No Place to Hide: An Analysis on Protected Order Flow Sandwich Attacks

Lioba Heimbach
Category Labs
liobaheimbach.cs@gmail.com

Ozan Solmaz
ETH Zurich
osolmaz@ethz.ch

Burak Öz
Flashbots
burak@flashbots.net

Christof Ferreira Torres
INESC-ID & Instituto Superior Técnico (IST), University of Lisbon
christof.torres@tecnico.ulisboa.pt

Abstract

Front-running has long plagued Ethereum's public mempool, earning it the nickname of a "dark forest", where predators lurk for profitable transactions. In response, Ethereum and other blockchain ecosystems increasingly rely on private RPCs and native protections to shield transactions from adversaries, which we refer to as protected order flow. Yet the effectiveness of these mechanisms in preventing front-running, and what trust assumptions they entail, remain poorly understood.

In this work, we conduct the first longitudinal, three-year measurement study of sandwich attacks against protected order flow across six blockchains: Ethereum, Solana, Tron, Base, Arbitrum, and Monad. We introduce detection heuristics that capture wide attacks, both within and across blocks, and filter on bot behavior to distinguish sandwiches from legitimate trading activity. We identify 28.0 million sandwich attacks on Solana, 38,567 on Tron, 30,607 on Ethereum, and 1,889 on Base against transactions intended to be protected from front-running. Reorged blocks expose a further 2,875 Ethereum victims. Unlike conventional public-mempool sandwiches, these attacks rarely occur tightly around their victims and, outside Solana, are carried out by a small number of entities.

Our analysis uncovers exposures at every layer: validator- and application-level exposure on Solana, order-flow auctions and reorged blocks on Ethereum, first-come-first-served ordering that fails to prevent latency-based front-running on Tron, and both an RPC bug that exposes pending transactions and predictable victim behavior on Base. These findings show that existing front-running protections can provide substantially weaker guarantees than users expect, highlighting the need for stronger end-to-end defenses against sandwich attacks.

1 Introduction

Blockchains initially propagated transactions through public peer-to-peer (P2P) networks, leaving their ordering to block proposers. Bitcoin adopted this model at launch in 2009, followed by Ethereum in 2015. While transaction ordering mattered little for their original payment use cases, the rise of DeFi transformed public mempools into a *dark forest* of front-running bots, with sandwich attacks becoming particularly prominent on Ethereum around 2020 [25, 26]. In response, blockchain ecosystem has gradually introduced mechanisms to protect users, ranging from private order flow that bypasses Ethereum's public mempool to alternative transaction-submission and ordering designs.

On Ethereum, private RPCs bypass the public mempool and relay transactions directly to block builders, limiting their exposure to front-running. Solana and Monad avoid a public mempool altogether, instead forwarding transactions to upcoming proposers through local mempools. While not designed specifically for sandwich protection, this design substantially reduces pre-inclusion transaction visibility. Tron retains a public mempool but orders transactions by the time the proposer first observes them, claiming to prevent around 98% of front-running [95]. Finally, L2s such as Base and Arbitrum use private mempools in which transactions are visible only to the centralized sequencer until inclusion.

Users therefore increasingly rely on mechanisms that are intended to limit sandwich attacks, yet their effectiveness remains largely unexplored.

In this work, we aim to bridge this gap. We study sandwich attacks targeting transactions protected by private submission or other mechanisms described above, which we term *protected order flow sandwich attacks*. We use *protected order flow* as an umbrella term for transactions subject to a submission or ordering mechanism intended to reduce front-running relative to unrestricted public-mempool ordering. Our three-year longitudinal measurement study across six chains—Ethereum, Solana, Tron, Base, Arbitrum, and Monad— finds persistent sandwich attacks on all but Arbitrum and Monad.

For each chain, we investigate how these protections fail. A transaction traverses multiple stages before inclusion: (1) the user initiates a trade, (2) an application such as a wallet, trading bot, or DApp constructs and signs the transaction, (3) a submission channel, typically an RPC provider, relays it to the block producer, and (4) the producer orders it into a block. Figure 1 illustrates this pipeline, the submission designs of

the six chains, and the stage at which we identify exposures.

On Ethereum, private transactions may pass through Order-Flow Auctions (OFAs), which reveal transaction details to searchers bidding to back-run trades for arbitrage. We identify exposures through which OFA information can become sufficient for sandwiching. Reorgs provide a second exposure channel: we identify 2,875 private transactions exposed through reorgs; enabling 2,576 additional sandwiches.

On Solana, transactions are shared with multiple validators, including upcoming proposers. In the first half of our dataset, a subset of validators accounts for a disproportionate share of sandwich attacks, providing evidence of validator-level exposure. From 2025 onward, this signal weakens, while victim concentration shifts toward specific applications, pointing to the application layer as the dominant exposure source.

On Tron, transactions enter a public mempool, while first-come-first-served ordering is intended to prevent a later transaction from overtaking an earlier one. The attacks we observe are consistent with bots exploiting latency races: upon observing a victim transaction, an attacker races its front-run to the block producer despite the ordering protection.

On Base, we identify two distinct exposures. One bot's activity coincides with a documented RPC-level incident that leaked pending transactions, while other attacks exploit highly predictable victim behavior and therefore do not require pending transaction visibility. More broadly, unlike conventional public-mempool sandwiches, protected order flow attacks typically span wider positions or separate blocks, consistent with attackers operating under more limited information.

Our Contributions. We summarize our main contributions:

- To the best of our knowledge, we provide the first longitudinal measurement of sandwich attacks across six chains with heterogeneous front-running protection mechanisms.
- We introduce novel heuristics to detect non-tight sandwich attacks, including attacks spanning multiple blocks, while filtering bot behavior to distinguish sandwiches from legitimate trading activity.
- We conduct a three-year measurement study across six chains, identifying 28.0M protected order flow sandwiches on Solana, 38,567 on Tron, 30,607 on Ethereum, and 1,889 on Base, yielding attackers over $346M net profit. Reorgs expose a further 2,576 sandwiches on Ethereum.
- We trace these attacks to exposures at OFAs and reorgs on Ethereum, first-come-first-served ordering in a public mempool on Tron, validators and applications on Solana, and RPC exposure and predictable behavior on Base.

2 Background

2.1 Sandwich Attacks

We define a sandwich attack as one or more front-running transactions that trade in the same direction as the targets and execute before them, followed by one or more back-running transactions that reverse the position. The front-runs move the pool price against the targets, while the back-runs realize the profit. We distinguish four types of sandwich attacks. **Tight** sandwiches place the front- and back-running transactions immediately around a single target transaction within the same block. **Within-block** sandwiches also occur entirely within a single block and preserve the front-run–target–back-run ordering, but have at least one intervening transaction. **Cross-block** sandwiches preserve this ordering but span multiple blocks. Finally, **evasive** sandwiches split front- or back-runs across multiple trades or distribute the attack across separate accounts linked by a token transfer. At scale, we solely observe this pattern on Solana (cf. Appendix F).

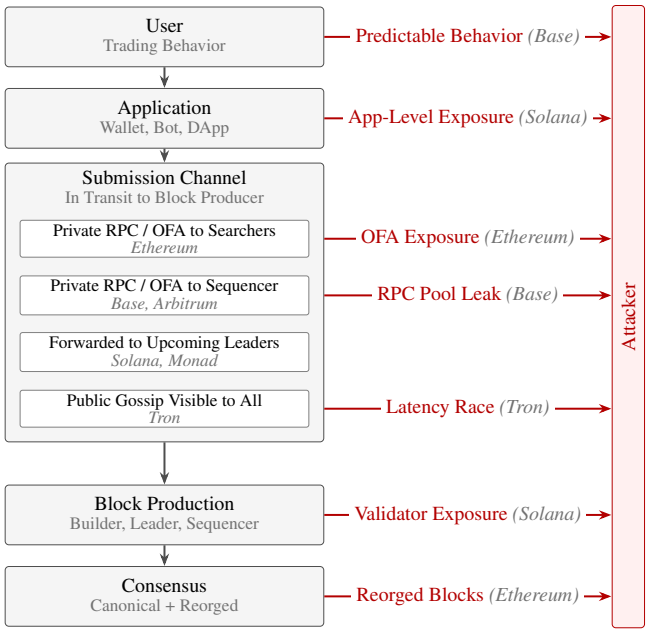


Figure 1: Protected order flow transaction paths and identified exposure points. The submission-channel covers the four transaction-submission designs across the six chains we study. Red arrow marks an exposure we document.

2.2 Front-Running Protections

Table 1 summarizes the studied front-running protections.

Private Submission (Ethereum). Users can bypass the public mempool via a private RPC or an OFA, so transactions reach builders without being publicly gossiped. OFAs additionally expose transaction information to searchers for back-running and return part of the resulting value to users, combining front-running protection with rebates.

Centralized Sequencers (Base, Arbitrum). A single sequencer orders transactions and its input queue is not externally observable [75, 77], leaving no mempool to observe.

Local Mempools (Solana, Monad). Clients forward transac-

Mechanism	Mempool	Chain	Trusted Party
Private RPC	Private	Ethereum	Builders
OFA	Private	Ethereum	OFA Operator, Searchers
Private RPC	Private	Arbitrum, Base	Sequencer
Private RPC	Local	Monad, Solana	Validators/Leaders
FCFS	Public	Tron	Ordering

Table 1: Overview of front-running protection mechanisms.

tions directly to validators/leaders of upcoming slots [72, 73], rather than broadcasting them network-wide. Although this design primarily targets performance [107], it nevertheless offers greater front-running protection by reducing the MEV surface area and narrowing searchers' extraction window [91].

First-Come-First-Served Ordering (Tron). Transactions enter a public mempool and are included in the block by arrival order [100], preventing attackers from purchasing priority through a fee-based auction, as on Ethereum.

2.3 Order-Flow Auctions on Ethereum

OFAs protect users from front-running by privately routing transactions to searchers, who bid to back-run them. Profitable back-runs return part of the proceeds to the user; otherwise, the transaction proceeds as a regular private transaction. They reject harmful bids, such as front-running and sandwiching, regardless of their value [37, 70], and forward the signed user transaction with the highest harmless bid to builders. OFAs differ in the amount and type of transaction information disclosed to searchers and in searcher access to transaction feeds (cf. Table 2). We consider four Ethereum OFAs identified by a recent benchmarking study [52]:

MEV-Share [37]. Users control transaction information disclosed to searchers through configurable hints, ranging from partial to full transaction data. A distinguishing feature is its use of *double-hashing* [38], which applies a second Keccak-256 hash to the transaction hash to preserve user anonymity.

