



# Rolling in the Shadows: Analyzing the Extraction of MEV Across Layer-2 Rollups

Christof Ferreira Torres  
ETH Zurich  
Zurich, Switzerland  
christof.torres@inf.ethz.ch

Albin Mamuti  
ETH Zurich  
Zurich, Switzerland  
amamuti@student.ethz.ch

Ben Weintraub  
Northeastern University  
Boston, Massachusetts, USA  
weintraub.b@northeastern.edu

Cristina Nita-Rotaru  
Northeastern University  
Boston, Massachusetts, USA  
c.nitarotaru@northeastern.edu

Shweta Shinde  
ETH Zurich  
Zurich, Switzerland  
shweta.shinde@inf.ethz.ch

## Abstract

The emergence of decentralized finance has transformed asset trading on the blockchain, making traditional financial instruments more accessible while also introducing a series of exploitative economic practices known as Maximal Extractable Value (MEV). Concurrently, decentralized finance has embraced rollup-based Layer-2 solutions to facilitate asset trading at reduced transaction costs compared to Layer-1 solutions such as Ethereum. However, rollups lack a public mempool like Ethereum, making the extraction of MEV more challenging.

In this paper, we investigate the prevalence and impact of MEV on Ethereum and prominent rollups such as Arbitrum, Optimism, and zkSync over a nearly three-year period. Our analysis encompasses various metrics including volume, profits, costs, competition, and response time to MEV opportunities. We discover that MEV is widespread on rollups, with trading volume comparable to Ethereum. We also find that, although MEV costs are lower on rollups, profits are also significantly lower compared to Ethereum. Additionally, we examine the prevalence of sandwich attacks on rollups. While our findings did not detect any sandwiching activity on popular rollups, we did identify the potential for cross-layer sandwich attacks facilitated by transactions that are sent across rollups and Ethereum. Consequently, we propose and evaluate the feasibility of three novel attacks that exploit cross-layer transactions, revealing that attackers could have already earned approximately 2 million USD through cross-layer sandwich attacks.

## CCS Concepts

• **Security and privacy** → **Domain-specific security and privacy architectures; Distributed systems security.**

## Keywords

Ethereum; layer-2; rollups; MEV; cross-layer; sandwiching; DeFi

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0636-3/24/10

<https://doi.org/10.1145/3658644.3690259>

## ACM Reference Format:

Christof Ferreira Torres, Albin Mamuti, Ben Weintraub, Cristina Nita-Rotaru, and Shweta Shinde. 2024. Rolling in the Shadows: Analyzing the Extraction of MEV Across Layer-2 Rollups. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24)*, October 14–18, 2024, Salt Lake City, UT, USA. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3658644.3690259>

## 1 Introduction

Blockchain markets have revolutionized the finance industry. Entirely new classes of assets are now regularly traded at massive scales. Partially responsible are smart contract technologies and the subsequent advent of decentralized finance (DeFi), which create markets that are transparent and freely accessible to anyone with an internet connection. As of April 10th, 2024, over 95 billion USD worth of assets have been deposited across the major DeFi smart contracts platforms [24], a number comparable to the 90 billion USD market cap of well-known traditional banks such as UBS [16], thus grounding the magnitude of DeFi's popularity.

However, with DeFi's important position within the global economy comes commensurate traffic and trade volume. Ethereum, the most popular among DeFi ecosystem blockchains [17], has seen an increase in traffic thus increasing competition to push transactions through a throughput-limited system. This competition for a limited resource has resulted in increased transaction fees. Several solutions have been proposed for increasing Ethereum's throughput. While some of these solutions are improvements in the Ethereum protocol (e.g., sharding [22, 68]), others operate as a layer of abstraction over the underlying Ethereum blockchain. One method that has seen support by the Ethereum community, is the outsourcing of transaction processing to external solutions. These solutions can be characterized into four categories; side-chains (e.g., Polygon [51]), generic bridges (e.g., Ronin [12], Wormhole [67], etc.), payment or state channels (e.g., Raiden [43], Perun [49], etc.), and rollups (e.g., Arbitrum [6], zkSync [73]). This class of solutions is often referred to as *Layer-2* or *L2*, whereas Ethereum is referred to as *Layer-1* or *L1*. Among the four categories, rollups have emerged to be the most adopted solution [38]. This is due to their low transaction fees, high transaction throughput, and compatibility with Ethereum. At the time of writing, a simple transaction on Ethereum to send cryptocurrency to another address costs about 1.47 USD,

whereas the same transaction could be performed on any of the popular rollups for less than 0.01 USD [39].

Rollups operate by allowing users to lock their assets on Ethereum, which grants them access to trade equivalent assets on a rollup chain. Users can make trades on the rollup as frequently and to whatever degree their capital allows. Rollups perform periodic state checkpoints with the Ethereum blockchain, which consist of batches of compressed rollup transactions stored in a data field of a traditional Ethereum transaction. Each rollup has its own ordering mechanism, a service called a *sequencer*. In most cases, this is a centralized server operated by the rollup developer following a first-come, first-served strategy. Because rollups follow a centralized approach towards transaction ordering, the transaction throughput can be much higher than in Ethereum. Moreover, as multiple rollup transactions can be compressed into a single Ethereum transaction, the high fees associated with the Ethereum transaction can be amortized across all rollup transactions of the same batch.

Maximal Extractable Value (MEV) has become an essential part of Ethereum [56, 59]. The idea behind MEV is to extract value (i.e., monetary profit) from transactions performed by other entities on the blockchain by influencing their order of execution. Several strategies have been proposed, the most popular ones being arbitrage, liquidation, and sandwiching [56, 59]. MEV is a double-edged sword. While strategies such as arbitrage and liquidation help maintaining markets healthy across different DeFi protocols (e.g., liquidation of unhealthy loans, price balancing across exchanges, etc.), they can also have negative side effects. For example, they result in increased transaction fees as a result of gas price auctions [21] or wasteful transactions due to failed attempts to extract MEV. Some strategies, such as sandwiching, are even considered entirely destructive. Sandwiching occurs, for example, by placing a buy order right before a pending trade (i.e., frontrunning) and a sell order right after it (i.e., backrunning), thereby forming a “sandwich” and resulting in traders making less profit on their trades.

In contrast to Ethereum, rollups do not provide a mempool that publicly advertises pending transactions. Only sequencers can see transactions before they are finalized. Thus, normal users can only make use of information contained in the latest block to extract MEV. This means that arbitrages and liquidations are still possible, but that traditional sandwich attacks are not possible anymore since these require extractors to identify victim transactions upfront and then frontrun them before they are finalized. Hence, only sequencers have the power to mount such attacks. However, sequencers are assumed to be trusted entities and, therefore, users assume that sandwich attacks are not an issue on rollups.

In this paper, we shed light into how much MEV is being extracted on popular rollups such as Arbitrum [6], Optimism [46], and zkSync [73] as these are the rollups with the highest market share in terms of total value locked at the time of writing [38]. We analyze how MEV extraction on these rollups compares to Ethereum in terms of volume, profit, and costs across a 32 month period. We also analyze the usage of flash loans, code reuse and competition among MEV extractors as well as the time extractors take to respond to MEV opportunities. Furthermore, we check for traditional sandwiching on rollups, confirming that so far sequencers behave as intended and do not perform any evident sandwiching. Nonetheless, after carefully studying the way transactions are sent across

rollups and Ethereum, we propose and evaluate three novel cross-layer sandwich attacks that normal users can leverage to mount sandwich attacks on rollups. These attacks exploit transactions that perform trades on rollups but which are emitted via Ethereum. Our simulation using past mainnet data reveals that attackers could have made roughly 2 million USD profit. Finally, we also discuss the generalization of our attacks and potential countermeasures.

**Ethical Considerations.** For ethical reasons, we demonstrate the feasibility of our attacks by executing them only on the testnet, thereby targeting only our own victim transactions. On the mainnet, we only perform a local simulation of our attacks and do not actually carry out the attacks by broadcasting the transactions. The attacks performed on the testnet did not rely on real victim transactions, but on victim transactions emitted by us. Our simulation on real victim transactions on the mainnet cannot have an impact anymore on these victims as their transactions are already finalized on L1 and L2, and cannot be exploited anymore.

**Contributions.** We summarize our contributions as follows:

- We conduct the first large-scale measurement of MEV practices across Arbitrum, Optimism, and zkSync, and compare them to Ethereum over a period of nearly 3 years.
- We present novel insights in terms of volume, profits, costs, flash loans, code reuse, competition, and response time to MEV opportunities across Ethereum and rollups.
- We propose three novel cross-layer sandwich attacks and simulate them using real mainnet data from Arbitrum and Optimism. Our simulation shows that attackers could make approximately 2 million USD profit via cross-layer sandwich attacks.

## 2 Background

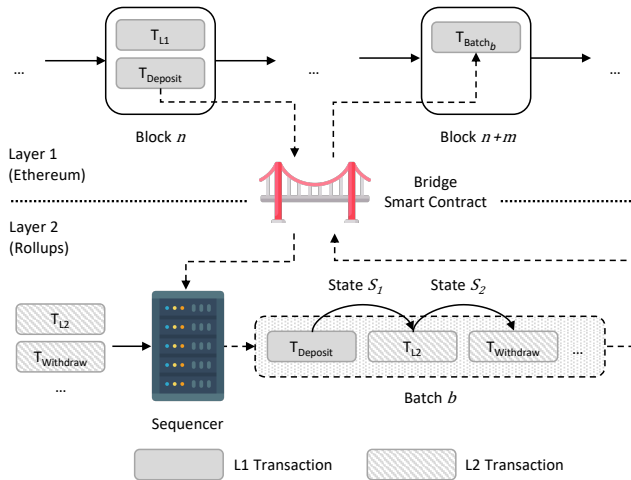
In this section, we provide background on Ethereum, rollups, and Maximal Extractable Value.

### 2.1 Ethereum

Ethereum is a blockchain and cryptocurrency system. The Ethereum blockchain consists of a series of ordered blocks, which themselves contain a sequence of ordered transactions. Transactions can be simple exchanges of the main currency, Ether (ETH), or they can contain more complex programmatic logic called smart contracts.

Smart contracts are Turing-complete programs executed via the Ethereum Virtual Machine (EVM). They could execute infinitely if not for a particular design intervention called *gas*. Gas represents a fee that is charged for every instruction executed by a smart contract. The relationship between gas and instructions is deterministic (e.g., an instruction that adds two numbers always costs three gas units [66]). Gas is paid in the form of ETH in proportion to the so-called gas price, which dictates how much ETH one gas unit costs. To prevent unexpectedly expensive computations, every transaction is provided a gas limit, which means that the execution will stop once the limit has been reached. When a transaction is aborted or reverted, external changes are rolled back but the gas, and the resulting ETH, is still spent.

Transaction ordering is at the discretion of the block builders, who construct blocks of pending transactions and then send them to proposers who validate them and send them out to the network to be added to the head of the blockchain. Rational block builders



**Figure 1: An overview on the interplay between Ethereum (Layer 1) and rollups (Layer 2).**

order transactions based on the gas fees they will accrue when the block is validated, thereby giving priority to transaction that yield a higher income. In effect, a user can get a transaction to appear higher in a block by increasing the gas price they are willing to pay.

