we want to make them available in a concise form for the community. A possible extension of the historical data analysis could be the creation of a live feed of sandwich attacks happening in the Ethereum network.
The tool we built is a research prototype and there are several possibilities to make it more user-friendly. Apart from improving the design, it could be extended by including multiple exchanges, or functionality that lets users directly sign and broadcast the suggested order split transactions.
A future research project will revolve around the user perception of sandwich attacks. For this purpose, we will adapt and extend our survey with more participants. We will also conduct in-depth interviews to better understand the human perspective on sandwich attacks. In general, we hope that with our work we were able to contribute to the ongoing and important MEV research.

Bibliography

[1] R. Leshner and G. Hayes, "Compound finance whitepaper," Feb. 2019, [Accessed Mar. 13, 2021]. [Online]. Available: https://compound.finance/documents/Compound.Whitepaper.pdf

[2] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core whitepaper," Mar. 2021, [Accessed May. 07, 2021]. [Online]. Available: https://uniswap.org/whitepaper-v3.pdf

[3] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," in *arXiv preprint arXiv:1904.05234*, Apr. 2019.

[4] R. Dan and K. Georgios, "Ethereum is a dark forest," Aug. 2020, [Accessed Feb. 15, 2021]. [Online]. Available: https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dfff

[5] "Flashbots," [Accessed May 25, 2021]. [Online]. Available: https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dfff

[6] P. Daian, "Mev... wat do?" [Accessed Jun. 1, 2021]. [Online]. Available: https://pdaian.com/blog/mev-wat-do/

[7] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-frequency trading on decentralized on-chain exchanges," in *arXiv preprint arXiv:2009.14021v1*, Sep. 2020.

[8] "Taichi network," [Accessed May. 13, 2021]. [Online]. Available: https://taichi.network/

[9] "Stealth transactions on 1inch," [Accessed Jun. 10, 2021]. [Online]. Available: https://help.1inch.io/en/articles/4695716-what-are-stealth-transactions-and-how-do-they-work/

[10] "Comparison of dex usage," [Accessed Apr. 25, 2021]. [Online]. Available: https://coinmarketcap.com/de/rankings/exchanges/dex/

[11] J. W. Markham, "Front-running - insider trading under the commodity exchange act," in *38 Cath. U. L. Rev. 69*, 1988.

[12] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" in *arXiv preprint arXiv:2101.05511v2*, Jan. 2021.

[13] J. C. Shayan Eskandari, Seyedehmahsa Moosavi, “Sok: Transparent dishonesty: Front-running attacks on blockchain,” in *arXiv preprint arXiv:1902.05164v3*, Apr. 2019.

[14] C. F. Torres and R. Statee, “Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain,” in *arXiv preprint arXiv:2102.03347v1*, Feb. 2021.

[15] “Flashbots dashboard,” [Accessed Jun. 25, 2021]. [Online]. Available: https://dashboard.flashbots.net/network

[16] “Wrecking sandwich traders for fun and profit,” [Accessed Jun. 24, 2021]. [Online]. Available: https://github.com/Defi-Cartel/salmonella

[17] R. Miller, “Twitter thread about attacks on sandwich bots,” [Accessed Jun. 24, 2021]. [Online]. Available: https://twitter.com/bertcmiller/status/1381296074086830091

[18] D. J. Thomas and J. E. Tyworth, “Pooling lead-time risk by order splitting: A critical review,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 42, no. 4, pp. 245–257, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1366554505000037

[19] “Sandwiched.wtf,” [Accessed Jun. 25, 2021]. [Online]. Available: https://sandwiched.wtf/

[20] “mistx,” [Accessed Jun. 25, 2021]. [Online]. Available: https://mistx.io/