MEVBlocker [70]. Exposes transaction information alongside synthetic or decoy transactions. In swap transactions, it additionally removes sensitive parameters, such as slippage tolerance, to mitigate sandwich attacks. Furthermore, it does not share the transaction with searchers if the likelihood of a back-run is low. Order flow access is controlled through RPC configuration [70], with transactions distributed across permissionless and curated searcher streams via the `shareAll` and `shareSafe` flags. Decoy transactions obscure genuine user transactions, reducing the risk of identification and subsequent sandwiching through alternative channels.

Blink and Merkle [12]. Restricts order flow access to approved searchers who must complete an on-boarding process. Approved searchers receive full transaction data, making the on-boarding process the primary protection mechanism.

OFA	Entity	Feed Access	Protection
MEV-Share [37]	Flashbots	Public	Hints
MEVBlocker [70]	CoW Protocol/SMG	Public/Private	Obfuscation
Blink [12]	Blink Labs Ltd.	Private	Whitelisting
Merkle*	Blink Labs Ltd.	Private	Whitelisting

*No longer operates as an independent entity. Merged with Blink in 2026.

Table 2: Overview of studied OFA solutions on Ethereum.

3 Data Collection

We collect data for six of the ten largest chains by Total Value Locked (TVL) as of 17 August 2026 [27]: Ethereum, Solana, Tron, Base, Arbitrum, and Monad. We select those chains as they either natively mitigate front-running, through private mempools for instance, or expose which transactions received protection. Our collection covers the largest AMMs on each chain [28], including Uniswap V2–V4 and forks such as SushiSwap, PancakeSwap, and SunSwap on EVM chains, Aerodrome V1 on Base, and Raydium, Orca Whirlpools, Meteora, and Pump.fun AMM on Solana. We release the full data collection and detection pipeline as open-source code [49]. Our measurement spans from 1 July 2023 until 30 June 2026, with Monad data beginning at its mainnet launch on 24 November 2025. Appendix A details data collection, Ethereum mempool labeling, and OFA victim attribution.

4 Detection

At a high level, our heuristics to identify protected order flow sandwich attacks are the same on all six chains. We detect wide and cross-block attacks alongside tight ones, since an attacker on protected order flow places its legs without seeing the victim and rarely lands directly around it. Ethereum is the main exception, where we restrict the scope to transactions that never entered the public mempool. Solana differs in smaller ways, which account for the attack patterns we observe there and for one leader controlling four consecutive slots. We detail per-chain parameters in Appendix B.

H1: Structure. A **candidate** is a swap address that trades in *both directions* on a pool. We scan its swaps in chain order and aggregate consecutive same-direction swaps into a **front** leg and the following opposite-direction run into a **back** leg, which catches split legs [89]. The back legs must resell what the front legs bought, i.e., $A_{\text{b,in}} \in [(1-\varepsilon),(1+\varepsilon)]A_{\text{f,out}}$ with $\varepsilon = 0.01$, within five blocks on the EVM chains or two leader rotations on Solana. At least one swap on the same pool in the front direction, by another wallet and not itself a candidate leg, must sit strictly between the legs, and we count each such swap as a victim.

H2: One Role. A transaction cannot take several roles in the same sandwich, and we disqualify a victim that also serves as a leg of any candidate. Across sandwiches, a swap cannot act

as both a front and a back-run.

H3: Persistence. We attribute each detected sandwich to a bot and filter for the persistent ones. On EVM chains, the bot is a single sender, a recurring sender pair or a contract (cf. Appendix B). On Solana the bot is the trading wallet that submits the front leg, including in an **evasive** sandwich where a token transfer links it to a second wallet submitting the back leg (cf. Appendix F). We retain a bot with at least 100 sandwiches, at least 60% of them profitable, whose sandwich legs make up at least 25% of its swaps. An attack is profitable when the round trip gains in the traded token, i.e., $A_{\text{b,out}} > A_{\text{f,in}}$, before gas and fees.

To summarize, H1 identifies the sandwich structure, H2 keeps us from overcounting sandwiches, and H3 filters for persistence. The main difference between our heuristics and those of prior work [43, 67] lies in H3. It keeps us from mislabeling high-frequency trading in both directions as sandwiching. When a bot makes a million swaps and fewer than 1% of them carry the sandwich structure, those matches are more likely a by-product of its trading than genuine attacks. Appendix C reports an ablation of the three heuristics.

A sandwich can enclose more than one victim when several swaps in the front direction sit between the legs. In such cases, we count everyone as a victim. Yet, we cannot tell which the attacker targeted, as it's likely that only one is the actual victim. For the Ethereum private scope, this means a sandwich may enclose several private victims. Multi-victim sandwiches are rare on the EVM chains, with only a single victim sitting in 88.3% of sandwiches on Ethereum, 97.5% on Tron, and 98.8% on Base, whereas on Solana, 36.0% of sandwiches enclose more than one victim. We address this limitation for the victim analyses that follow in the corresponding sections.

5 Protected Order Flow Sandwiches

We characterize protected order flow sandwich attacks across the six chains in our study and contrast their structure with public-mempool sandwiches previously studied on Ethereum.

Recall the protections at play. Ethereum lets users submit transactions privately, and good data coverage exists on which ones did. Unless stated otherwise, we consider only victims of these private Ethereum transactions. Solana and Monad run local mempools, i.e., a transaction goes to a small set of upcoming block proposers instead of to everyone. Tron sequences its public mempool first-come, first-served, with proposers ordering transactions by arrival time. Base and Arbitrum route transactions to a single sequencer.

Table 3 holds the summary statistics for protected order flow sandwich attacks across the six chains. The number of attackers and the number of attacks split the chains into four groups. Solana stands alone with 28.0 million sandwiches by 8,631 bots. Ethereum and Tron follow with 7 and 12 bots and tens of thousands of attacks, i.e., 30,607 sandwiches on Ethereum and 38,567 on Tron. Base forms a third group with 4 bots and 1,889 attacks. Arbitrum and Monad make up the final group without persistent sandwich attackers.

	Attacks			Profit (USD)		
Chain	**Count**	**Entities**	**Profitable**	**Gross**	**Net**	**Priced**
Ethereum	30,607	7	91.9%	622,358	254,566	99.96%
Via reorged blocks[†]	2,576	93	98.9%	279,848	171,777	45.7%
Tron	38,567	12	95.4%	901,069	642,542	90.8%
Base	1,889	4	97.2%	3,031	2,416	99.7%
Solana	28,042,725	8,631	88.0%	383,433,932	345,185,014	>99.99%
Arbitrum	0	0	–	–	–	–
Monad	0	0	–	–	–	–

Table 3: Protected sandwich attacks by persistent bots per chain, 1 July 2023 to 30 June 2026. *Profitable* is the share whose round trip gained in the traded token, *priced* the share with a USD value, over which gross and net are summed. The reorged-block row lists sandwiches on private transactions that a stale block likely made public (cf. Section 6.1.6). [†]Profit and profitable share cover only the 45.7% of attacks whose fee is attributable to a single attack, the rest are multi-pool bundles (cf. Appendix E).

Protected order flow sandwiching is rampant on Solana, with 8,631 bots extracting $383.4 million in gross profit and retaining $345.2 million after fees, which take 10.0% of gross profit.

Turning to Ethereum, we find 30,607 sandwiches by 7 bots. Fees cut the profit from $622,358 to $254,566, i.e., 59.1% of the gross profit goes to fees. Note that these are only the sandwiches of bots specializing in private order flow. Sandwiches originating in the public mempool are covered in Appendix E.

A further 2,576 sandwiches executed by 93 bots land on 2,875 private transactions that a reorged block likely made public, i.e., the block carrying them was broadcast and then reorged out (cf. Section 6.1.6). Note that we price only 1,178 of the attacks, and the remaining bot transactions serve multiple attacks across pools (cf. Appendix E). For the priced attacks, the gross profit is $279,848, while the net is $171,777.

Tron operates on a similar scale with 38,567 attacks by 12 bots, at a higher gross profit of $901,069 and a higher net profit of $642,542, i.e., only 28.7% of the gross profit goes to fees. Base sits far below both, with 1,889 attacks by 4 bots and a gross profit of $3,031, of which $2,416 remains after fees, i.e., 20.3% of the gross profit goes to fees.

Two chains show no protected sandwiches. Arbitrum's protection design resembles Base's at a high level, with a centralized sequencer, and yet we observe no protected sandwiches on Arbitrum. Monad runs a local mempool design similar to Solana's and likewise shows none. We note that Monad has been live for less than a year, whereas Solana launched in March 2020, and the first protected sandwich we observe falls on 16 August 2023, i.e., years into its operation.[1] Thus,

[1] Our Solana window opens on 1 July 2023, six weeks before that first attack. Given how slowly attacks ramp up thereafter (cf. Figure 2d), we consider attacks before our window unlikely.

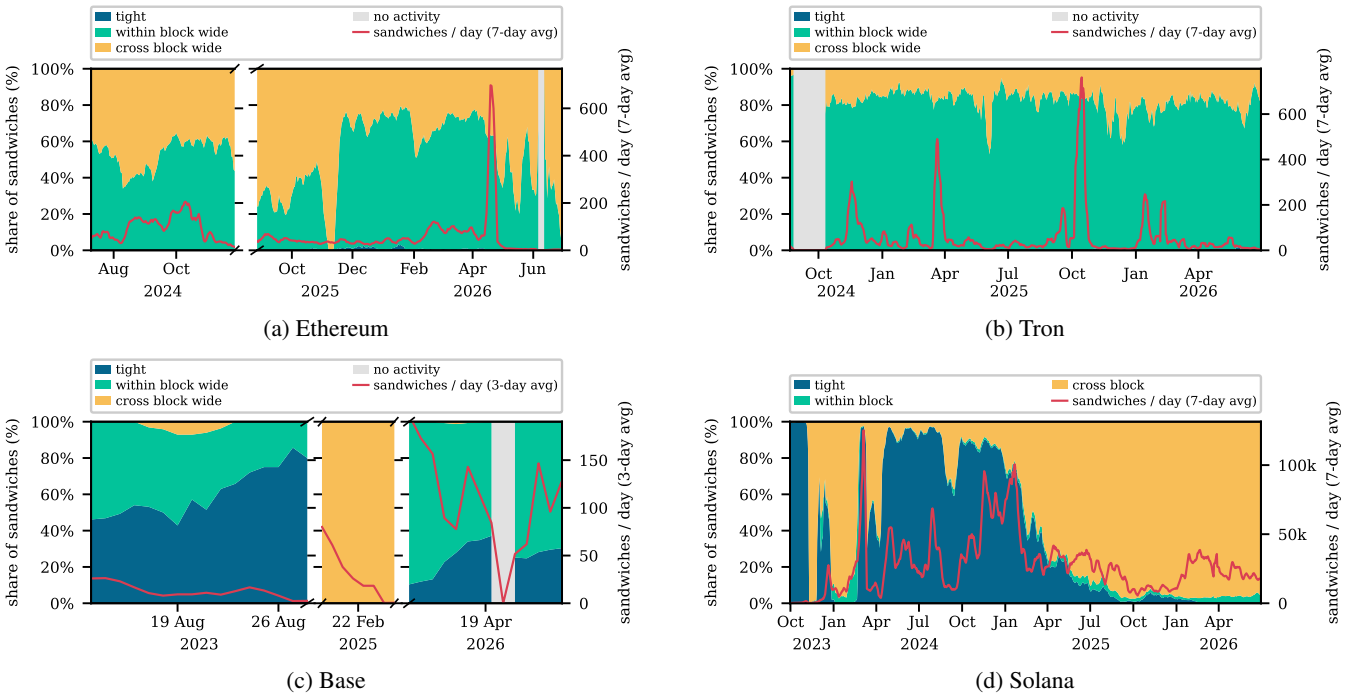


Figure 2: Sandwich attacks per day, per chain. The shaded area gives the attack-type composition (left axis) and the line the daily attack count (right axis), both smoothed over a rolling window. Gray marks days with no attacks.

even though the protections on Arbitrum and Monad mirror those on Base and Solana, respectively, we do not see them exploited in the same way. This observation points to a theme that recurs throughout our analysis: attack counts capture what attackers do today, not what is in theory possible.