Smart contracts cannot communicate directly with the off-chain world (e.g., programs running outside the blockchain) as the EVM runs as an isolated synchronous deterministic environment. However, the EVM allows smart contracts to emit *events*, which allow developers to store data during execution that is persistent and quickly accessible by applications running outside the blockchain.

## 2.2 Rollups

Scalability and cost remain the salient issues of blockchain-based cryptocurrencies. While numerous strategies have been proposed for increasing transaction throughput, from payment channel networks [43, 52] to sharding [37], rollups have emerged as the favored solution for Ethereum.

A rollup derives its benefits by bringing the expensive computations off of Ethereum i.e., Layer-1 (L1), and uses L1 only for storage and state checkpointing. This class of solutions and applications are called Layer-2 (L2) solutions, highlighting the fact that they are conceptually built on top of L1. A user enters a rollup by depositing funds to an L1 smart contract, called a *bridge*. Bridges allow users to transfer assets between separate blockchains and/or layers (see Figure 1). In this case, it transfers assets from the L1 blockchain to the L2 rollup. Deposited funds become spendable within the rollup but unspendable on L1.

Rollups can either be *optimistic* or *zero-knowledge*. Optimistic rollups assume that the sequencer is behaving correctly, but allows users to dispute any irregularities within a challenge period—it is a “trust but verify” model. On the other hand, zero-knowledge rollups (ZK rollups) are untrusted from the start. Every ZK rollup must include a cryptographic validity proof to be considered valid. When a user wishes to make a payment, they submit the payment directly to the *sequencer*. The sequencer accepts these transactions

and organizes them into batches, which are analogous to blocks in L1. It executes the transactions in an off-chain EVM to make sure they are valid and also to track the state changes that may result from contract execution. Table 1 provides a comparison between Ethereum and different rollup releases studied in this work. Currently there is only a single sequencer in each of the major rollup platforms and none of them provide a public mempool. For most rollups the ordering of transactions follows a “first-come, first-served” (FCFS) policy, whereas on Ethereum transaction ordering can be done arbitrarily at the discretion of the block builder.

Periodically, after assembling several batches, a sequencer compresses the batches and puts them into a raw data field—called *calldata*—in an L1 Ethereum transaction. The sequencer then submits this well-formed Ethereum transaction to the L1 blockchain. Its visibility and availability on L1 allow participants to look at the published transactions and dispute any inconsistencies. Since the the sequencer is trusted to process these L2 transactions correctly, and should be able to do so using a standard EVM implementation, any deviations from correct behavior are assumed to be malicious. For optimistic rollups, users can submit fraud proofs during the challenge period if the rollup is invalid for whatever reason. After the challenge period ends, the transactions can no longer be reversed and have achieved *finality*.

Users can get their funds out of a rollup by submitting a request to the sequencer. The sequencer includes a transaction in the rollup that *burns* the user’s funds on the rollup. This means that the funds on the rollup are no longer spendable. Meanwhile, a similar transaction, called a *withdrawal proving transaction* is propagated to L1, which confirms that the L2 transaction did occur—this prevents a user from double spending both the L2 funds and their deposited L1 funds. The last step is another L1 transaction called a *withdrawal finalizing transaction* that confirms that the challenge period has ended, which ensures that the funds do not become spendable on L1 if they are disputed on L2. Note that these extra assurances are not needed for ZK rollups, which results in simplified exit procedures.

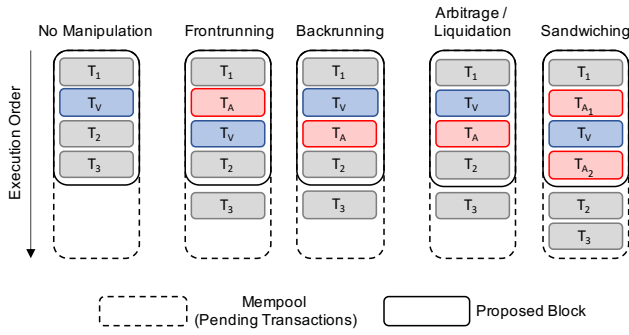
## 2.3 Maximal Extractable Value

Transactions in Ethereum are typically publicly visible before they are officially committed to the blockchain via the mempool (i.e., pool of pending transactions). Similar to traditional financial markets, Ethereum users have noticed that profit can sometimes be made off of these uncommitted transactions [40]. Daian et al. [21]

Chain	Release	Rollup Type	Mempool	Ordering
Ethereum	-	-	Public	(D) Gas Price
Arbitrum	Classic	Optimistic	Private	(C) FCFS
Arbitrum	Nitro	Optimistic	Private	(C) FCFS
Optimism	Pre-Bedrock	Optimistic	Private	(C) FCFS
Optimism	Post-Bedrock	Optimistic	Private	(C) Gas Price
zkSync	Era	Zero-Knowledge	Private	(C) FCFS

(D): Decentralized, (C): Centralized

**Table 1: A comparison between Ethereum and rollups studied in this work (i.e., Arbitrum, Optimism, and zkSync).**



**Figure 2: Examples of common MEV extraction strategies. Blue boxes depict victim transactions and red boxes depict MEV transactions for each type.**

first characterized this phenomenon, which we now call *Maximal Extractable Value* (MEV).

In general, this value is extracted through a process of influencing transaction ordering. There are many contexts in which this is applicable. The Ethereum ecosystem allows for the creation of *tokens* via smart contracts. These act as individual currencies, with value independent of Ethereum’s native currency, ETH. A decentralized exchange (DEX) operates much like a traditional currency exchange. A user requests to trade in some amount of one token for the equivalent value in another token. The exchange rate can either come from an off-chain source, which is manually administered, or it can be modulated dynamically on-chain via so-called *Automated Market Makers* (AMMs). The most popular DEX protocols across chains in terms of Total Value Locked (TVL) [23] are *Uniswap* [61] (including forks such as *SushiSwap* [58]), *Balancer* [9], and *Curve* [20]. Apart from DEXes, lending platforms also play a prominent role in MEV extraction. Users can borrow assets from these platforms given that they provide another asset as collateral and payback the loan including some interest. At the time of writing, the most popular lending platforms across chains in terms of TVL [25] are *Aave* [1] and *Compound* [18].

There are two transaction ordering primitives that are leveraged to extract MEV: *frontrunning* and *backrunning* (see Figure 2). Backrunning is when the MEV extractor ensures that their own transaction  $T_{MEV}$  is only executed *after* some target transaction  $T_{Target}$ . Examples of backrunning include liquidating loans as quickly as possible or selling an asset that just had its price increase as the result of a large target transaction  $T_{Target}$ . The alternative ordering primitive is called *frontrunning*. Frontrunning is when an MEV extractor manipulates the ordering such that their own transaction  $T_{MEV}$  precedes some target transaction  $T_{Target}$ . The motivation for such an action could be either to acquire some asset before its price changes as a result of a large victim transaction  $T_{Target}$ , or to be the first to execute some contract (e.g., copying profitable transactions from other users [55]).

There exist three popular MEV extraction techniques that exploit the aforementioned transaction ordering primitives: *arbitrage*, *liquidation*, and *sandwiching* (see Figure 2). Arbitrage is the practice of concurrently selling and purchasing assets across different

exchanges to capitalize on variations in market prices. Arbitrageurs engage in arbitrage by monitoring blockchain state changes. These changes can be monitored by either analyzing pending transactions via the mempool or by analyzing the state of the latest block. Liquidations enable users to purchase collateral at a discount when repaying debt. The discount can either be fixed or determined via an auction. MEV extractors typically focus on liquidations with fixed discounts as these can be performed via a single transaction. Similar to arbitrage, liquidators either analyze the mempool or the state of the latest block to find opportunities. Sandwiching is a classic trading strategy and well-known in traditional finance. It involves analyzing pending transactions and wrapping a victim’s pending transaction  $T_V$  within two adversarial transactions  $T_{A_1}$  and  $T_{A_2}$ . Typically,  $T_V$  aims to perform a large trade, whereas the sandwicher first frontruns  $T_V$  by buying the same asset at a cheaper price and afterwards backruns  $T_V$  to sell the purchased asset at a much higher price, thereby profiting from  $T_V$  (see Figure 2).

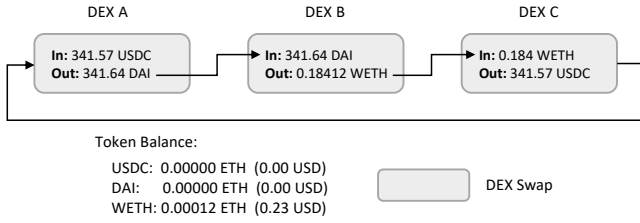
Arbitrage and liquidation are generally viewed as “good” MEV as they play an essential role in fostering market health. For example, arbitrage helps DEXes in keeping their prices synchronized across other DEXes. However, sandwiching is considered “bad” MEV, since sandwichers manipulate the price that other traders get. But the dichotomy is not always so simple and even good MEV can have negative side effects. In particular, gas price auctions [21]—when multiple transactions are competing for preferential block position by increasing their gas price—results in increased transaction fees for all uses and produces block congestion as a result of the flood of MEV extractions. Flashbots aims to solve these issues via private mempools [28]. However, Flashbots only operates on Ethereum and rollups currently do not provide public mempools. Rollups also typically order transactions based on FCFS. Hence, sandwiching is theoretically only possible by the sequencer. Moreover, arbitrage and liquidation can be performed by anyone observing the latest state and optimizing transaction latency. Additionally, block congestion is still possible, since arbitrageurs and liquidators cannot observe competitors in real time as opposed to Ethereum.

### 3 Detecting MEV Across Layers

In this section, we describe our methodology on detecting arbitrage, liquidation, sandwiching, flash loans, MEV opportunities, and competition across Ethereum, Arbitrum, Optimism, and zkSync through the analysis of historical blockchain data.

#### 3.1 Detecting Arbitrage

We detect arbitrage based on heuristics proposed by previous works on detecting cyclic arbitrage in DEXes [56, 64, 65]. We start by scanning past blocks for token swap events emitted by DEXes such as Uniswap V2, Uniswap V3, Balancer V1, Balancer V2, and Curve (see Appendix A). A swap event denotes a successful exchange of a token  $A$  for another token  $B$ . In other words,  $X$  amount of token  $A$  going into the DEX smart contract and  $Y$  amount of token  $B$  going out of the DEX smart contract. DEXes may implement and call these events differently. For example, while Uniswap V2 and Uniswap V3 emit the same event called “*Swap*”, their event topic hashes are different since they encode different values (e.g., `Swap(address,uint,uint,uint,uint,address)` for Uniswap V2



**Figure 3: Example of our arbitrage detection across 3 swaps.**

and Swap(address, address, int256, int256, uint160, uint128, int24) for Uniswap V3). Once we retrieve all swap events, we extract information such as: token address in, token address out, token amount in, token amount out, and the address of the DEX that performed the swap. Afterwards, we group swaps by the transactions they are a part of and iterate through these swaps following the order in which they were emitted. Next, we link the swaps together, since arbitrages are essentially a chain of swaps. Hence, for each swap we check whether the token address out is equivalent to the token address in of the subsequent swap and whether the token amount out of the current swap is equivalent or higher than the token amount in of the subsequent swap (see Figure 3). Moreover, we also verify that the DEX address of the current swap is different from the DEX address of the subsequent swap. We find an arbitrage whenever we encounter a swap where the token address out is equivalent to the token address in of the very first swap of the current sequence, thus, forming a cycle. We restart the process for the remaining swap events to detect further arbitrages, since a single transaction may perform multiple arbitrages.