The share of profitable attacks in Table 3 distinguishes protected sandwiches from Ethereum public mempool sandwiches. Across the four chains with protected sandwiches, the profitable share ranges from 88.0% on Solana to 97.2% on Base, with Ethereum at 91.9%. Public mempool sandwiches on Ethereum reach 98.9% (cf. Appendix E). Figure 15 in Appendix G further reports per-attack profit distribution per chain. Attacking protected flow likely leaves the attacker with less control over the execution order.

Thus, we turn to the structure of protected sandwiches. Figure 2 plots the number of protected sandwiches per day together with their structure. Our analysis spans 1 July 2023 to 30 June 2026 for all chains, and we restrict each panel to the periods with sandwich attacks by real bots.

On Ethereum we observe two periods of protected sandwich activity. The first sets in at the middle of 2024 and averages 96 attacks per active day, of which 53.4% are within-block wide and 46.6% cross-block wide. The second averages 59 attacks per day at a comparable split of 59.1% and 40.5%. It contains a single spike in April 2026, where the seven-day average peaks at 698 attacks per day before falling back. Tight sandwiches are extremely rare (0.25% of attacks).

Tron follows a similar pattern, with activity sustained from the end of 2024 to the middle of 2026. Attacks average 62 per active day, and 82.9% of them are within-block wide against 17.0% cross-block wide, the remainder is tight. On Tron most attacks are inside a single block, whereas half are spread across block boundaries on Ethereum.

On Base, activity is confined to three short bursts. The first one runs for 16 days in August 2023, with 13 attacks per day, and is the only period on Base in which tight sandwiches dominate, at 56.4%. The second lasts eight days in February 2025, with 30 attacks per day, and is entirely cross-block wide. The third covers two weeks in April 2026 and accounts for 76.7% of all Base attacks, with 76.0% within-block-wide and 23.8% tight. Each period carries the signature of a short-lived vulnerability and a distinct bot (cf. Section 6.3).

Solana has the most sandwiches per day, averaging 25,586 attacks. Activity starts at the end of 2023 and peaks at 125,169 just before Jito shuts down their temporary public mempool. Sustained activity continues at an average of 41,063 attacks per day until January 2025 and is dominated by tight sandwiches. It then declines to 26,127 per day through 2025 and to 20,663 after the last wave of validator delistings in October 2025 (cf. Section 6.4), shifting to cross-slot attacks.

Overall, tight protected sandwiches are rare, with Solana before 2025 and a short period on Base being notable exceptions. In the public mempool on Ethereum, the majority of sandwiches are tight, i.e., the front-running transaction, the

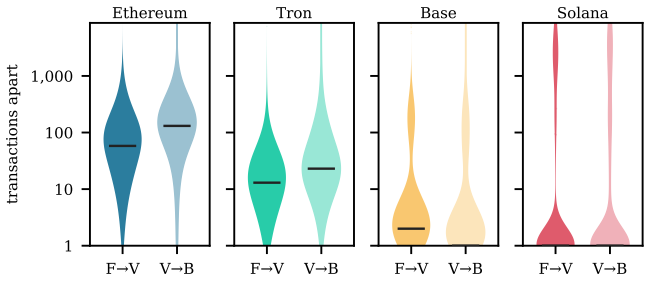


Figure 3: Transactions between the legs of a sandwich attack. F→V, from the front-running transaction to the first victim; V→B, from that victim to the back-running transaction.

Bot	SWs	T/W/C (%)	Gross (USD)	Fees	Succ.	Active
0x841...697	13,661	1 / 67 / 33	217,579	37%	86%	25-11–26-06
0x8a5...4db	10,202	0 / 55 / 45	240,985	86%	77%	24-08–24-11
0xe87...716	3,031	0 / 48 / 52	92,717	53%	74%	24-07–24-08
0x69b...b78	1,563	0 / 29 / 70	35,112	44%	81%	25-10–25-11
0x013...5ea	983	0 / 31 / 69	14,248	56%	80%	25-08–25-09
0xe9b...e02	808	0 / 23 / 77	18,462	35%	82%	25-09–25-10
0xc72...59b	359	0 / 55 / 45	3,255	30%	91%	26-04–26-06

Table 4: Ethereum bots, sandwiches (SWs), tight (T) / within-block-wide (W) / cross-block-wide split (C), gross profit in USD, the share of it consumed by fees, success rate (%), and active period. Rows with a gray background mark the bots we attribute to a single entity.

victim transaction, and the back-running transaction land one after the other. An attacker there sees the victim in the mempool and generally submits both legs around it in a single bundle through an MEV auction [66, 104]. An attacker on protected flow, on the other hand, often places its legs blind, as we will see next. Consequently, unrelated transactions may land between them or the legs can fall in separate blocks.

Figure 3 reports the distance from the front-run transaction to the first victim, and from that victim to the back-run transaction. Bots on Ethereum and Tron spread their legs widely, at medians of 58 and 131 transactions on Ethereum and 13 and 23 on Tron. For Base the medians sit at 2 and 1 transactions, and on Solana at 1 and 1. Solana's median comes from the period up to 2025, when tight sandwiches dominated, and the legs move apart once the attacks turn cross-block afterwards. Both chains still carry long tails, i.e., a 99th percentile of 314 on Base and 8,651 on Solana. Thus, protected order flow sandwich attacks are in stark contrast with Ethereum public mempool sandwiches, which have a median of 1 transaction and 99.4% of them within 6 (cf. Appendix E).

We further report the number of transaction between the front and back leg in Appendix G which in addition to the distance also reveals whether it has an effect on the attack. Distance alone does not make an attack suboptimal. A sandwich with 100 transactions between its legs still extracts the full amount as long as none of them swaps in the same pool. Any such swap in between, however, has the potential to make the attack unprofitable. We call an attack interfered with when any trade in the attacked pool other than its first victim executes between the two legs. We find that this interference is not rare, and it is most common on Solana, where 44.6% of attacks have an additional swap in the attacked pool between their legs, against 28.4% on Ethereum, 10.3% on Tron and 4.4% on Base (cf. Figure 14). Interference separates the failed attacks from the successful ones, i.e., a swap between the legs appears in 96.2% of unprofitable attacks on Ethereum against 22.4% of profitable ones, in 91.0% against 6.5% on Tron, and in 88.7% against 2.0% on Base.

To summarize, protected order flow sandwiches look nothing like their public mempool counterparts. Their legs sit far from the victim, and often in separate blocks, and their lower share of profitable attacks likely follows from the same gap, since the attacker acts on incomplete information.

6 Exposure Paths and Attacker Competition

We next examine how sandwich bots front-run seemingly protected order flow and compete with one another.

6.1 Ethereum

On Ethereum, we observe seven sandwich attack bots that specialize in front-running transactions that never enter the public mempool. Table 4 reports per-bot statistics. The two largest bots dominate, with 13,661 and 10,202 sandwiches, totaling 78.0% of the 30,607 attacks. Tight sandwiches are almost absent. The within-block share ranges from 23% to 67%, with the remainder falling across block boundaries. Success rates fall within a similar range, from 74% to 91%. Profit follows the attack count only loosely. For example, 0x8a5...4db earns a higher gross profit than 0x841...697 on 25% fewer sandwiches. The fee share separates the bots by period rather than by size. The bots active in 2024 give up the most, at 86% and 53% of their gross profit, whereas the two active into 2026 give up 37% and 30%. Ethereum fees fell over this period, which likely accounts for the difference.

Finally, the bots are generally active over non-overlapping windows. Figure 4 plots the weekly sandwich count with each bot shown separately. The bots hand over from one to the next inside a single week, and no two of them run alongside each other for longer. The exceptions are 0x841...697 and 0xc72...59b, which are active simultaneously for one week without following the handover pattern of the others. The handover pattern alone suggests that the first six bots are controlled by one entity. Bytecode similarity and deployment history indicate that, in fact, all seven bots belong to a single entity (cf. Appendix H.1). MEV is already known to be highly concentrated, yet for protected order flow sandwiches, a single entity is often the dominant case. This concentration is repeatedly observed throughout the paper.

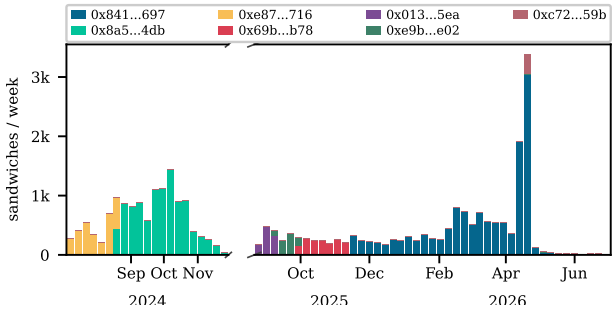


Figure 4: Weekly sandwiches on Ethereum by attacker bot.

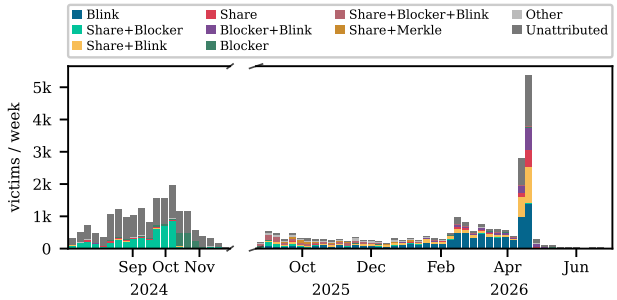


Figure 5: Weekly evolution of sandwich victims by observed OFA label combination for the persistent bots in Table 4. *Share* denotes MEV-Share and *Blocker* denotes MEVBlocker. Victims may appear in multiple datasets; for example, Share+Blocker denotes victims observed in both. *Unattributed* denotes victims not matched by any of the four datasets, while *Other* aggregates all remaining combinations, each accounting for less than 1% of victims overall.