For every detected arbitrage, we keep track of the token balance of each token involved. We start with a balance of zero for every token and add or deduct the token amount depending on whether the token is going into the DEX (i.e., adding token amount) or going out of the DEX (i.e., deducting token amount). We then compute the cost of each arbitrage by converting all balances to Ether using CoinGecko’s public API [15] and assigning negative balances to costs and positive balances to gains. The profit of the arbitrage is computed by simply deducting the costs from the gains. However, this does not return the final profit of the arbitrage transaction as the transaction fees are missing and a transaction may contain multiple arbitrages. Thus, we compute the final profit by first summing up all the profits of all the arbitrages involved in the transaction and then deducting the transaction fees. In case of Ethereum, we also check whether the transaction was part of a Flashbots bundle using Flashbots’ public API [29]. If the transaction is part of a bundle, we deduct the coinbase transfer value associated to the transaction. The coinbase transfer is another way (besides traditional transaction fees) for MEV extractors to transfer funds to block producers. Hence, MEV extractors use coinbase transfers for bribing (i.e., incentivizing) block producers to prioritize the inclusion of their transaction in the next block.

### 3.2 Detecting Liquidation

We detect liquidations by scanning past blocks for specific liquidation events emitted by smart contracts of popular DeFi lending

protocols. In this work, we analyze liquidations performed by smart contracts that follow *Aave*’s [2] and *Compound*’s [4] lending protocol implementations. In detail, our detection script searches for *Aave*’s V1, V2, and V3 “*LiquidationCall*” event topic hashes and *Compound*’s V2 “*LiquidateBorrow*” event topic hash (see Appendix A). These events are triggered whenever a loan has been successfully liquidated. For each event we extract information such as liquidator, liquidated user, liquidated debt, received collateral, etc.

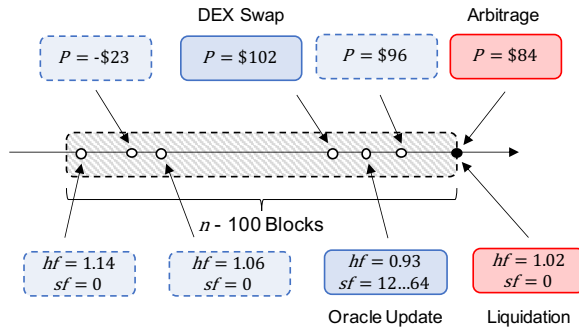
Loans that follow *Compound*’s protocol do not directly return the received collateral. Instead, users receive the collateral in form of “cTokens”, which they can leave in *Compound* to gain interest, or redeem for the actual collateral. We therefore also scan the block for emitted “*Redeem*” events, which allows us to obtain the actual collateral, including its actual amount. We compute the profit for each liquidation by deducting the cost, which is the value of the debt token, from the gain which is the value of the collateral. Similar to the arbitrage detection, we convert the value of both, debt and collateral, into its equivalent amount in Ether using CoinGecko’s public API [15]. MEV extractors might liquidate multiple loans within the same transaction. The final profit is computed by summing up all the profits of each liquidation and deducting the transaction costs as well as any tips that the MEV extractor may have paid to the block proposer via coinbase transfers (the latter only applies if the MEV extractor performed the liquidation on Ethereum).

### 3.3 Detecting Sandwiching

We base our sandwich detection on a combination of heuristics proposed by previous works [56, 59]. We begin by scanning past blocks for token transfer events. For Ethereum, we analyze the events per block, whereas for the rollups we scan token events across multiple blocks (i.e., batches) using a sliding window of 100 blocks with a distance of one block apart. Due to the small size of L2 batches, it is more likely that sandwiches will be spread across batches. Theoretically this could also be the case for Ethereum, however previous works [56, 59] have observed that analyzing sandwiches per block is sufficient in practice.

For each detected transfer event we aim to find its counterpart, namely a transfer event that was emitted afterwards via a different transaction by the same token smart contract but which transfers the tokens in the opposite direction. Hence, we search for transactions  $T_{A_1}$  and  $T_{A_2}$ , such that  $T_{A_2}$  has a larger transaction index value than  $T_{A_1}$  and where the sender of  $T_{A_1}$  is equivalent to the receiver of  $T_{A_2}$  and the receiver of  $T_{A_1}$  is equivalent to the sender of  $T_{A_2}$ . Moreover, the amount transferred in  $T_{A_2}$  has to be either the same or smaller than the amount transferred in  $T_{A_1}$ . The same applies for rollups, except that we verify the order by first checking batch height and then within the same batch the transaction index. Once such a pair has been identified, we search for one or multiple victim transactions (i.e.,  $T_V$ ), by checking if there are any token transfer events emitted by the same token in between the identified attacker events, which have the same sender as  $T_{A_1}$  but a different receiver. The sender for  $T_{A_1}$  and  $T_V$  are expected the same since it is the address of the same DEX, however the receiver must be different since it is the address of the actual attacker/victim.





**Figure 4: Example of our opportunity detection methodology.**  $P$  means profit and a change from a negative  $P$  to a positive  $P$  indicates an opportunity for arbitrage. An  $hf$  below 1.0 or an  $sf$  above zero indicates an opportunity for liquidation. Boxes on top show arbitrage, boxes below show liquidation.

### 3.4 Detecting Flash Loans

Flash loans are risk-free loans which enable users to borrow a large amount of assets without being required to provide any collateral as a security [63]. Flash loans exploit the atomicity of blockchain transactions. Users are required to pay back the loan plus some interest within the same transaction. Otherwise the entire transaction is reverted, meaning that the transaction has no effect and the borrowed assets are returned back to the flash loan provider. There exist several DeFi protocols that provide flash loans.

In this work, we analyze and compare two of the largest protocols: *Aave* [2] and *Balancer* [10]. These two protocols are deployed on Ethereum, Arbitrum, Optimism, and zkSync. Flash loans enable MEV extractors to take out large quantities of MEV from protocols without requiring them to own upfront any assets in those protocols. Extractors are only required to own enough funds to pay back the interest and the execution costs of the loan. However, due to flash loans being bound to the execution of a single transaction, they can only be used to extract liquidations and arbitrages but not sandwiches as these span across multiple transactions.

For every detected arbitrage and liquidation, we scout for specific events that are emitted by flash loan providers whenever a flash loan was successful (i.e., the loan has been borrowed and paid back with interest). In more detail, for every transaction we check for the “FlashLoan” event topic hashes emitted by Aave V1, Aave V2, Aave V3, and Balancer (see Appendix A). We parse the event data for each protocol individually and extract information such as borrowed token, borrowed amount, interest fee, etc.

### 3.5 Detecting Opportunities

An opportunity is a transaction that triggers a state change which enables the extraction of MEV. For example, for an arbitrage to be possible, there needs to be a price difference for the same asset across two different exchanges. This is common for DEXes as the price of an asset is determined by their local view on supply and demand. As a result, DEXes are not aware of price changes across other DEXes. Thus, a large trade on one DEX might create a large

price difference for a particular asset, which can then be exploited by trading the asset across different exchanges.

Detecting transactions that triggered sandwiches is straightforward, since by definition sandwiches encompass the opportunity transaction (i.e., the victim transaction  $T_V$ ). The challenge resides in detecting transactions that triggered arbitrages and liquidations as these are (most often) decoupled from the actual transactions performing the MEV extraction.

**3.5.1 Detecting Arbitrage Opportunities.** Arbitrage opportunities are typically triggered via transactions performing large trades which cause price fluctuations for the an asset on a DEX. Therefore, for every arbitrage that we detect, we retrieve all previous DEX swap events that are up to 100 blocks prior to the detected arbitrage (see Figure 4). Our preliminary results have shown that 100 blocks are enough to capture most opportunities (see section 4.4). Next, we filter out all swap events which do not originate from a DEX that was involved in the arbitrage. For each of the remaining swap events we simulate the arbitrage using the events’ block number. We start with the largest block number (i.e., closest block to the arbitrage) and iterate towards the smallest block number (i.e., farthest block from the arbitrage). We stop iterating when we find a block where the simulation of the arbitrage returns a negative profit  $P$ . This indicates that the arbitrage is not profitable anymore, hence the swap transaction that triggered the arbitrage must be included within the last block where the arbitrage was profitable.

**3.5.2 Detecting Liquidation Opportunities.** A loan typically opens up for liquidation when the collateral that was provided by a user as a security for a borrowed asset loses significant value. In case of Aave, this is observable via the *health factor* ( $hf$ ) which can be obtained by calling the smart contract function `getUserAccountData(address user)` [3]. If the health factor of a loan is below 1.0, then the collateral lost a significant amount of value and the loan is open for liquidation. In case of Compound, this is observable via the *shortfall* ( $sf$ ) which can be obtained by calling the smart contract function `getAccountLiquidity(address account)` [19]. Shortfall is a positive integer and defines the amount by which a loan exceeds the required collateral. Hence, shortfall should always be zero. A loan that has a shortfall higher than 0 is liquidable. To know the current value of a collateral, protocols such as Aave and Compound rely on price oracles. Most instances of these two protocols make use of off-chain data that is available via Chainlink’s price oracles [13]. Thus, a loan typically opens up for liquidation as a result of a price update via a Chainlink oracle.

Similar to our arbitrage opportunity detection, we detect liquidation opportunities by first retrieving all previous Chainlink “AnswerUpdated” events (see Appendix A) that are up to 100 blocks away from the detected liquidation (see Figure 4). For each of the oracle update events we obtain the block number and retrieve either the  $hf$  or the  $sf$  for Aave or Compound, respectively. Again, we start with the largest block number (i.e., closest block to the liquidation) and iterate towards the smallest block number (i.e., farthest block from the liquidation). We stop iterating when we find a block where either  $hf$  is larger or equal to 1.0 or  $sf$  is equal to 0. This indicates that the liquidation is not open anymore. Hence, the oracle update transaction that triggered the liquidation must

Chain	Time Period	Block Range	
Ethereum	Jan. 1, 2021 - Aug. 31, 2023	11,565,019	- 18,037,987
Arbitrum	Mar. 28, 2021 - Aug. 31, 2023	0	- 126,855,340
Optimism	Jan. 14, 2021 - Aug. 31, 2023	0	- 108,963,811
zkSync	Feb. 14, 2023 - Aug. 31, 2023	0	- 12,689,375

**Table 2: Data collection time frame for each chain.**

be included in the previous block where  $hf$  is smaller than 1.0 or  $sf$  is larger than 0.

### 3.6 Detecting Competition

We define competition as two or more MEV extractors targeting the same MEV opportunity. Hence, we measure competition by analyzing whether two or more transactions that have been previously identified to extract the same type of MEV, have also been identified to target the same opportunity transaction as defined by our methodology on detecting MEV opportunities.

### 3.7 Limitations

Our methodology currently does not capture all MEV as it searches for hard-coded events triggered by a specific set of DeFi protocols, thus we will miss MEV emitted by other protocols. Hence, our results should be considered as a lower bound on the volume of extraction as well as in terms of comparison between DeFi protocols. Moreover, our competition measurement does not capture MEV transactions that did not make it on-chain. Similar to prior works [42, 56, 59, 65], which measured MEV on L1, our price calculation relies on CoinGecko’s public API [15] to identify the value of a token at a given time. As a result, the profit calculation might not precisely reflect the actual profit that the extractor made on that day. Moreover, CoinGecko only tracks popular tokens, thus, for less popular tokens we are unable to calculate the profits made by the extractor.

## 4 Analyzing MEV Across Layers

In this section, we analyze MEV extracted between January 1st, 2021 and August 31st, 2023 (2 years and 8 months) across Ethereum, Arbitrum, Optimism, and zkSync (see Table 2 for block ranges) using the methodology discussed in Section 3.

### 4.1 Volume

For each of the rollups, we see gradual but significant increases in the number of liquidations and arbitrages over the course of our measurement period. In the bottom plot of Figure 5 we see a lot of variability in the number of liquidations on Ethereum, vacillating between 498 and 13,121 for each four month period. For Arbitrum and Optimism, we see rapid increase in the early months after the technology was deployed. Arbitrum then stabilizes between 309 and 665 liquidations per month. Optimism appears to decrease after its peak of 1,307 liquidations in the summer of 2022. Interestingly, there is a decline in the number of liquidations on Ethereum between June 2022 and June 2023. This could be a result of a smaller number of loans or broader economic trends causing the collaterals to maintain their value with higher probability.

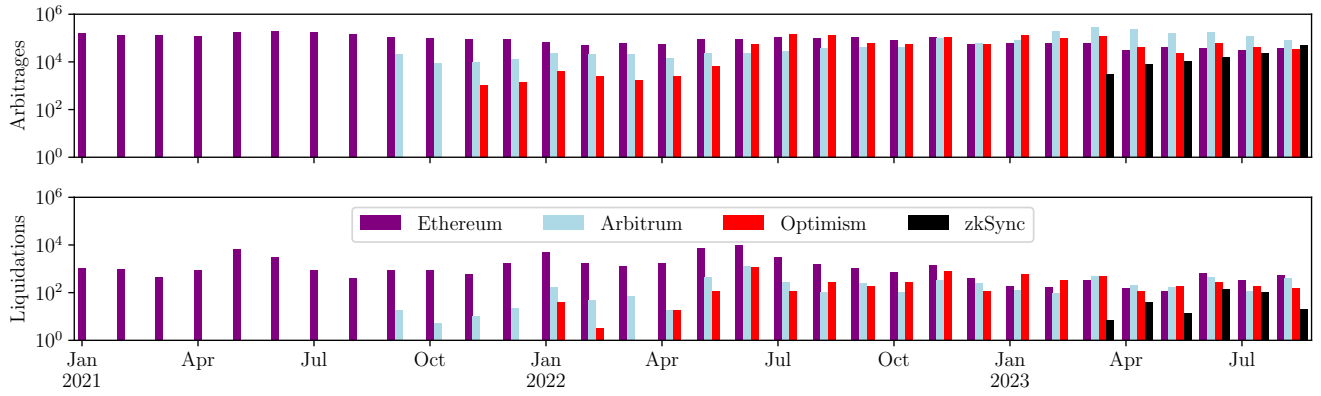
Strategy	Profit	Ethereum	Arbitrum	Optimism	zkSync
Arbitrage	Total	209,981,692.18	17,419,113.84	2,514,046.36	640,243.36
	Max	25,614,599.91	1,913,418.44	79,927.65	23,891.42
	P <sub>90</sub>	87.51	6.55	1.94	7.52
	Mean	74.12	10.14	2.21	6.12
	Median	4.80	0.27	0.23	0.83
	Min	-15,828.08	-1,659.02	-15,963.71	-21.39
Liquidation	Total	232,498,892.47	856,503.70	1,308,358.38	10,733.13
	Max	2,675,295.15	71,224.66	288,573.28	4,014.54
	P <sub>90</sub>	4,814.07	225.64	74.82	60.10
	Mean	5,323.75	206.44	296.08	42.59
	Median	116.91	1.60	0.67	4.98
	Min	-127,586.84	-821.46	-2,422.01	-1,682.03
Total		442,480,584.65	18,275,617.54	3,822,404.73	650,976.49

**Table 3: Profits in USD made from MEV across each chain.**

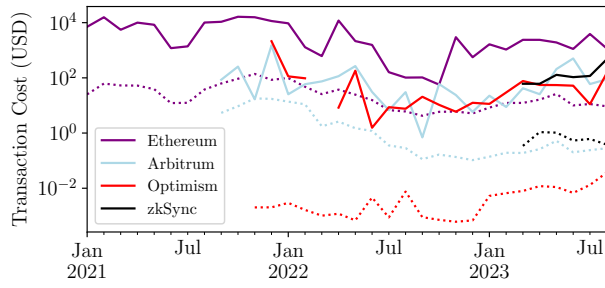
We see similar trends in the top plot of Figure 5. The salient difference between the two plots, though, is that all three of the rollups eventually surpass Ethereum in the number of detected arbitrages. In the month of April 2023, Arbitrum had 7.2× more arbitrages than Ethereum, although the difference narrows in the subsequent months. Again, this is likely related to broader economic trends and the increasing usage of rollups as the transaction fees tend to be much lower. After opening up to developers in February 2023, the first MEV was detected around March 25, 2023 on zkSync, highlighting the rapid adoption of MEV extractors even on ZK-based rollups. Finally, we found no instances of sandwiching on any of the rollup platforms suggesting that the sequencers are not performing sandwich trades. However, in Section 5, we show that cross-layer sandwiching is a viable threat vector to traders on rollups. Cross-layer sandwich attacks can be mounted by normal users and do not require control over pending transactions.

### 4.2 Profits

In Table 3, we see that Ethereum is still the platform with the highest profits, both cumulatively and on average. In fact, the cumulative profits from MEV on Ethereum are close to 10× the profits on the other three platforms combined for both arbitrage and liquidation. This is likely because Ethereum is simply more popular and thus has more pools and more opportunities and more liquidity. Additionally, we note that the mean values are much higher than the median values while the maximum values are much higher than the P<sub>90</sub> values for all platforms—this indicates that a small percentage of transactions account for a disproportionate fraction of the total profits. So while by definition, 90% of transactions are below the P<sub>90</sub> in terms of profits, those that are above the 90th percentile are way above. Among the rollups, Arbitrum has significantly greater arbitrage profits than either Optimism or zkSync. While for liquidations, Optimism has slightly higher profits. We suspect that the arbitrage profits on Arbitrum are higher due to users performing trades with higher amounts as compared to Optimism. We also suspect that liquidation profits are higher on Optimism due to higher collaterals.



**Figure 5: Number of detected arbitrages and liquidations per month on Ethereum, Arbitrum, Optimism, and zkSync.**



**Figure 6: Median monthly transaction cost in USD for extracted MEV on each chain. Dotted lines depict arbitrages, while solid lines depict liquidations.**

### 4.3 Costs

In Figure 6, we present the median monthly transaction costs in USD over our measurement period for each platform. The first insight we can observe is that liquidation transactions are significantly more expensive on all platforms. The one exception being in November 2021, when both arbitrages and liquidations on Arbitrum cost the same. For both arbitrage and liquidation, respectively, we see that transactions on Ethereum are more expensive than any of the rollups. This suggests that these rollup platforms are achieving one aspect of their goal: making transactions cheaper.

### 4.4 Opportunities

Figure 7 depicts the cumulative distribution of the block distance between extracted MEV and MEV opportunities for up to a distance of 100 blocks. We see that for Ethereum, over 50% of the MEV extractions occurred within the same block as the opportunity (i.e., block distance of 0). Interestingly, for zkSync and former releases of Arbitrum and Optimism (i.e., Classic and Pre-Bedrock, respectively) we find that the smallest block distance is 1 (i.e., the MEV extraction never occurred in the same block as the opportunity). However, for newer releases of Arbitrum and Optimism (i.e., Nitro and Post-Bedrock, respectively) we do find a small amount of MEV

extractions that occur within the same block. This is an interesting result since newer releases of Arbitrum and Optimism still do not have a public mempool. Yet, some extractors were able to insert their transactions within the same block as the opportunity. For instance, according to our dataset, address 0x4b9406...5e3021 performed a total of six successful liquidations on Optimism, where all liquidations were performed within the same block as the opportunity, which hints towards a systematic approach. One example is block 108,560,468, which contains 18 transactions and the extractor's transaction was included at position 11 whereas the opportunity is included at position 4. The block was built on August 22, 2023 i.e., towards the end of our measurement period.

### 4.5 Competition

For Ethereum, the largest number of MEV competitors that targeted the same arbitrage opportunity and were successful was 14. Whereas for Arbitrum, Optimism, and zkSync it was 15, 9, and 7, respectively. Similarly, the largest number of MEV competitors that targeted the same liquidation opportunity and were successful was 10, 5, 5, and 0, for Ethereum, Arbitrum, Optimism, and zkSync, respectively. This confirms that there is typically more competition on Ethereum than on rollups regarding MEV extraction. Moreover, we used the public APIs of Etherscan [26], Arbiscan [5], Optimistic Etherscan [27], and zkSync's Block Explorer [75] to obtain a list of transactions performed by each competitor and computed the percentage of reverted transactions across chains. For arbitrage, Ethereum has overall a rate of 8% of reverted transactions among competitors, whereas Arbitrum, Optimism, and zkSync have 39%, 29%, and 38%, respectively. A similar pattern is seen for liquidation, where Ethereum has overall a rate of 17% of reverted transactions, whereas Arbitrum, Optimism, and zkSync have 29%, 27%, and 0% (due to no liquidation competition detected), respectively. This highlights the advantage of proposer-builder separation (PBS) [33] on Ethereum, where builders control which MEV transactions are forwarded to proposers. This allows them to forward only transactions that successfully extract MEV, thus reducing the amount of reverted transactions.



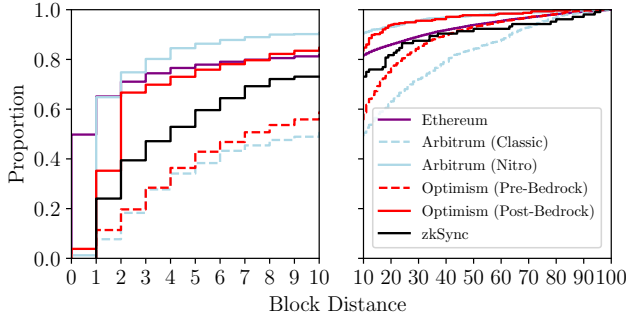


Figure 7: Block distance between extraction and opportunity.

Strategy	Protocol	Ethereum	Arbitrum	Optimism	zkSync
Arbitrage	Aave	0.08%	0.21%	0.03%	0.00%
	Balancer	0.16%	6.50%	13.20%	0.00%
Liquidation	Aave	4.82%	39.78%	44.38%	2.37%
	Balancer	0.82%	9.17%	14.57%	20.62%

Table 4: Number of flash loans across rollups and Ethereum.