6.1.1 Ruling Out Block Builders

On Ethereum, proposer-builder separation (PBS) separates block construction from block proposal [47]. Builders assemble transactions into candidate blocks and compete by submitting bids to proposers, who select the highest-bidding block for inclusion. As builders control transaction ordering, they occupy a priveleged position for MEV extraction, including sandwich attacks. To understand how transactions that never entered the public mempool were nevertheless front-run, we first examine whether protected sandwich attacks concentrate in the blocks of particular builders, given the emergence of integrated searcher-builder constellations [48]. Because builders receive private transactions in plaintext, they are an obvious potential exposure point in the trust model. However, no builder exhibits the clear over-representation that would be expected from a builder-specific exposure source (cf. Appendix I.1.1). We therefore move upstream in the transaction-submission pipeline of Figure 1 and examine the channels through which private transactions reach builders.

6.1.2 OFA Attribution

We turn to OFAs and examine the proportion of victim transactions that can be attributed to them. Among the 34,360 victims of 28,128 successful private sandwich attacks, 66.8% appear in at least one OFA dataset, with 35.5% appearing in multiple ones. We additionally identify 5,101 victims across 2,479 unsuccessful private sandwich attempts, of which 35.5% appear in at least one OFA dataset. These attempts satisfy the same structural criteria as successful sandwiches, but the attacker's front–back sequence fails our profitability criterion. We include successful and unsuccessful attempts in the following exposure analysis because, regardless of realized profit, an attempted sandwich indicates that the attacker was able to act around the victim transaction.

Across all sandwich attempts, 62.7% of victims appear in at least one OFA dataset. Through February 2025, we attribute 6,870 of 17,614 victims to at least one OFA, of which 70.4% appear jointly in MEV-Share and MEVBlocker. From August 2025 onward, we observe 21,847 victims, of which 17,889 are attributed to at least one OFA. Of these, 86.2% appear in the Blink data, while 46.2% are seen exclusively in Blink. Figure 5 shows the weekly evolution of OFA labels, with combinations accounting for less than 1% grouped as *Other*.

These temporal patterns should be interpreted in light of our data coverage. In particular, the Blink data begins only in August 2025, while 61.0% of victims up to February 2025 remain unattributed. More generally, the 37.3% of victims without an OFA attribution over the full period may still have passed through an OFA that is not captured by our external datasets. As discussed in Appendix A.1.2, these datasets provide positive evidence of observed OFA exposure but cannot establish that a transaction did not appear elsewhere. We therefore use the observed changes in OFA composition to motivate the mechanism-level analysis below, rather than as evidence of changes in underlying OFA usage. The full breakdown of observed OFA label combinations is reported in Table 21.

A further limitation is the presence of multi-victim sandwiches. Our methodology treats each qualifying transaction between the attacker legs as a victim, although in a multi-victim sandwich only one transaction may have been deliberately targeted and the others may have been included incidentally. Overall, 68.5% of victims on Ethereum are the sole target of their sandwich, while the remainder appear in multi-victim sandwiches. This may explain part of the unattributed population. Among victims without an observed OFA label, 56.5% occur in multi-victim sandwiches and therefore may not themselves have been the transaction whose exposure enabled the attack. Conversely, an OFA-attributed transaction in a multi-victim sandwich may also have been included incidentally rather than exploited through its OFA submission path. We report the single-victim share for each OFA label combination in Table 21 and account for this distinction in the mechanism-level analysis below.

6.1.3 OFA Exposure Mechanisms

In the following, we study how sandwich bots may exploit information exposed through OFAs.

Cross-OFA Correlation. Overall, 53.0% of all attributed sandwich victims appear in multiple OFA datasets. This is particularly relevant for MEV-Share and MEVBlocker, whose disclosure mechanisms can be correlated to reveal additional information. MEV-Share selectively discloses transaction information together with a double-hashed transaction identifier [38]. MEVBlocker mixes genuine transactions with decoys, and for swap transactions, withholds sensitive information. Transactions unlikely to yield a back-run are not exposed [70]. A searcher observing both feeds can double-hash transaction hashes exposed by MEVBlocker and compare them against the MEV-Share stream. A match identifies a genuine MEVBlocker transaction and can reveal information unavailable from either feed in isolation.

Among attributed victims, 28.3% appear in both MEV-Share and MEVBlocker, of which 75.3% are in no other OFA (cf. Table 21). Importantly, 88.6% of victims in this MEV-Share–MEVBlocker-only category occur in single-victim sandwiches, making incidental inclusion unlikely. Together, the prevalence of this combination and its concentration among sole-target victims make cross-OFA correlation a plausible exposure mechanism for a substantial class of attacks.

Full-Transaction Disclosure. Blink and Merkle follow a different security model. Rather than obscuring transaction contents from searchers, they restrict order flow access to an approved set of participants, to whom the full transaction is disclosed. Thus, once a searcher has access to the feed, no additional cross-feed inference is needed to identify or size a potential sandwich. Instead, protection depends on admitted searchers' behavior and the access policy enforcement.

This model is relevant for a substantial fraction of attributed victims. Blink appears in 62.2% of attributed victims, and 33.4% of attributed victims are observed only in Blink (cf. Table 21). Among Blink-only victims, 89.8% occur in single-victim sandwiches, making incidental inclusion unlikely.

Blink also appears jointly with other OFAs in several large categories, including Blink–MEV-Share (14.3% of attributed victims), Blink–MEVBlocker (6.9%), and Blink–MEV-Share–MEVBlocker (5.8%). In these cases, Blink's full-transaction disclosure is already sufficient to expose the information needed for a potential sandwich. The additional OFA observations are therefore not necessary to explain informational sufficiency. They may nevertheless reflect other routing or exposure paths. These observations do not establish that the attacker obtained the transaction from Blink, since other unobserved exposure paths may exist. Nevertheless, they show that a large class of victim transactions was available in full to Blink's approved searchers before inclusion.

Merkle follows the same disclosure model, although it appears in only 5.1% of attributed victims and was subsequently merged into Blink. Only 104 victims are observed exclusively in Merkle, corresponding to 0.4% of attributed victims. Among these Merkle-only victims, 43.3% occur in multi-victim sandwiches, making the evidence for Merkle as the exposure source less direct in these cases.

Unresolved Single-OFA Exposure. We also observe victims associated with MEV-Share or MEVBlocker in isolation in our attribution data: 1,727 MEV-Share-only victims (7% of attributed) and 1,553 MEVBlocker-only victims (6.3%; cf. Table 21). Figure 19 shows that these cases are strongly concentrated in time. MEVBlocker-only victims exhibit a short-lived spike around October–November 2024 and are nearly absent thereafter. MEV-Share-only victims are more dispersed, but show a pronounced spike in April 2026.

For these isolated labels, we do not identify an exposure mechanism analogous to those described above. An additional exposure source may therefore be missing from our data. For the MEVBlocker-only cases, we cannot rule out exposure through another OFA for which we lack a corresponding label. The MEV-Share-only spike in April 2026, in turn, coincides with a period of high Blink-associated sandwich activity. Because our Blink labels are derived from bundles observed by BuilderNet, Blink bundles not observed by BuilderNet would be absent from our attribution data.

Multi-victim sandwiches provide another possible explanation, particularly for the MEV-Share-only cases. We find that 70.5% of MEV-Share-only victims occur in multi-victim sandwiches, and may therefore have been included incidentally rather than being the transaction whose exposure enabled the attack. In contrast, only 32.6% of MEVBlocker-only victims occur in multi-victim sandwiches, leaving the majority of these cases unexplained by incidental inclusion alone.

6.1.4 Attacker Behavior

Table 5 shows that OFA association also differs across bot contracts, despite our evidence that they are likely controlled by the same entity. The largest bot, `0x841...697`, is predominantly associated with Blink, which appears among 74% of its victims. In contrast, `0x8a5...4db` and `0xe87...716` have no observed Blink-associated victims and are instead concentrated in MEV-Share and MEVBlocker order flow. Smaller attackers active in the later period exhibit more heterogeneous OFA associations, often spanning several providers. Together with the temporal patterns in Figure 4 and Figure 5, this suggests that the entity adapts its exposure or routing strategy as the observed OFA environment changes, rather than using a single strategy throughout the measurement period.

6.1.5 Back-Running Opportunities

OFAs expose transaction information to searchers, enabling arbitrage back-runs that generate value for users. Yet, nearly two in three sandwich victims are attributed to at least one

Bot	Victims	MEV-Share	MEVBlocker	Blink	Merkle	Unattributed
0x841...697	17,355	30%	14%	74%	3%	19%
0x8a5...4db	13,400	30%	38%	0%	0%	59%
0xe87...716	4,214	30%	29%	0%	0%	67%
0x69b...b78	1,781	53%	24%	64%	20%	14%
0x013...5ea	1,101	72%	59%	59%	10%	10%
0xe9b...e02	995	67%	34%	47%	27%	11%
0xc72...59b	615	33%	16%	58%	0%	35%

Table 5: Observed OFA association by Ethereum attacker bot, for the persistent bots of Table 4. Each OFA column reports the share of the bot's victims observed in the corresponding OFA dataset; columns need not sum to 100% because a victim may appear in multiple datasets. *Unattributed* denotes victims with no observed OFA label.

OFA. This raises a natural question: why were these transactions attractive sandwich targets despite being exposed to competing back-runners? An arbitrage back-run executed between the victim transaction and the sandwicher's back-run can reduce or even eliminate the sandwich profit (cf. Appendix I.1.3 for a benchmark). However, such an opportunity exists only when sufficient alternative liquidity is available. Thus, we examine whether these victims offered profitable arbitrage back-running opportunities.

A direct arbitrage against the sandwiched X–Y pool requires at least another active X–Y pool, while additional liquidity in X and Y may enable multihop routes. For each of the 39,461 victim transactions across our 30,607 sandwiches, we therefore compute two measures: the number of active X–Y pools (*direct count*) and, as a proxy for potential multihop routes, the minimum number of active pools containing X or Y (*endpoint proxy*). Both measures include the source pool and indicate only potential routing opportunities, but not whether an arbitrage route is executable or profitable.

At the end of the preceding block, the direct pool count is one for 26,887 victims (68.1%), two for 8,333 (21.1%), and greater than two for 4,241 (10.7%). The endpoint proxy is one pool for 22,824 victims (57.8%), two for 5,893 (14.9%), and greater than two for 10,744 (27.2%). Thus, 31.9% have at least one active direct alternative, while 42.2% satisfy the broader criterion for a potential multihop route.