#### 4.6 Flash Loans

We compare flash loan usage between rollups and Ethereum in Table 4. As expected, we observe that flash loans are not widely used for arbitrage, but are often used for liquidations. Moreover, we see that flash loans are more popular on rollups than on Ethereum. For example, for liquidations, only 5.64% of the MEV extractors on Ethereum used a flash loan, whereas on Arbitrum, Optimism, and zkSync they are used at rates of 48.88%, 58.95%, and 22.99%, respectively. In addition, we find that Aave is used more often for liquidations on Arbitrum and Optimism, but Balancer is used more often on zkSync. Conversely, for arbitrage, Balancer is used more for arbitrages on all of our measured platforms.

#### 4.7 Code Reuse

We explore whether MEV extractors reuse their smart contract bytecode across rollups and Ethereum. Both Arbitrum and Optimism are EVM compatible, meaning that developers can simply redeploy the same smart contracts on Arbitrum and Optimism which they previously deployed on Ethereum, without having to recompile them. Smart contracts on zkSync on the other hand are not EVM compatible. Hence, developers are required to use a dedicated compiler [74] to translate the EVM bytecode into a format that is executable on zkSync [72]. To that end, we only download the runtime bytecode of smart contract addresses that are the destination (i.e., “to” field) of transactions that we previously identified performing either arbitrage or liquidation on Ethereum, Arbitrum, and Optimism. We ignore any bytecode that is verified on Etherscan and that contains the opcode `DELEGATECALL`. The intuition behind the former is that bots will not have their source code verified on Etherscan, whereas the latter prevents counting proxy smart contracts as bots [45]. Finally, we remove every `PUSH` opcode and its associated data from the bytecode including any hard-coded strings and metadata that

are appended to the end of the bytecode [57] as this data would be case-specific.

For arbitrages, we identified 2,160, 873, and 454 bots on Ethereum, Arbitrum, and Optimism, respectively. While for liquidations, we found 484, 101, and 87 bots on Ethereum, Arbitrum, and Optimism, respectively. Overall, it seems that more users engage in arbitrage than liquidation, on both rollups and Ethereum. We checked whether the exact same bytecode is deployed across the same chain. For arbitrage, we found 27 clusters of identical bytecode on Ethereum, with the largest cluster containing 318 identical contracts. For Arbitrum and Optimism we found 16 different clusters, with the largest cluster containing 11 contracts on Arbitrum and 16 on Optimism. For liquidation, we also found clusters of identical contracts, although less as compared to arbitrage. We found six clusters on Ethereum with the largest one containing 10 identical contracts and two clusters on Arbitrum with the largest one containing three identical contracts. We did not find identical contracts on Optimism. Finally, we checked for identical bytecode across different chains. We found three arbitrage bots and two liquidation bots with identical bytecode deployed across Ethereum, Arbitrum, and Optimism (see Table 5).

### 5 Cross-Layer Sandwich Attacks

In this section, we propose three novel cross-layer sandwich attacks that leverage the fact that users can send L2 transactions via L1. We execute these attacks against an Ethereum testnet—a network that runs the Ethereum protocol, but that carries no value and is only used for experiments and tests. *Even in the testnet, we only target our own transactions.* We also simulate our attacks on the Ethereum mainnet—the full value Ethereum network—these simulations show how much MEV we could have extracted if we had been running this attack in real time.

#### 5.1 Attacker Model

Our model assumes normal blockchain user capabilities (e.g., observing the mempool, sending out transactions, etc.). An attacker can pursue different strategies to conduct cross-layer sandwich attacks, depending on whether they have access to the mempool on L1 and whether they operate accounts on L1 and L2. A user can select transaction fees arbitrarily, but transaction ordering is at the sequencer’s discretion on rollups, with different ordering mechanisms discussed in Table 1. We assume that the sequencer is not colluding with any other users, and that more generally, users can only view transactions in the public mempool.

Strategy	Ethereum	Arbitrum	Optimism
Arbitrage	0xF51Fe2...3b677D	0x570CA2...16edf8	-
	0x1A8f43...72f58E	0x1A8f43...72f58E	0x1A8f43...72f58E
	0x51C7D6...f0C9c9	0x568FBE...85D328	-
	0x143647...5991b5	0x8076E6...A94AFC	-
	0xfa6d80...1EBF90	0xa062B2...B2B84d	0xB4a513...Cac090
Liquidation	0x2a2Cdd...e80C63	-	0xE8Ab13...FBE6c7
	0x81c3B1...4E4aEB	0x8FEe8E...4f745D	0x4540f2...3d0598
	0x373f2b...8Ac495	0x71a073...7d2D82	0x373f2b...8Ac495

Table 5: MEV bots with identical bytecode across chains.

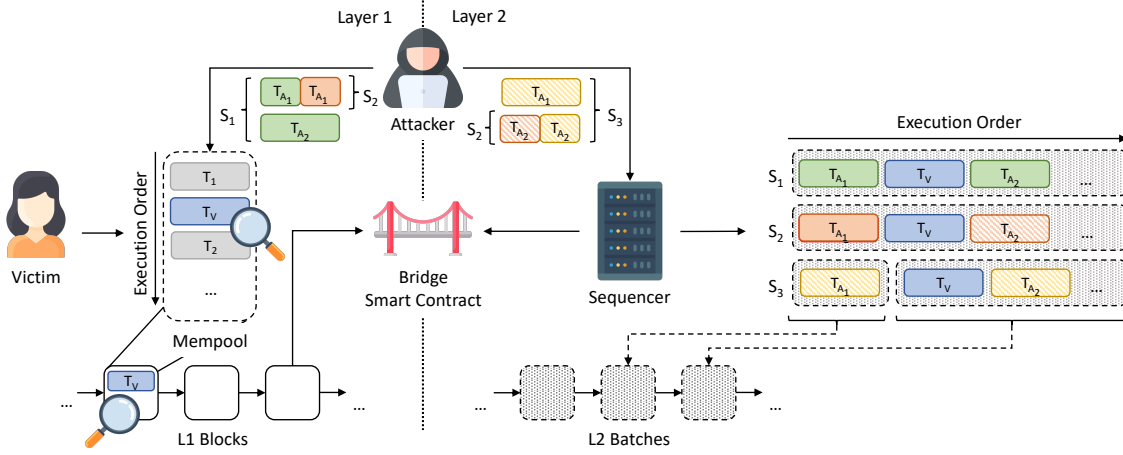


Figure 8: An illustrative example of all three cross-layer sandwich attacks (i.e.,  $S_1$  highlighted in green,  $S_2$  highlighted in orange, and  $S_3$  highlighted in yellow) presented in this work. L1 transactions are solid and L2 transactions are striped.

## 5.2 Attack Strategies

We present the following three cross-layer sandwiching strategies:

**Strategy  $S_1$ : Classical Sandwiching.** In this strategy, the attacker performs a classical sandwich attack on L1. Users can send L2 transactions via L1, which as a result are publicly visible in the mempool before finalization. The attacker exploits the fact that L1 is vulnerable to frontrunning by monitoring the mempool on L1 and searching for transactions that contain L2 transactions that perform token swaps (i.e.,  $T_V$ ). Afterwards, the attacker performs classical sandwiching by bribing the block builder to include transaction  $T_{A1}$  before the observed transaction  $T_V$ , and transaction  $T_{A2}$  after the observed transaction  $T_V$ . Despite all transactions being submitted via L1, the actual effect and the resulting price manipulation only occurs on L2 (see green boxes in Figure 8).

**Strategy  $S_2$ : Hybrid Sandwiching.** In this strategy, the attacker also monitors the mempool on L1 for L2 transactions that perform token swaps (i.e.,  $T_V$ ). However, the attacker only performs transaction  $T_{A1}$  on L1 and sends out transaction  $T_{A2}$  directly to the sequencer on L2. The attacker exploits the fact that sequencers place L2 transactions originating from L1 at the top of L2 batches. This not only provides the attacker with a deterministic backrunning mechanism, but also helps the attacker reducing its costs, as transaction fees are lower on L2 (see orange boxes in Figure 8).

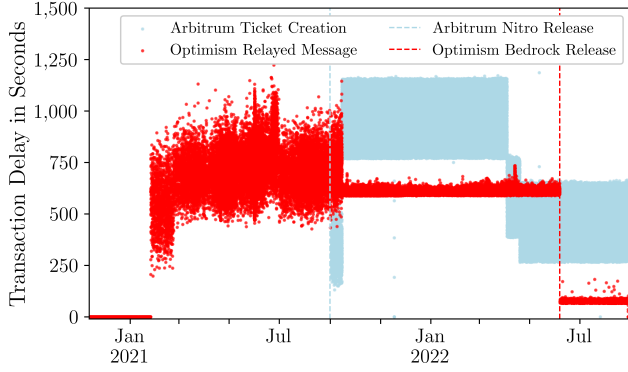
**Strategy  $S_3$ : Speculative Sandwiching.** In this strategy, the attacker does not monitor the mempool on L1, but does monitor the transactions included in the latest blocks on L1 and searches for transactions that contain L2 transactions that perform token swaps (i.e.,  $T_V$ ). The attacker exploits the fact that L2 transactions, which are emitted via L1 transactions, are not immediately included in L2 batches. Instead, the sequencer waits for a timeout period to elapse before extracting the transactions from L1 and including them in an L2 batch. This delay allows the attacker to send the transaction  $T_{A1}$  on L2, wait for  $T_V$  to be included, then send transaction  $T_{A2}$  on L2 as well. Since both attacker transactions are emitted via L2, this enables the attacker to further save on transaction fees as compared to strategy  $S_1$  (see yellow boxes in Figure 8).

## 5.3 Mainnet Simulation

We simulate our three attack strategies using past mainnet data to analyze their feasibility and measure their potential impact.

**5.3.1 Victim Inference.** To simulate the impact of our three attacks we first need to find real potential victim transactions. A potential victim is defined as an L2 transaction that is sent via L1 and which performs a trade on a DEX deployed on L2. To identify such transactions, we start by scanning past mainnet transactions for L1 transactions which encapsulate L2 transactions. By analyzing the bridges deployed by Arbitrum, Optimism, and zkSync, we identified several events that are triggered whenever an L2 transaction is sent via L1. For Arbitrum, an event called “*InboxMessageDelivered*” is emitted, while for Optimism an event called “*TransactionEnqueued*” or “*TransactionDeposited*” is emitted, and for zkSync an event called “*NewPriorityRequest*” is emitted (see Appendix A). We leverage these events to collect L1 transactions that encapsulate L2 transactions.

Next, we need to identify L2 counterpart to the L2 transaction that was sent via L1. This is so that we can determine whether the L2 transaction performs a trade on a DEX on L2. For zkSync, this is trivial as the L1 transaction hash is identical to the L2 transaction hash on zkSync. In the case of Arbitrum and Optimism, this is more complicated as the corresponding L2 transaction has a different transaction hash than the L1 transaction. However, L2 transactions on Arbitrum and Optimism emit specific events when they originate from L1. For Arbitrum, we search for L2 transactions which emitted an event called “*RedeemScheduled*” and for Optimism we search for L2 transactions which emitted an event called “*RelayedMessage*”. As a final step for Arbitrum and Optimism, we need to link the L2 transactions to their corresponding L1 transactions. For Arbitrum we use the *message number*, which is a unique sequential identifier generated via Arbitrum’s bridge smart contract on L1 and which is part of the event data emitted by the L2 transaction. For Optimism, we leverage the *message hash* which is computed as the Keccak hash of the L2 transaction content that was sent via L1.



**Figure 9: Delay in seconds between L1 transactions and L2 transactions on Arbitrum and Optimism.**

Once linked, we scan the L2 transactions for DEX swaps. A swap typically results in two ERC-20 “Transfer” events being emitted. Hence, we scan the events emitted by the identified L2 transactions and mark these as potential victims if we find two “Transfer” events emitted by the same token smart contract, where the sender and receiver of the tokens are swapped. We found a total of 1,916,524 L2 transactions emitted via L1, out of which 1,459,570 were observed on Arbitrum, 424,212 on Optimism, and 32,742 on zkSync. From the observed 1.9M L2 transactions, only 170,674 ( $\approx 9\%$ ) can be considered potential victims based on our methodology to detect token swaps, where 87,921 are from Arbitrum and 82,753 from Optimism. We could not find potential victims on zkSync. This is most likely due to the fact that there was no cross-layer DEX currently deployed on zkSync at the time of our measurement. We observe that essentially all potential victim transactions originate from users using the Hop Protocol [54]. The Hop protocol allows users to send tokens from one rollup or chain to another by employing their own market makers, hence making it a perfect victim for our cross-layer sandwich attacks.

**5.3.2 Transaction Inclusion Delays.** As mentioned earlier, L2 transactions that are sent via L1 are not directly included in L2 batches by the sequencer. The sequencer typically does this only after a certain time has passed. This is to obtain some confirmation on the finality of the observed L1 transactions, as their inclusion might still change early on due to uncle blocks or chain reorganizations (i.e., reorgs). We compared the timestamps of L1 transactions and their corresponding L2 transactions of the identified potential victims to understand the delay that is imposed by the sequencers. This allows us to determine whether attack strategy  $S_3$  is feasible in practice. If the delay is too short or zero, then attackers do not have enough time to react to emitted L2 transactions that they observed via L1 blocks. We computed the delay by subtracting the L1 transaction timestamp from the L2 transaction timestamp of the previously identified victim transactions to see how long it takes for an L1 transaction to be included on L2. Figure 9 depicts our results.

For Arbitrum, we find that the smallest delay (i.e., the fastest inclusion of an L1 transaction) was 0 seconds, whereas the mean is around 798 seconds ( $\approx 13$  minutes) and the median is around

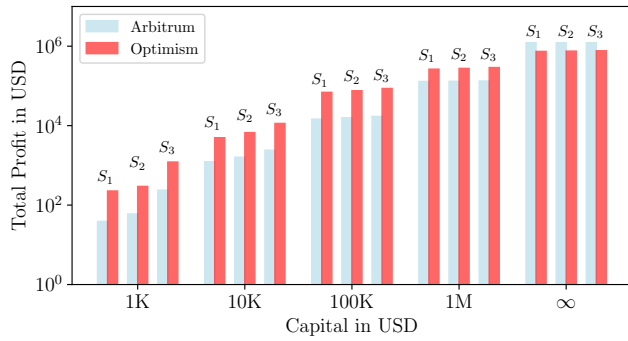
868 seconds ( $\approx 14$  minutes). The maximum that we observed (i.e., the slowest inclusion of an L1 transaction) is 3,694 seconds ( $\approx 1$  hour). In Figure 9, we see that the delay usually stays within some boundaries and that these boundaries have moved over time towards shorter delays. However, we can also observe that a delay of zero seconds is typically an outlier and that almost always the delay is at least 120 seconds (i.e., minutes). This means that, on Arbitrum, attackers typically have a window of at least 2 minutes to send their frontrunning transaction  $T_{A_1}$  after observing the L2 transaction on L1. Theoretically, attackers can have even more time since the delays that we presented here are related to the ticket creation transactions and not the ticket redeem transactions (i.e., which actually execute the victim’s L2 transaction). However, it is up to the user/protocol to send the redeem transactions, after the ticket creation transactions have been included. This means that the smallest possible delay is near zero seconds after the ticket creation. Therefore, the ticket creation transaction delay is a good reference point to measure the delay that an attacker can leverage.

For Optimism, we can compare transaction delays before and after the Bedrock release (i.e., version 2.0). For both releases (i.e., pre- and post-Bedrock), we find that the minimum is zero seconds (i.e., instant inclusion). However, in Figure 9, we observe that delays of zero seconds occurred only in the earlier days of Optimism’s existence and that it changed around February 2022. For the post-Bedrock release the zero second delays seem to be outliers. Pre-Bedrock, the mean and median delays were around 583 seconds ( $\approx 9$  minutes) and 609 seconds ( $\approx 10$  minutes) respectively. While post-Bedrock, the mean and median delays are around 83 seconds and 74 seconds ( $\approx 1$  minute), respectively. The maximum (i.e., largest delay) that we observed pre- and post-Bedrock was 1,266 seconds and 22,296, respectively. Similar to Arbitrum, the delays seem to stay within certain bounds. In contrast to Arbitrum, there is more variation on Optimism, though it decreased over time.

**5.3.3 Estimated Profits.** We calculate potential profits by simulating our three attack strategies using the previously identified victim transactions. For each victim transaction, we extract the amount that is being traded. The attacker’s profit hinges on maximizing its purchase amount within the victim’s slippage threshold without exceeding it. We compute the optimal amount using a ternary search algorithm as suggested by other works [34] and use this amount to simulate and compute the gain of performing a cross-layer sandwich attack against the victim transaction. The final profit is calculated for each attack strategy by deducting the individual

Capital	Arbitrum			Optimism		
	$S_1$	$S_2$	$S_3$	$S_1$	$S_2$	$S_3$
1K	10	17	216	7	29	709
10K	86	126	254	196	302	815
100K	220	237	277	534	643	847
1M	289	295	320	773	845	965
$\infty$	314	322	345	825	898	1,015

**Table 6: Profitable victim transactions per attack strategy and attacker capital.**



**Figure 10: Total profit in USD on Arbitrum and Optimism for each attack strategy across different attacker capitals.**

transaction costs (which depend on the employed attack strategy) from the gained amount. Moreover, since sandwich attacks require the a priori purchase of assets in  $T_{A_1}$ , attackers are required to possess a certain upfront capital. To gauge the profitability across various capital ranges, we consider budgets of 1K USD, 10K USD, 100K USD, 1M USD, and  $\infty$  capital.

Interestingly, not all identified potential victim transactions are profitable. These transactions are either too small to offer revenue outweighing costs, or have low slippage tolerance settings which do not provide enough room for frontrunning. In Table 6, we observe that the number of profitable transactions increases together with the attacker’s capital, suggesting that several victim transactions require large purchases by attackers in order to be profitable. Nonetheless, we can state that the maximum total profit is 1.2M USD for Arbitrum and 767K USD for Optimism. Figure 10 further highlights that Optimism is more profitable than Arbitrum for attackers that possess a capital up to 1M USD. In addition, we observe that attack strategy  $S_3$  is always more profitable than  $S_2$  and  $S_1$ . However, the difference diminishes as the attacker capital increases. Hence, we conclude that either strategies are profitable, but that for attackers with smaller capital (e.g., 1K USD) strategy  $S_3$  is more profitable than for attackers with larger capital (e.g., 1M USD).

#### 5.4 Testnet Validation

While our mainnet simulation provides insights such as number of potential victims and estimated profits, it does not validate the deployability of our attacks in practice. To that end, we leverage live test networks to demonstrate the practicality of our cross-layer sandwich attacks. Given the similarities between testnets and mainnets, we can validate the feasibility of our attacks in an ethical manner without impacting real users and requiring substantial financial resources. For our analysis, we used the Sepolia testnet alongside its corresponding L2 counterparts for Arbitrum, Optimism, and zkSync. We developed a script to automate the process of deploying our own attacker and victim accounts on each chain including our own Uniswap V2-inspired DEXes on Arbitrum, Optimism, and zkSync. *The only victim was our own account even on the testnet.* The script automatically performs the three attack strategies. First, a victim transaction  $T_V$  is sent via the official L1 bridge of the rollup that is under test. The victim transaction simply performs a

swap on the Uniswap-alike DEX that the script previously deployed on L2. Afterwards, the script simulates the attacker transactions  $T_{A_1}$  and  $T_{A_2}$  according to the selected strategy. The script either scans the mempool or the latest blocks on L1 and searches for our victim transaction  $T_V$  as detailed in Section 5.3.1 and crafts the corresponding transactions  $T_{A_1}$  and  $T_{A_2}$ , which essentially buy and sell the same asset as  $T_V$  on the Uniswap-alike DEX on L2. See Appendix B for a detailed list of our successfully deployed attack transactions across Arbitrum, Optimism, and zkSync.

## 6 Discussion

In this section, we discuss the generalization as well as limitations of our cross-layer sandwich attacks and potential countermeasures.

### 6.1 Generalization and Limitations

We performed our measurements on Arbitrum, Optimism, and zkSync, as these were the most popular rollups by market share at the time of writing [38]. However, since then other rollups such as Base [14] and Blast [11] have gained tremendously on popularity. As these rollups are based on Optimism’s OP Stack or Superchain Ecosystem [47], our scripts to detect MEV on Optimism can easily be repurposed to measure MEV extraction on any Optimism fork. While we use sandwich attacks as a case study, our proposed strategies  $S_1$ ,  $S_2$ , and  $S_3$  can be further generalized to cross-layer frontrunning attacks. We posit that our work applies any time there is a victim transaction making L1-to-L2 transactions on a blockchain with a public mempool (strategies  $S_1$  and  $S_2$ ), or when a rollup imposes a delay in including L1 transactions (strategy  $S_3$ ). Sandwich attacks exploit both frontrunning and backrunning, so it is possible that we could see our strategies applied to other techniques besides sandwiching that would also yield profit when transactions are frontrun (e.g., generalized frontrunners [55, 70]). All of our proposed cross-layer sandwich attacks exploit the fact that users submit L2 transactions via L1. While this seems counter-intuitive, there exist legitimate use cases which require such behavior. For example, they are used in cross-chain swaps as provided via the Hop Protocol [54] or the Across Protocol [53]. The reason why we were only able to find victim transactions related to Hop, was due to the fact that Across leverages their own bridges and does not rely on the native bridges deployed by individual rollup technologies. However, Across’s bridge also hinges on emitting events to trigger cross-chain swaps, hence, our victim inference could be adjusted for Across as well. Strategies  $S_1$  and  $S_2$  depend on the fact that attackers can observe victim transactions via the mempool. Hence, any L1 blockchain that has a public mempool is susceptible to these two strategies. Finally, we did not find any victim transactions on zkSync due to the fact that during our data collection period no cross-chain exchange such as Hop was operating.