Appendix I.1.3 evaluates a victim-only direct-route counterfactual for the 31,259 victims whose front-run occurs in the same block. We classify a victim as *not back-runnable under our coverage* when either (i) its endpoint proxy is one, or (ii) its endpoint proxy equals its direct count and every direct alternative has a conclusive, non-positive gross-profit optimum. In the first case, at least one endpoint has no active non-source pool, ruling out a source-pool-disjoint route. In the second, all alternatives incident to the lower-degree endpoint are direct X–Y pools, exhausting the possible simple source-pool-disjoint routes, and each is conclusively unprofitable.

Of the 31,259 same-block cases, 17,993 satisfy the first condition. Among the remainder, 3,475 have direct alternatives but no gross-profitable direct back-run, and 1,417 of these also have an endpoint proxy equal to their direct count. Hence, we rule out a profitable back-run for 19,410 victims (62.1%) under our coverage. Of the remaining 11,849 victims, 3,905 have a gross-profitable direct route, 2,504 have an inconclusive direct-route result, and 5,440 have no detected profitable direct route but remain unresolved because of possible multihop paths or an unmodeled source state, such as those of V4 hooks. Profit is ETH-evaluable for 3,863 of the gross-positive cases, of which 484 (1.5% of the full same-block cohort) remain profitable under our gas assumptions. Thus, profitable direct back-runs appear rare in our counterfactual, which may help explain why these transactions remained attractive sandwich targets despite their exposure to OFA searchers.

6.1.6 Reorged Blocks

In addition to OFA exposure, we examine reorged blocks as a second way in which private transactions can become exposed. Ethereum blocks are occasionally reorged, when a block is proposed but does not become part of the canonical chain [10, 32]. This generally happens because the block was not proposed in time and failed to gather enough votes before the deadline. Any private transaction in such a block loses its privacy since the block carrying it has been broadcast.

We look for transactions that were private when the reorged block was proposed, not part of a sandwich in that block, and later a victim of a sandwich on the canonical chain. Across the 9,297 reorged blocks in our data, covering January 2024 to June 2026, we find 140,854 swap transactions, of which 56,730 were private when the stale block was proposed. Publication of the stale block reveals all of these transactions. In addition, 62.0% (35,193) are later observed individually in the public mempool before their canonical inclusion. Upon re-inclusion, 2,875 are sandwiched: 72.4% (2,082) are republished in the public mempool beforehand, while 27.6% (793) are not. The counts keep H3's persistence filter but also includes the bots specializing on the public mempool.

Figure 6 plots the monthly number of these sandwiches. Until the start of 2025, nearly all of the victims had been republished in the public mempool. From the beginning of 2025, we observe sandwiches on victims that never appeared in the public mempool individually and became visible only as part of the reorged block. This suggests that some bots specialize in monitoring stale blocks and sandwiching the swaps they reveal. The attacker populations differ accordingly. In total, 93 bots attack these victims. Victims republished in the public mempool before they are sandwiched in the canonical block fall to 87 of them, whereas the extraction on victims never seen individually is highly concentrated, with one bot attacking 55% of the victims and the top five 82%.

Victims not observed individually in the public mempool face a narrower attacker set than republished victims, which are exposed to the broader public-mempool searcher population. The profit, thus, shows a clear difference (cf. Table 6).

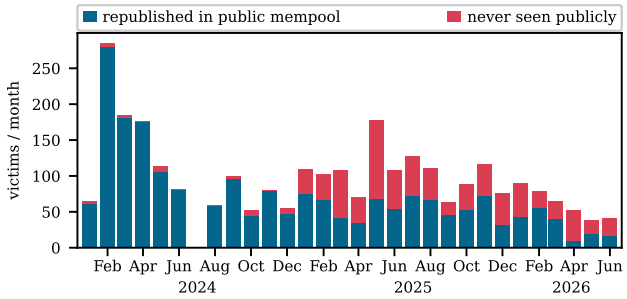


Figure 6: Reorged private transactions sandwiched after canonical re-inclusion, per month, split by whether the transaction was republished in the public mempool between the reorg and its re-inclusion or never observed publicly.

Victim Exposure	SWs	Gross (USD)	Fees	Median Net (USD)
Never seen publicly	592	$160,672	11%	$37.31
Republished	586	$119,176	76%	$0.01
All	**1,178**	**$279,848**	**39%**	**$6.85**

Table 6: Profit of reorg-exposed sandwiches by exposure mechanism, restricted to the 1,178 of 2,576 attacks whose fee is attributable to a single attack. The rest are multi-pool bundles, often carrying an arbitrage cycle, whose per-attack net profit would understate the return (cf. Appendix E).

The median attack nets $37 on a victim never seen publicly and one cent on a republished one, where competition pushes fees from 11% of the gross profit to 76%.

In summary, reorged blocks provide a second exposure avenue: transactions originally protected from front-running can become targets as if they had initially been submitted publicly. In principle, an attacker who could induce such a reorg might profit from the resulting exposure, for example by incentivizing validators to build on a competing branch. Whether such a strategy is economically or operationally feasible depends on the consensus conditions and the cost of inducing the reorg, and our empirical evidence is not sufficient to determine whether any of the observed reorgs were triggered for this purpose. Note that our estimates are conservative, as we observe only Xatu-recorded reorgs and recover 65.9% of stale block bodies (cf. Appendix A.2).

6.2 Tron

Next, we turn to Tron. Table 7 summarizes the per-bot statistics. As on Ethereum, activity is highly concentrated: two bots account for 19,247 and 10,438 sandwiches, respectively. Together, they conduct 77.0% of the 38,567 attacks. Between 72% and 89% of each entity's attacks are within-block wide, with the remainder falling across block boundaries. Profit tracks the attack count only loosely. The largest entity grosses $540,247, more than three times the second largest, and under twice the attacks. We note that it also has the lowest success rate, at 74%, suggesting a higher risk tolerance. The remaining success rates range from 80% to 96%. The largest bot has also been active the longest, from August 2024 to June 2026. It faced no competition until February 2025; a second possible explanation for its profit.

Bot	SWs	T/W/C (%)	Gross (USD)	Fees	Succ.	Active
0xc83...28b	19,247	0 / 84 / 16	540,247	35%	74%	24-08–26-06
0x35e...9a1	10,438	0 / 86 / 14	169,339	28%	88%	25-07–26-06
0x728...6e9	2,218	0 / 77 / 23	35,727	9%	90%	25-02–26-05
0x1f0...600	1,731	0 / 75 / 25	85,159	11%	88%	25-03–26-06
0x8ad...267	1,355	0 / 78 / 22	24,270	13%	87%	25-03–26-04
0x1f0...f9e	1,033	0 / 72 / 27	7,321	45%	81%	25-03–26-06
0x1f0...bd4	749	0 / 79 / 21	17,118	23%	83%	25-03–26-06
0x1f0...e9f	650	0 / 82 / 18	9,549	9%	93%	25-03–25-10
0x34f...978	374	0 / 83 / 16	6,469	5%	89%	25-03–26-06
0xfe9...ef4	322	1 / 73 / 26	1,111	62%	80%	25-10–26-06
0x9f0...70d	302	0 / 82 / 18	3,883	7%	91%	25-03–26-05
0xb2c...713	148	0 / 89 / 11	878	9%	96%	25-02–26-06

Table 7: Tron bots, sandwiches (SWs), tight (T) / within-block-wide (W) / cross-block-wide split (C), gross profit in USD, the share of it consumed by fees, success rate (%), and active period. Gray rows mark the nine bots we attribute to one entity (cf. Appendix H.3).

As on Ethereum, we cluster the Tron bots into entities. Table 7 marks in gray the nine bots we attribute to one entity. Three observations support the grouping (cf. Appendix H.3). First, all nine received their first on-chain funding from a single shared wallet. Second, they partition the pools they attack, sharing no pool across any of their 36 pairs. Third, four of them further share an address prefix that only brute-forcing produces, so one operator created them together. Thus, protected order flow sandwiching on Tron rests with four entities behind twelve bots.

6.2.1 Ruling Out Block Producers

On Tron, transactions become visible as they are gossiped through the public P2P network. Front-running protection comes from the ordering policy of the block producer, which is intended to provide first-come-first-served inclusion. This ordering rule is implemented at the client level and could, in principle, be replaced by a custom policy. As on Ethereum, we first look at the block producers (Super Representatives on Tron), as a potential source of attacks on protected order flow. However, attacks are distributed almost uniformly across producers (cf. Appendix I.2), making producer misbehavior an unlikely explanation. Thus, we turn to potential weaknesses in Tron's front-running-protection mechanism itself.

6.2.2 Front-Running Through Latency Races

A plausible explanation is that attackers obtain their front-running position through a latency race, although our data cannot establish this directly. One possible strategy is to query

Bot	SWs	T/W/C (%)	Gross (USD)	Fees	Succ.	Active
0xc9c...41b	750	17 / 83 / 0	673	64%	80%	26-04–26-04
0xfcf...a26	698	31 / 69 / 0	268	23%	84%	26-04–26-05
0x14d...b75	239	0 / 0 / 100	206	4%	51%	25-02–25-03
0x000...b3e	202	56 / 42 / 1	1,885	6%	99%	23-08–23-08

Table 8: Base bots, sandwiches (SWs), tight (T) / within-block-wide (W) / cross-block-wide split (C), gross profit in USD, the share of it consumed by fees, success rate (%), and active period. We believe the two shaded rows to be controlled by the same operator (cf. Appendix H.2).

pending transaction pool, which nodes expose through the `gettransactionlistfrompending` endpoint.[2] An attacker polling this endpoint across multiple nodes can learn of a victim swap shortly after it reaches one of the monitored nodes, immediately submits the front leg, and race the victim transaction to the block producer. Peering with many nodes and minimizing latency to producers could improve the chances of the attacker of winning this race often enough to make the strategy profitable. The back leg can then be submitted with a short delay. Tron states that its ordering mechanism blocks around 98% of front-running transactions and attributes the remainder to bots that mostly fail [95]. Our data does not support this: the twelve bots we identify land 38,567 sandwiches with high per-entity success rates between 74% and 96%.

6.3 Base

On Base, our detector identifies four sandwich bot contracts that form three distinct groups, each active during a separate period (cf. Table 8). 0x000...b3e operates almost entirely within a single block where 56% of its sandwiches are tight, 42% are within-block wide, and only 1% are cross-block wide. Despite conducting the fewest sandwich attempts (202), it earns the largest gross profit ($1,885) and has the highest success rate (99%). We attribute these sandwiches to transaction-pool leakage (cf. Section 6.3.1). In contrast, 0x14d...b75 operates exclusively across blocks and has the lowest success rate, at 51%. These attacks point to the exploitation of predictable victim behavior (cf. Section 6.3.2). Finally, 0xc9c...41b and 0xfcf...a26 account for the most sandwiches, with 750 and 698, respectively, and have the highest fee shares, at 64% and 23%. All but three of their sandwiches occur within a single block. We likewise attribute this group's attacks to predictable victim behavior (cf. Section 6.3.2).