### 6.2 Countermeasures

Private pools [28], including encrypted mempools [36], provide an effective countermeasure against strategies  $S_1$  and  $S_2$  since attackers are not able to observe victim transactions via the mempool anymore and thus cannot frontrun them. However, strategy  $S_3$  still remains feasible as it does not rely on victim transactions being visible before a block is finalized on L1. Strategy  $S_3$  exploits the fact



that there is a delay between L1 and L2 transactions. Rollups could try to reduce the delay almost to zero, such that attackers cannot react fast enough. However, this would make rollups vulnerable to time bandit attacks or uncle bandit attacks, which are already viable issues on Ethereum [32]. Another solution could be randomized ordering of transactions. However, this would break the first-come, first-served ordering policy of many rollups and would not entirely solve the issue of sandwich attacks as shown by previous works [60]. Finally, a number of works aim to counter sandwiching on AMMs by proposing new frontrunning resistant AMM protocols [35, 62, 71]. However, these protocols were exclusively designed to protect AMMs against frontrunning and not any other protocols. Moreover, it is not clear whether these protocols are still applicable in the context of cross-layer sandwiching.

## 7 Related Work

Gudgeon et al. [30] presented the first SoK on Layer-2, however, their discussion was before the advent of rollups. The first academic work on MEV was done by Daian et al. [21]. Torres et al. [59] and Qin et al. [56] were the first to measure the historical prevalence of MEV on the blockchain. Piet et al. [50] analyzed private pool usage by measuring mempool transactions from several vantage points, but did not study MEV. Weintraub et al. [65] used related methods to measure the prevalence of MEV in private pools especially on Flashbots [28]. While these were important works for identifying the widespread phenomenon of MEV, their measurements were before the advent of rollups and, hence, only focused on Ethereum.

The first work to study MEV on rollups was by Ha et al. [31], which measured MEV on Optimism and Polygon. These measurements, however, persisted for only a couple months and did not consider Arbitrum or any ZK rollups which had not been released yet. Also at the intersection of MEV and rollups, Bagourd and Francois [8] scanned Polygon, Optimism, and Arbitrum for MEV. However, they did not scan for sandwiches, or analyze the negative aspects of MEV, nor did they compare their results to Ethereum. Besides providing results for a longer measurement period, our methodology has found more instances of MEV for the same time period than Bagourd and Francois [8] reported. Recently, Öz et al. [48] have studied MEV extraction on Algorand, which follows a first-come, first-served policy similar to rollups.

In the context of cross-chain MEV, Obadia et al. [44] contributed a formalization of cross-domain MEV, but their definitions do not directly yield any method for quickly finding new instances of cross-chain MEV. In a similar vein, Zhang et al. [69] defined cross-shard frontrunning and contributed a model for defending against it. These, however, are orthogonal to the frontrunning issue in rollups.

McLaughlin et al. [42] present a different technique to detect arbitrages as compared to existing works that is more application agnostic. Li et al. [41] characterize several novel forms of MEV and measure their appearance within Flashbots bundles. Likewise, Babel et al. [7] leverage machine learning to learn new strategies to extract MEV—their model is capable of detecting two new strategies. However, neither of the aforementioned works considers MEV in the context of rollups. Moreover, besides comparing MEV extraction between Ethereum and rollups, we also present three novel strategies that enable users to perform sandwich attacks on rollups.

## 8 Conclusion

MEV is omnipresent, even on rollups. We analyzed the extraction of MEV across Ethereum, Arbitrum, Optimism, and zkSync for a period of nearly three years. We found a significant amount of arbitrages and liquidations occurring on rollups. Compared to Ethereum, and despite similar volume and lower costs, MEV on rollups yield smaller profits. The absence of proposer-builder separation in rollups reflects in the frequency of failed transactions by competitors. Moreover, MEV opportunities take longer to materialize on rollups as compared to Ethereum. Nonetheless, we found instances where extraction and opportunity coincided on rollups, hinting at potential for shorter extraction delays. While no traditional sandwiching was uncovered on rollups, we detected three potential cross-layer sandwich attacks. These exploit L2 transactions routed via L1 to execute sandwiching on rollups, sidestepping the need for a public mempool on rollups as it is required for traditional sandwiching. Assessing these attacks, we estimated potential profits nearing 2 million USD. This underscores the necessity for further research to comprehend MEV's impact on cross-chain communication.

## Acknowledgments

This work was supported by the Zurich Information Security & Privacy Center (ZISC).

## References

- [1] Aave. 2024. Aave - Open Source Liquidity Protocol. <https://aave.com/> Online; accessed 20 April 2024.
- [2] Aave. 2024. Flash Loans - Developers. <https://docs.aave.com/developers/guides/flash-loans> Online; accessed 20 April 2024.
- [3] Aave. 2024. LendingPool - Developers. <https://docs.aave.com/developers/v/2.0/the-core-protocol/lendingpool#getuseraccountdata> Online; accessed 20 April 2024.
- [4] Aave. 2024. Liquidations - Developers. <https://docs.aave.com/developers/guides/liquidations> Online; accessed 20 April 2024.
- [5] Arbiscan. 2024. Accounts | Arbiscan. <https://docs.arbiscan.io/api-endpoints/accounts> Online; accessed 20 April 2024.
- [6] Arbitrum. 2024. Arbitrum — The Future of Ethereum. <https://arbitrum.io/> Online; accessed 20 April 2024.
- [7] Kushal Babel, Mojan Javaheripi, Yan Ji, Mahimna Kelkar, Farinaz Koushanfar, and Ari Juels. 2023. Lanturn: Measuring economic security of smart contracts through adaptive learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1212–1226.
- [8] Arthur Bagourd and Luca Georges Francois. 2023. Quantifying MEV On Layer 2 Networks. *CoRR* (2023). <https://doi.org/10.48550/arXiv.2309.00629>
- [9] Balancer. 2024. Balancer DeFi Liquidity Protocol. <https://balancer.fi/> Online; accessed 20 April 2024.
- [10] Balancer. 2024. Flash Loans | Balancer. <https://docs.balancer.fi/reference/contracts/flash-loans.html> Online; accessed 20 April 2024.
- [11] Blast. 2024. About Blast - Blast Developer Documentation. <https://docs.blast.io/about-blast> Online; accessed 5 September 2024.
- [12] Ronin Chain. 2024. Ronin Bridge. <https://docs.roninchain.com/apps/ronin-bridge> Online; accessed 20 April 2024.
- [13] Chainlink. 2024. Chainlink Data Feeds | Chainlink Documentation. <https://docs.chain.link/data-feeds#price-feeds> Online; accessed 20 April 2024.
- [14] Coinbase. 2024. Base. <https://www.base.org/> Online; accessed 5 September 2024.
- [15] CoinGecko. 2024. Crypto API Documentation | CoinGecko. <https://www.coingecko.com/api/documentation> Online; accessed 20 April 2024.
- [16] CompaniesMarketCap.com. 2024. Companies ranked by Market Cap - CompaniesMarketCap.com. <https://companiesmarketcap.com/> Online; accessed 20 April 2024.
- [17] CompaniesMarketCap.com. 2024. Total Value Locked All Chains - DeFiLlama. <https://defillama.com/chains> Online; accessed 20 April 2024.
- [18] Compound. 2024. Compound. <https://compound.finance/> Online; accessed 20 April 2024.