6.3.1 Transaction-Pool Leakage

The sandwich attempts by 0x000...b3e exhibit two patterns consistent with transaction leakage through an RPC or other order flow provider. First, the bot's front- and back-run legs

[2] https://developers.tron.network/reference/gettransactionlistfrompending

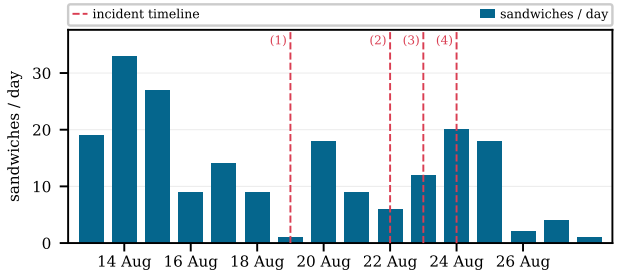


Figure 7: Daily sandwich attacks by 0x000...b3e on Base against the public timeline of the August 2023 transaction-pool leak: (1) incident reported, (2) op-geth fix #118, (3) op-geth fix #122 / Base fix #100, (4) incident resolved. The bot's detected sandwich activity spans these 16 days.

are positioned tightly around the victim, and their fee settings closely track those of the victim even when victim fees vary substantially. Second, the bot's active period coincides with a documented Base incident in which transactions from local pending pools were exposed [6–8, 76] (cf. Figure 7). Together, these observations point to the documented transaction-pool leak as the likely exposure source for this bot.

6.3.2 Predictable Behavior

The victims of 0x14d...b75, 0xc9c...41b and 0xfcf...a26 exhibit recurring trading patterns that can make their future transactions predictable. The clearest case is 0x14d...b75, which conducts sandwiches exclusively across blocks and targets a concentrated group of victims that follow a fixed trading loop. Three wallets, one of them 0xd3f...f17, sell a similar amount of WETH roughly every two minutes for a different token, over 230 times in total, and the bot places its sandwich legs in the neighboring blocks (cf. Table 9). The combination of this highly regular victim behavior and exclusively cross-block sandwiches points to prediction rather than pending-transaction visibility: recurring trade becomes predictable, thus the attacker need not observe the transaction before submitting its front leg.

The victims of 0xc9c...41b and 0xfcf...a26 likewise exhibit regular, funding-linked trading patterns, although less rigidly than in the fixed loop above. At the same time, both bots closely match their victims' fee settings, including the few transactions with irregular fees. Predictable victim behavior may account for part of these attacks, while some additional transaction visibility may also have been available to set fees appropriately (cf. Appendix I.3).

Overall, Base illustrates two distinct ways in which front-running protection can fail. Transactions may become exposed within the submission pipeline, as in the documented leakage incident, while sufficiently regular victim behavior can make future trades predictable without requiring pending-transaction visibility.

Block	Role	Account	Trade	Transaction
26,721,580	Front-Run	0x14d...b75	0.000688 WETH → 2.6M ANNON	0x0bd7aef3
26,721,581	Victim	0xd3f...f17	0.075800 WETH → 275.7M ANNON	0x723adb6e
26,721,583	Back-Run	0x14d...b75	2.6M ANNON → 0.000726 WETH	0xe89200b4
26,721,644	Front-Run	0x14d...b75	0.000726 WETH → 2.8M GROK3 AI	0x740cb250
26,721,646	Victim	0xd3f...f17	0.070300 WETH → 264.4M GROK3 AI	0x8d66df63
26,721,647	Back-Run	0x14d...b75	2.8M GROK3 AI → 0.000763 WETH	0x1a223921
26,721,700	Front-Run	0x14d...b75	0.000763 WETH → 2.6M DEEPSEEK	0x8d1e9914
26,721,702	Victim	0xd3f...f17	0.098500 WETH → 317.3M DEEPSEEK	0xc6171d66
26,721,703	Back-Run	0x14d...b75	2.6M DEEPSEEK → 0.000819 WETH	0x1f93db48

Table 9: Three consecutive cycles of the scripted loop that 0x14d...b75 sandwiches. The near-identical cycle repeats over 230 times. Roughly every two minutes, one of three wallets, among them 0xd3f...f17, sells a similar WETH amount for a different token. The repetition makes the next swap, so the bot needs no pending-transaction visibility.

6.4 Solana

Finally, we turn to Solana, which accounts for by far the largest number of protected sandwich attacks in our dataset. We note that the degree of protection varies across chains. Base relies on a centralized sequencer whose pending transaction queue is not visible externally. Solana instead forwards transactions to a small set of upcoming leaders rather than broadcasting them network-wide, limiting their pre-inclusion visibility. As the Solana Foundation notes, this design "reduces the MEV surface area compared to chains with public mempools" [91], while not eliminating it entirely.

On Solana, attack volume is substantially larger than on the other chains, accompanied by fiercer competition among attackers. Moreover, the nature of sandwich attacks changes over time in response to interventions by the ecosystem (cf. Table 10). Thus, we organize our Solana analysis by era rather than the in-depth per-bot analysis used for the other chains.

The first era covers the Jito mempool, which operated through Jito's Block Engine until March 2024. A validator running Jito-Solana connected to a Jito relayer, which received the transactions destined for that validator. The relayer held each transaction for up to 200 ms before forwarding it, while searchers subscribed to the stream could observe transactions before execution [55, 56]. In effect, this created a public mempool for a subset of Solana transactions.

Our collection window begins in July 2023, with the first notable attack volume appearing in October (cf. Table 10). Activity increases sharply around February 2024 and exceeds 100,000 sandwiches per day (cf. Figure 2d). It then drops abruptly when Jito suspends the mempool, citing "negative externalities impacting users on Solana" [54]. Sandwich activity is also concentrated among a subset of validators. We find that 50% of attacks occur in slots led by validators that are over-represented relative to their share of blocks (cf. Table 10). This is consistent with only a subset of validators receiving the transaction flow exposed through the Jito mempool.

The second, third, and fourth eras end each with a validator-delisting event. In June 2024, the Solana Foundation removed validators from its delegation program [64]. Large stake pools, such as Marinade, subsequently took similar measures, blocklisting validators from its stake auction in February 2025 [68], and Jito's stake pool (JitoSOL) delisted additional validators in October 2025 [101]. These interventions targeted validators identified as facilitating sandwich attacks and removed delegated stake from them.

Era	From	SWs	Daily	T/W/C (%)	Bots	Supp/Elev	Elev
Jito mempool	07/23	2.3M	9k	54 / 7 / 39	386	1,022/282	50%
Mempool closed	03/24	2.6M	28k	87 / 1 / 13	287	61/27	7%
SFDP delist	06/24	11.1M	46k	84 / 1 / 15	2,082	40/172	16%
Marinade MIP.9	02/25	6.7M	26k	18 / 3 / 79	4,256	13/115	9%
JitoSOL delist	10/25	5.2M	20k	1 / 3 / 96	3,115	6/8	2%

Table 10: Solana sandwiching per era: daily sandwiches, tight (T) / within-block-wide (W) / cross-block-wide split (C), suppression (supp) / elevation (elev) counts leaders admitting under 0.5× or over 1.5× their share of the era's attacks given the blocks they produced, counted over leaders with at least 5,000 blocks in the era, and elev representing the share of attacks landing in the latter's slots.

Across these eras, the share of sandwiches landing in slots led by over-represented validators falls sharply, from 50% under the Jito mempool era, to 2% in the final era. Daily attack volume, however, does not follow. It climbs to roughly 46,000 attacks per day and later settles around 20,000 (cf. Figure 2d). What changes substantially is the structure of the attacks. Tight sandwiches comprise between 54% and 87% of attacks in the eras before the delisting, but only 18% in the Marinade era and 1% after the JitoSOL delisting.

The final era begins after the JitoSOL delisting. Same block sandwiches collapse to 4% of attacks: 1% of attacks are tight and 3% within-block wide. Thus, 96% of attacks span blocks. Daily attack volume settles at roughly 20,000, while the share of attacks landing in slots led by over-represented leaders drops to 2%. These patterns make leader-specific exposure an unlikely explanation for most attacks in this era. Most sandwiches span slots produced by different leaders, and only six leaders are under- and eight are over-represented. Instead, the victims concentrate among the users of a few applications.

We thus observe two distinct exposure signals over time on Solana. The first appears at the validator level, which we examine next (cf. Section 6.4.1). Their role is most apparent during the Jito mempool era and recedes over time as the ecosystem interventions accumulate, from the closure of the mempool to the successive validator delistings. The second appears at the application layer (cf. Section 6.4.2) and becomes the strongest signal in the later period.

6.4.1 Validator-Level Exposure

Recall that the first four eras show elevated sandwich activity in the slots of a subset of validators. Figure 8 compares, per leader, the sandwich rate in its own slots with the rate in

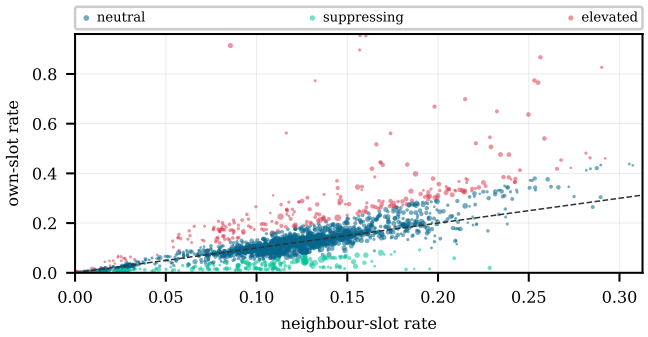


Figure 8: Sandwiches per slot in each Solana leader's own slots against the rate in neighboring slots of other validators.