- [19] Compound. 2024. Compound V2 Docs | Comptroller. <https://docs.compound.finance/v2/comptroller/> Online; accessed 20 April 2024.
- [20] Curve. 2024. Curve: Swap. <https://curve.fi> Online; accessed 20 April 2024.
- [21] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. 910–927.
- [22] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. 2019. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 international conference on management of data*. 123–140.
- [23] DappRadar. 2024. Top Ethereum DeFi TVL. [https://dappradar.com/rankings/defi/chain/ethereum?category=defi\\_dex](https://dappradar.com/rankings/defi/chain/ethereum?category=defi_dex) Online; accessed 20 April 2024.
- [24] DeFiLlama. 2024. DeFiLlama - DeFi Dashboard. <https://defillama.com/> Online; accessed 20 April 2024.
- [25] DeFiLlama. 2024. Lending TVL Rankings. <https://defillama.com/protocols/Lending/Ethereum> Online; accessed 20 April 2024.
- [26] Etherscan. 2024. Accounts | Etherscan. <https://docs.etherscan.io/api-endpoints/accounts> Online; accessed 20 April 2024.
- [27] Optimistic Etherscan. 2024. Accounts | Optimism Etherscan | Optimism. <https://docs.optimism.etherscan.io/api-endpoints/accounts> Online; accessed 20 April 2024.
- [28] Flashbots. 2024. Flashbots. <https://www.flashbots.net> Online; accessed 20 April 2024.
- [29] Flashbots. 2024. Flashbots Blocks API. <https://blocks.flashbots.net> Online; accessed 20 April 2024.
- [30] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. 2020. Sok: Layer-two blockchain protocols. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 201–226.
- [31] Huy Ha, Vasiliki Vlachou, Quintus Kilbourn, and Cesare De Michellis. 2021. Flashbabies: Mev on l2. <https://timroughgarden.github.io/fob21/reports/r11.pdf>
- [32] Elan Halpern. 2021. Unmasking the Ethereum Uncle Bandit. <https://medium.com/alchemy-api/unmasking-the-ethereum-uncle-bandit-a2b3eb694019> Online; accessed 20 April 2024.
- [33] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. 2023. Ethereum's Proposer-Builder Separation: Promises and Realities. In *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC 2023, Montreal, QC, Canada, October 24–26, 2023*, Marie-José Montpetit, Aris Leivadreas, Steve Uhlig, and Mobin Javed (Eds.). ACM, 406–420.
- [34] Lioba Heimbach and Roger Wattenhofer. 2022. Eliminating Sandwich Attacks with the Help of Game Theory. In *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (Eds.). ACM, 153–167.
- [35] Lioba Heimbach and Roger Wattenhofer. 2022. Eliminating Sandwich Attacks with the Help of Game Theory. In *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (Eds.). ACM, 153–167.
- [36] Alireza Kavousi, Duc V Le, Philipp Jovanovic, and George Danezis. 2023. Blind-perm: Efficient mev mitigation with an encrypted mempool and permutation. *Cryptology ePrint Archive* (2023).
- [37] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 583–598.
- [38] L2BEAT. 2024. L2BEAT – The state of the layer two ecosystem. <https://l2beat.com/scaling/summary> Online; accessed 20 April 2024.
- [39] L2Fees.info. 2024. L2 Fees. <https://l2fees.info/> Online; accessed 20 April 2024.
- [40] Michael Lewis. 2014. *Flash Boys*. W.W. Norton & Company.
- [41] Zihao Li, Jianfeng Li, Zheyuan He, Xiapu Luo, Ting Wang, Xiaozhe Ni, Wenwu Yang, Xi Chen, and Ting Chen. 2023. Demystifying DeFi MEV Activities in Flashbots Bundle. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 165–179.
- [42] Robert McLaughlin, Christopher Kruegel, and Giovanni Vigna. 2023. A Large Scale Study of the Ethereum Arbitrage Ecosystem. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9–11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 3295–3312.
- [43] Raiden Network. 2024. Raiden Network. <https://raiden.network/> Online; accessed 20 April 2024.
- [44] Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. 2021. Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value. *CoRR abs/2112.01472* (2021). arXiv:2112.01472 <https://arxiv.org/abs/2112.01472>
- [45] OpenZeppelin. 2024. Proxy Upgrade Pattern - OpenZeppelin Docs. <https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies> Online; accessed 20 April 2024.
- [46] Optimism. 2024. Optimism | Home. <https://www.optimism.io/> Online; accessed 20 April 2024.
- [47] Optimism. 2024. Superchain Ecosystem. <https://www.superchain.eco> Online; accessed 5 September 2024.
- [48] Burak Öz, Jonas Gebele, Parshant Singh, Filip Rezabek, and Florian Matthes. 2024. Playing the MEV Game on a First-Come-First-Served Blockchain. *arXiv preprint arXiv:2401.07992* (2024).
- [49] Perun. 2024. Perun | Blockchains in real-time. <https://perun.network/> Online; accessed 20 April 2024.
- [50] Julien Piet, Jaiden Fairoze, and Nicholas Weaver. 2022. Extracting God! [sic] from the Salt Mines: Ethereum Miners Extracting Value. *arXiv:2203.15930 [cs]* (March 2022). <http://arxiv.org/abs/2203.15930> arXiv: 2203.15930.
- [51] Polygon. 2024. Web3, Aggregated. <https://polygon.technology/> Online; accessed 20 April 2024.
- [52] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments.
- [53] Across Protocol. 2024. Home | Across Protocol. <https://across.to/> Online; accessed 20 April 2024.
- [54] Hop Protocol. 2024. A Short Explainer | User Docs | Hop Docs. <https://docs.hop.exchange/basics/a-short-explainer> Online; accessed 20 April 2024.
- [55] Kaihua Qin, Stefanos Chaliasos, Liyi Zhou, Benjamin Livshits, Dawn Song, and Arthur Gervais. 2023. The Blockchain Imitation Game. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9–11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 3961–3978.
- [56] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying Blockchain Extractable Value: How dark is the forest?. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22–26, 2022*. IEEE, 198–214.
- [57] Solidity. 2024. Contract Metadata - Solidity 0.8.26 documentation. <https://docs.soliditylang.org/en/latest/metadata.html> Online; accessed 20 April 2024.
- [58] SushiSwap. 2024. Buy and Sell Instantly on Sushi. <https://www.sushi.com/> Online; accessed 20 April 2024.
- [59] Christof Ferreira Torres, Ramiro Camino, and Radu State. 2021. Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain. In *USENIX Security Symposium, Virtual 11–13 August 2021*.
- [60] Vytautas Tumas, Beltran Borja Fiz Pontiveros, Christof Ferreira Torres, and Radu State. 2023. A Ripple for Change: Analysis of Frontrunning in the XRP Ledger. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–9.
- [61] Uniswap. 2024. Uniswap Protocol. <https://uniswap.org/> Online; accessed 20 April 2024.
- [62] Sarisht Wadhwa, Luca Zanolini, Francesco D'Amato, Aditya Asgaonkar, Chengrui Fang, Fan Zhang, and Kartik Nayak. 2023. Data Independent Order Policy Enforcement: Limitations and Solutions. *Cryptology ePrint Archive* (2023).
- [63] Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. 2021. Towards a step to understand flash loan and its applications in defi ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. 23–28.
- [64] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. 2022. Cyclic arbitrage in decentralized exchanges. In *Companion Proceedings of the Web Conference 2022*. 12–19.
- [65] Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. 2022. A Flash(bot) in the Pan: Measuring Maximal Extractable Value in Private Pools. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC '22)*. Association for Computing Machinery, Nice, France.
- [66] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.
- [67] Wormhole. 2024. The best way to build cross-chain. <https://wormhole.com/> Online; accessed 20 April 2024.
- [68] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J Andrew Zhang, and Ren Ping Liu. 2020. Survey: Sharding in blockchains. *IEEE Access* 8 (2020), 14155–14181.
- [69] Jianting Zhang, Wuhui Chen, Sifu Luo, Tiantian Gong, Zicong Hong, and Aniket Kate. 2023. Front-running Attack in Distributed Sharded Ledgers and Fair Cross-shard Consensus. *arXiv preprint arXiv:2306.06299* (2023).
- [70] Zhuo Zhang, Zhiqiang Lin, Marcelo Morales, Xiangyu Zhang, and Kaiyuan Zhang. 2023. Your Exploit is Mine: Instantly Synthesizing Counterattack Smart Contract. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9–11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 1757–1774.
- [71] Liyi Zhou, Kaihua Qin, and Arthur Gervais. 2021. A2MM: Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges. *CoRR abs/2106.07371* (2021). arXiv:2106.07371 <https://arxiv.org/abs/2106.07371>
- [72] zkSync. 2024. zkEVM FaQ | zkSync Documentaion. <https://docs.zksync.io/zkevm/> Online; accessed 20 April 2024.
- [73] zkSync. 2024. zkSync | Scaling the Ethos and technology of Ethereum. <https://zksync.io/> Online; accessed 20 April 2024.
- [74] zkSync. 2024. zkSync Era Developer Tools | Compiler Toolchain | Overview. <https://era.zksync.io/docs/tools/compiler-toolchain/overview.html> Online; accessed 20 April 2024.
- [75] zkSync Era Explorer. 2024. ZkSync Block Explorer API. <https://block-explorer-api.mainnet.zksync.io/docs> Online; accessed 20 April 2024.

Category	Protocol/Chain	Event Name	Event Topic Hash
Arbitrage	Uniswap V2	<i>Swap</i>	0xd78ad95fa46c994b6551d0da85fc275fe613ce37657fb8d5e3d130840159d822
	Uniswap V3	<i>Swap</i>	0xc42079f94a6350d7e6235f29174924f928cc2ac818eb64fed8004e115fbcca67
	Balancer V1	<i>LOG_SWAP</i>	0x908fb5ee8f16c6bc9bc3690973819f32a4d4b10188134543c88706e0e1d43378
	Balancer V2	<i>Swap</i>	0x2170c741c41531aec20e7c107c24eecd15e69c9bb0a8dd37b1840b9e0b207b
	Curve	<i>TokenExchangeUnderlying</i>	0xd013ca23e77a65003c2c659c5442c00c805371b7fc1ebd4c206c41d1536bd90b
	Curve	<i>TokenExchange</i>	0x8b3e96f2b889fa771c53c981b40daf005f63f637f1869f707052d15a3dd97140
Liquidations	Aave V1	<i>LiquidationCall</i>	0x56864757fd5b1fc9f38f5f3a981cd8ae512ce41b902cf73fc506ee369c6bc237
	Aave V2/V3	<i>LiquidationCall</i>	0xe413a321e8681d831f4dbccbca790d2952b56f977908e45be37335533e005286
	Compound V2	<i>LiquidateBorrow</i>	0x298637f684da70674f26509b10f07ec2fbc77a335ab1e7d6215a4b2484d8bb52
	Compound	<i>Redeem</i>	0xe5b754fb1abb7f01b499791d0b820ae3b6af3424ac1c59768edb53f4ec31a929
	Compound	<i>Redeem</i>	0xe02f6383e19e87c24e0c03e2cd5dbd05156cb29a1b0f3dbca1fa3430e444f63d
Sandwiches	ERC-20	<i>Transfer</i>	0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
Oracle Updates	Chainlink	<i>AnswerUpdated</i>	0x0559884fd3a460db3073b7fc896cc77986f16e378210ded43186175bf646fc5f
Flash Loans	Aave V1	<i>FlashLoan</i>	0x5b8f46461c1dd69fb968f1a003acee221ea3e19540e350233b612ddb43433b55
	Aave V2	<i>FlashLoan</i>	0x631042c832b07452973831137f2d73e395028b44b250dedc5abb0ee766e168ac
	Aave V3	<i>FlashLoan</i>	0xefefaba5e921573100900a3ad9cf29f222d995fb3b6045797eaea7521bd8d6f0
	Balancer	<i>FlashLoan</i>	0xd7d75e01ab95780d3cd1c8ec0dd6c2ce19e3a20427eecd8bf53283b6fb8e95f0
L1 Messages	Arbitrum	<i>InboxMessageDelivered</i>	0xff64905f73a67fb594e0f940a8075a860db489ad991e032f48c81123eb52d60b
	Optimism	<i>TransactionEnqueued</i>	0x4b388aecf9fa6cc92253704e5975a6129a4f735bdbd99567df4ed0094ee4ceb5
	Optimism	<i>TransactionDeposited</i>	0xb3813568d9991fc951961fcb4c784893574240a28925604d09fc577c55bb7c32
	zkSync	<i>NewPriorityRequest</i>	0x4531cd5795773d7101c17bdeb9f5ab7f47d7056017506f937083be5d6e77a382
L2 Messages	Arbitrum	<i>RedeemScheduled</i>	0x5ccd009502509cf28762c67858994d85b163bb6e451f5e9df7c5e18c9c2e123e
	Optimism	<i>RelayedMessage</i>	0x4641df4a962071e12719d8c8c8e5ac7fc4d97b927346a3d7a335b1f7517e133c
Victim Inference	ERC-20	<i>Transfer</i>	0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
	StableSwap	<i>TokenSwap</i>	0xc6c1e0630dbe9130cc068028486c0d118ddcea348550819defd5cb8c257f8a38
	Uniswap V3	<i>Swap</i>	0xc42079f94a6350d7e6235f29174924f928cc2ac818eb64fed8004e115fbcca67

Table 7: Overview of events used in this work to perform measurements across chains and DeFi protocols.

Attack Strategy $S_1$				Attack Strategy $S_2$			Attack Strategy $S_3$			
		Arbitrum	Optimism	zkSync	Arbitrum	Optimism	zkSync	Arbitrum	Optimism	zkSync
$T_{A_1}$	L1	0x7237...561d	0x0c96...e38d	0xd27a...3148	0x3c26...22cb	0x7800...8182	0xe561...f522	-	-	-
	L2	0xe5d7...8c28	0x2f26...c4f8	0xefe7...4ed3	0xb5bb...fc1a	0x059d...f362	0x7275...7e92	0xc9c0...3321	0x4158...6c16	0xb53c...b39a
$T_V$	L1	0x875f...ab89	0x3bb4...1dda	0xae77...3ba8	0x5548...e448	0x4bda...f0e0	0xba2f...c173	0x6b10...e828	0xe843...219d	0x0159...b270
	L2	0x7426...48b5	0x0169...b82c	0xcc7c...e78e	0x9858...8841	0x9fea...d918	0xc766...5ab6	0xed70...2854	0x69cd...74e8	0x70bf...42ec
$T_{A_2}$	L1	0x35e8...43e0	0x0578...5e73	0x6e0a...beb3	-	-	-	-	-	-
	L2	0x31fa...5962	0x595a...3c0b	0xd8ae...851b	0xa6c6...ae00	0x8ebf...0394	0x0801...d346	0xae2d...039d	0xf24d...359c	0x86e6...2023

Table 8: Sepolia testnet transactions for each cross-layer sandwich attack strategy across Arbitrum, Optimism, and zkSync.

## A List of Used Events

Table 7 provides an overview of all the events that have been used in this paper to extract past information from the individual chains. Event topic hashes are grouped by individual categories.

## B Testnet Transactions

Table 8 provides an overview of successfully deployed transactions on the Sepolia testnet for the three attack strategies across Arbitrum, Optimism, and zkSync. Attacker transactions are highlighted in red, whereas victim transactions are highlighted in blue.