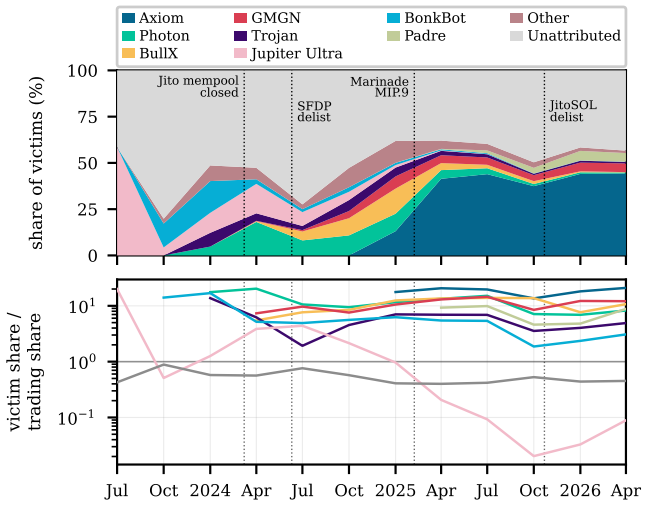


Figure 9: Quarterly victim share per application on Solana (top) and its ratio to swap share (bottom, from 80,896 sampled slots). Dotted rules mark interventions.

neighboring slots led by other validators, revealing a clear separation between groups. The leaders in blue lie close to the equality line, i.e., the attack rate in their slots is similar to the surrounding rate. The leaders in red carry far more attacks than their neighbors, whereas those shown in green carry substantially fewer. This heterogeneity is consistent with validator differences in how protected order flow or sandwiches are handled. The over-represented validators provide evidence of validator-level exposure, while the under-represented validators appear to suppress or exclude such attacks.

The concentration fades over time. During the Jito mempool era, the ten most over-represented leaders account for 26.4% of sandwiches while producing only 15.7% of blocks, corresponding to an excess of 1.68 times. This excess falls to 1.01 by the Marinade era. The distribution of per-leader sandwich rates, i.e., the share of each leader's blocks containing a sandwich, tells the same story. When sandwich activity is concentrated among specific leaders, their blocks carry substantially more sandwiches than those of other leaders. Under the Jito mempool, this distribution is bimodal, as the mempool was opt-in per validator: a validator either connected to the Jito relayer or not. By the Marinade era, the distribution becomes unimodal and the two groups largely merge, indicating that sandwich activity is no longer concentrated among a distinct subset of leaders. Appendix I.4 tracks this excess weekly alongside the ecosystem interventions and reports the corresponding Lorenz curves and per-leader sandwich-rate distributions (cf. Figures 22, 23, and 24). Attackers also adapt to these interventions. Evasive sandwiches increase after the Jito mempool closure, with bots increasingly splitting their legs as validator-specific exposure declines (cf. Appendix F).

To summarize, Solana forwards each transaction to only a small set of upcoming leaders, and leader-side confidentiality would substantially limit the opportunity for sandwiching in the absence of upstream exposure. Yet, particular validators show clear excess attack rates in their slots, most prominently in the early eras, providing evidence of validator-level exposure during the first part of our observations. This signal largely disappears around the start of 2025, while sandwich attacks persist. Their continued presence thus points to an exposure further upstream in the transaction-submission pipeline. We next examine applications as a potential source.

6.4.2 Application-Level Exposure

Figure 9 shows how victim concentration across applications evolves over time, for the eight applications with the largest number of victims. An application here is the interface a user trades through, such as a Telegram bot or a DApp like Axiom [4], Photon [85], BullX [17], Trojan [99], or GMGN [42], that constructs and submits the transaction on the user's behalf. The bottom panel normalizes each application's victim share by its share of all swap transactions. A value above one therefore indicates that the application's users are sandwiched more often than their share of trading activity would imply.

All applications except Jupiter Ultra [57] are over-represented among sandwich victims. This may partly reflect differences in the order flow they route. Telegram bots and DApps handle more long-tail token volume through thinner liquidity pools, where sandwich opportunities can be more attractive, whereas Jupiter Ultra routes less of this flow. Unattributed transactions are under-represented among victims. One possible explanation is that users submitting transactions directly, rather than through an identified application, are less exposed to the application layer providers. Our data, however, does not allow us to determine whether this is due to differences in user sophistication, submission infrastructure, or other characteristics of the underlying order flow.

Axiom emerges as validator-level exposure declines, accounting for 41.4% of victims in Q2 2025 and remaining above 37% thereafter. Its excess ratio ranges from 13.6 to 20.9. Photon, GMGN, and BullX exhibit similar but smaller

patterns, with excess ratios peaking at 20.3, 14.5, and 13.9, respectively. In multi-victim sandwiches, however, we cannot determine which of the enclosed transactions was deliberately targeted (cf. Section 4). Thus, we recompute the excess using only single-victim sandwiches. The ratios decrease, by roughly half for Axiom, but remain well above one for every over-represented application (cf. Table 23 in Appendix I.4).

Many of these transactions nevertheless use explicit front-running-protection mechanisms. Jito's `jitodontfront` feature lets a transaction opt-out of being front-run within a bundle. When the flag is set, Jito's Block Engine accepts only bundles that place the flagged transaction first, preventing another transaction in the same bundle from executing ahead of it. The guarantee therefore applies specifically to front-running within the Jito bundle. Appendix I.4 reports the per-application victim shares and adoption rates of `jitodontfront`, with adoption near universal among several of the most heavily sandwiched applications.

Axiom, the application associated with the most victims, makes the limits of this protection explicit. It offers three settings. *Off* provides no dedicated MEV protection. *Reduced* routes the transaction through Jito bundles and thus inherits exactly the guarantee above, i.e., no front-running within a bundle, but no protection beyond that path. *Secure*, the recommended setting, additionally restricts submission to a whitelist of validators. However, our measurements are hard to reconcile with interpreting these settings as providing end-to-end protection. Axiom sets `jitodontfront` on 98.5% of its 21.5M sandwiched victim transactions since the flag launched, while its victim share stays above 37% and its excess ratio never falls below 13. Near-total flag adoption therefore does not prevent substantial sandwich exposure in our data. The attack structure helps explain this limitation. Of the 5.02M sandwiches involving flagged Axiom victims, only 16.5% carry a Jito tip. Thus, the large majority do not exhibit the bundle-path signal governed by `jitodontfront`, consistent with these transactions reaching the leader through another submission path.

In conclusion, exposure on Solana shifts rather than disappears. From the end of 2023 to the beginning of 2025, we observe a strong validator-level signal, with a subset of validators carrying a substantially larger share of sandwiches than of blocks produced. This concentration fades as the ecosystem progressively delists involved validators. In the later period, the strongest signal instead appears at the application layer, although our data does not establish whether the applications themselves are the exposure source or whether correlated order flow characteristics contribute to the excess. Each intervention therefore appears to remove or weaken one source of exposure without eliminating the attacks.

7 Related Work

Quantifying MEV. Prior work has quantified MEV predominantly on Ethereum [25, 46, 48, 87, 88, 97, 103, 112], with Torres et al. [98] extending the analysis to L2s and finding no evidence of sandwiching through August 2023. Weintraub et al. [104] show that MEV increasingly shifted to private mempools, while Öz et al. [78] document cross-chain MEV through arbitrage. Mancino et al. [67] report that 40% of sandwich victims migrate to private routing within 60 days and identify 2,932 private sandwiches in November–December 2024. We observe substantially fewer, partly due to Mempool Dumpster outages that misclassified victims as private. Gogol et al. [43] study sandwiching across L2s but lack systematic evidence, whereas our stricter detection captures multi-block attacks and requires persistent attacker activity. Sandwiched.me [89] reports Solana sandwiching without examining leader or application exposure, while Gerzon et al. [41] study Jito bundles over four months. Our measurement spans three years and is not limited to Jito. Pahari et al. [80] show that private transactions can later become public, exposing them to sandwiching. Finally, recent work examines speculative MEV and its implications for the ecosystem [44, 81, 90, 92, 102, 105].

MEV Protections. A range of mechanisms mitigate front-running by modifying transaction ordering and limiting MEV extraction [9, 31, 50, 51, 108, 109, 111]. Fair-ordering schemes provide arrival-order guarantees [18, 24, 59–63, 79, 110], while blind ordering hides transaction contents until ordering is finalized [58, 65]. Other approaches distribute sequencing through systems such as SUAVE [34], protect transaction contents using TEEs [11, 93], or employ batched threshold encryption [15, 16]. Protocol-level proposals such as PEPC [74], ePBS [86], and FOCIL [96] constrain proposer ordering power, while OFAs [52] such as MEV-Blocker [70] and MEV-Share [37] provide private order flow and MEV redistribution. PROF [5] further combines private ordering with incentives for profit-seeking validators. Despite these defenses, their effectiveness and underlying trust assumptions remain largely untested. We provide the first large-scale empirical study of sandwiching across multiple protection mechanisms, revealing widespread violations of these assumptions.

8 Concluding Discussion

Our study shows that sandwiches against protected order flow occur across four chains, spanning L1s and L2s with substantially different transaction-submission and mempool designs. Outside Solana, however, we observe only a handful of entities exploiting the exposures we identify, possibly reflecting limited profitability or a less mature extraction ecosystem. On Solana, we observe no considerable sandwich activity until late 2023, after which attacks grow rapidly and adapt as successive exposure paths are mitigated.

More broadly, our findings highlight the need for protection mechanisms that account for information exposure across the full transaction-submission pipeline rather than at a single layer. Thus, the attacks we observe may represent only a lower bound on the risk posed by such exposure if these mechanisms become more widely exploited.

OFAs. Ethereum's transaction-submission pipeline is fragmented across components whose threat models are not necessarily designed to compose. If a transaction originator submits the same signed transaction through multiple RPCs or OFAs, information revealed along one path can weaken the protections of another. Such multi-path submission should therefore be treated as a composition risk: OFAs and RPC providers should ensure that their disclosure mechanisms remain safe when combined. This is particularly important because wallets and applications may route transactions through multiple providers to improve inclusion guarantees or rebates, leaving users with limited control over the resulting submission path.

OFAs can further reduce unnecessary exposure by withholding transactions that are unlikely to generate meaningful back-run value. Simple heuristics, such as avoiding disclosure for trades in tokens with no alternative liquid trading pool, could eliminate many cases in which users are exposed without a corresponding arbitrage opportunity. Finally, OFAs increasingly operate on L2s such as Base, Arbitrum, Robinhood chain, where signed transactions may be provided directly to integrated searchers [12]. Such designs can reintroduce pre-inclusion transaction visibility into environments whose sequencer-based submission model would otherwise keep pending order flow private.

Encrypted Mempools and TEEs. Encrypted mempools can mitigate many of the exposure paths identified in this study [29, 71, 84], with recent constructions making such designs increasingly practical [1–3, 13–16, 19, 21–23, 30, 33, 39, 45, 94, 106]. By hiding transaction contents until inclusion, they can prevent bots and block producers from front-running based on pre-inclusion transaction information. However, exposure before encryption, for example at the RPC or application layer, remains possible [40]. Delayed disclosure can also hinder timely price discovery and back-running, potentially reducing or delaying OFA rebates. TEE-based designs offer a different tradeoff: they allow searchers to operate on full transaction contents inside an isolated execution environment [36], preserving back-running opportunities while limiting information exposed outside the TEE for front-running.

Predictable Behavior. As we have seen on Base, predictable transaction patterns can be front-run even without any exposure through the submission-pipeline. As long as past on-chain behavior remains observable, sufficiently regular activity can reveal information about future transactions. Although users may in principle avoid such patterns, doing so consistently is often impractical. More comprehensive protection may therefore require stronger forms of transaction privacy, potentially including designs that conceal transaction intent or relevant on-chain activity until execution.

Users ultimately need protection that holds in practice. We hope that exposing this darker corner of the dark forest, where supposedly protected transactions can still be front-run, helps drive the development of more effective defenses.

Acknowledgments

We would like to thank DataAlways, Jonathan Passerat-Palmbach, Arthur Bagourd, Antony Denyer, and Albin Mamuti for their valuable comments and feedback.

References

[1] Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. Cryptology ePrint Archive, Paper 2025/2115, 2025. URL: https://eprint.iacr.org/2025/2115.

[2] Amit Agarwal, Sourav Das, Babak Poorebrahim Gilkalaye, Peter Rindal, and Victor Shoup. BTX: Simple and Efficient Batch Threshold Encryption. Cryptology ePrint Archive, Report 2026/754, 2026. URL: https://eprint.iacr.org/2026/754.

[3] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. Cryptology ePrint Archive, Paper 2024/1575, 2024. URL: https://eprint.iacr.org/2024/1575.

[4] Axiom. Axiom. https://axiom.trade, 2026. Accessed: 2026-08-16.

[5] Kushal Babel, Nerla Jean-Louis, Yan Ji, Ujval Misra, Mahimna Kelkar, Kosala Yapa Mudiyanselage, Andrew Miller, and Ari Juels. PROF: Protected Order Flow in a Profit-Seeking World. In *11th IEEE European Symposium on Security and Privacy, EuroS&P 2026, Lisbon, Portugal, July 6-10, 2026*, pages 398–418. IEEE, 2026. URL: https://doi.org/10.1109/EuroSP68448.2026.00034, doi:10.1109/EUROSP68448.2026.00034.

[6] Base. Base Incident Report 1. https://status.base.org/incidents/m9fnx4p3bhp5, 2026. Accessed: 2026-08-16.

[7] Base. Base Incident Report 2. https://status.base.org/incidents/czj7m0j86m09, 2026. Accessed: 2026-08-16.

[8] Base Github. Base Tx Pool Leak Github. https://github.com/base/node/pull/96, 2026. Accessed: 2026-08-16.

[9] Carsten Baum, James Hsin-yu Chiang, Bernardo David, Tore Kasper Frederiksen, and Lorenzo Gentile. SoK: Mitigation of Front-Running in Decentralized Finance. In Shin'ichiro Matsuo, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez Hernandez, Sam Werner, Thomas Haines, Aleksander Essex, Andrea Bracciali, and Massimiliano Sala, editors, *Financial Cryptography and Data Security. FC 2022 International Workshops - CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers*, volume 13412 of *Lecture Notes in Computer Science*, pages 250–271. Springer, 2022. doi:10.1007/978-3-031-32415-4_17.

[10] beaconcha.in. beaconcha.in Light: Ethereum Mainnet Beacon Explorer. https://light-mainnet.beaconcha.in/, 2026. Powered by Dora; accessed: 2026-08-21.

[11] Iddo Bentov, Yan Ji, Fan Zhang, Yunqi Li, Xueyuan Zhao, Lorenz Breidenbach, Philip Daian, and Ari Juels. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1521–1538. ACM, 2019. doi:10.1145/3319535.3363221.

[12] Blink Labs. Blink Documentation. https://docs.blinklabs.xyz/blink, 2026. Accessed: 2026-08-05.

[13] Dan Boneh, Evan Laufer, and Ertem Nusret Tas. Threshold batch identity-based encryption without epochs. Cryptology ePrint Archive, Paper 2025/1254, 2025. URL: https://eprint.iacr.org/2025/1254.

[14] Dan Boneh, Rohit Nema, Arnab Roy, and Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674, 2026. URL: https://eprint.iacr.org/2026/674.

[15] Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Paper 2025/1419, 2025. URL: https://eprint.iacr.org/2025/1419.

[16] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3457–3476, Seattle, WA, August 2025. USENIX Association.

[17] BullX. BullX. https://bullx.io, 2026. Accessed: 2026-08-16.

[18] Christian Cachin, Jovana Micic, Nathalie Steinhauer, and Luca Zanolini. Quick Order Fairness. In *Financial Cryptography and Data Security, 26th International Conference, FC 2022*, volume 13411 of *Lecture Notes in Computer Science*, pages 316–333. Springer, 2022. doi:10.1007/978-3-031-18283-9_15.

[19] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. Cryptology ePrint Archive, Paper 2021/1423, 2021. URL: https://eprint.iacr.org/2021/1423.

[20] cdump. Evmole: Extracts function selectors, arguments, state mutability and storage layout from evm bytecode, even for unverified contracts, 2026. GitHub repository. URL: https://github.com/cdump/evmole.

[21] Andrea Cerulli, Aisling Connolly, Gregory Neven, Franz-Stefan Preiss, and Victor Shoup. vetKeys: How a blockchain can keep many secrets. Cryptology ePrint Archive, Paper 2023/616, 2023. URL: https://eprint.iacr.org/2023/616.

[22] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. Cryptology ePrint Archive, Paper 2024/669, 2024. URL: https://eprint.iacr.org/2024/669.

[23] Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024. URL: https://eprint.iacr.org/2024/1516.

[24] Andrei Constantinescu, Diana Ghinea, Lioba Heimbach, Zilin Wang, and Roger Wattenhofer. A Fair and Resilient Decentralized Clock Network for Transaction Ordering. In *27th International Conference on Principles of Distributed Systems, OPODIS 2023*, volume 286 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.OPODIS.2023.8.

[25] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and

Ari Juels. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.

[26] Dan Robinson, Georgios Konstantopoulos. Ethereum is a Dark Forest. https://www.paradigm.xyz/writing/ethereum-is-a-dark-forest, 2026. Accessed: 2026-08-16.

[27] DefiLlama. DefiLlama Chains. https://defillama.com/chains, 2026. Accessed: 2026-08-16.

[28] DefiLlama. DefiLlama DEXs. https://defillama.com/dexs, 2026. Accessed: 2026-08-16.

[29] Stefan Dziembowski, Sebastian Faust, and Jannik Luhn. Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Paper 2024/1981, 2024. URL: https://eprint.iacr.org/2024/1981.

[30] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. McFly: Verifiable encryption to the future made practical. Cryptology ePrint Archive, Paper 2022/433, 2022. URL: https://eprint.iacr.org/2022/433.

[31] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. In *Financial Cryptography and Data Security, FC 2019 International Workshops*, volume 11599 of *Lecture Notes in Computer Science*, pages 170–189. Springer, 2019. doi:10.1007/978-3-030-43725-1_13.

[32] ethPandaOps. Xatu Data. https://ethpandaops.io/data/xatu/, 2024. Accessed: 2026-08-21.

[33] Rex Fernando, Guru-Vamsi Policharla, Andrei Tonkikh, and Zhuolun Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Paper 2025/2032, 2025. URL: https://eprint.iacr.org/2025/2032.

[34] Flashbots. Suave: A single unified auction for value expression. https://writings.flashbots.net/the-future-of-mev-is-suave, 2023.

[35] Flashbots. Mempool Dumpster. https://github.com/flashbots/mempool-dumpster, 2024. Accessed: 2026-06-25.

[36] Flashbots. Enabling tee searching for protect fast mode, August 2025. The Flashbots Collective, accessed 2026-08-25. URL: https://collective.flashbots.net/t/enabling-tee-searching-for-protect-fast-mode/5219.

[37] Flashbots. MEV-Share Introduction. https://docs.flashbots.net/flashbots-mev-share/introduction, 2026. Accessed: 2026-08-05.

[38] Flashbots. MEV-Share: Understanding double-hash. https://docs.flashbots.net/flashbots-mev-share/searchers/event-stream#understanding-double-hash, 2026. Accessed: 2026-08-16.

[39] Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. tlock: Practical timelock encryption from threshold BLS. Cryptology ePrint Archive, Paper 2023/189, 2023. URL: https://eprint.iacr.org/2023/189.

[40] Pranav Garimidi, Joseph Bonneau, and Lioba Heimbach. On the limits of encrypted mempools. https://a16zcrypto.com/posts/article/limits-encrypted-mempools/, 2025. a16z crypto. Accessed: 2026-08-21.

[41] Nicole Gerzon, Ben Weintraub, Junbeom In, Alan Mislove, and Cristina Nita-Rotaru. Quantifying the Threat of Sandwiching MEV on Jito: A Measurement of Solana's Leading Validator Client. In Paul Barford, Aruna Balasubramanian, and Alberto Dainotti, editors, *Proceedings of the 2025 ACM Internet Measurement Conference, IMC 2025, Madison, WI, USA, 28-31 October 2025*, pages 937–943. ACM, 2025. doi:10.1145/3730567.3764493.

[42] GMGN. GMGN. https://gmgn.ai, 2026. Accessed: 2026-08-16.

[43] Krzysztof Gogol, Manvir Schneider, Jan Gorzny, and Claudio J. Tessone. How to Serve Your Sandwich? MEV Attacks in Private L2 Mempools. *CoRR*, abs/2601.19570, 2026. URL: https://doi.org/10.48550/arXiv.2601.19570, arXiv:2601.19570, doi:10.48550/ARXIV.2601.19570.

[44] Krzysztof Gogol, Manvir Schneider, and Claudio Tessone. When Priority Fails: Revert-Based MEV on Fast-Finality Rollups. *arXiv preprint arXiv:2506.01462*, 2025.

[45] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. Cryptology ePrint Archive, Paper 2025/2103, 2025. URL: https://eprint.iacr.org/2025/2103.

[46] Tivas Gupta, Mallesh M. Pai, and Max Resnick. The Centralizing Effects of Private Order Flow on Proposer-Builder Separation. In *5th Conference on Advances*

in Financial Technologies, AFT 2023, volume 282 of *LIPIcs*, pages 20:1–20:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.AFT.2023.20.

[47] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In Marie-José Montpetit, Aris Leivadeas, Steve Uhlig, and Mobin Javed, editors, *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC 2023, Montreal, QC, Canada, October 24-26, 2023*, pages 406–420. ACM, 2023. doi:10.1145/3618257.3624824.

[48] Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-Atomic Arbitrage in Decentralized Finance. In *45th IEEE Symposium on Security and Privacy, SP 2024*, pages 3866–3884. IEEE, 2024. doi:10.1109/SP54263.2024.00256.

[49] Lioba Heimbach, Ozan Solmaz, Burak Öz, and Christof Ferreira Torres. No Place to Hide: An Analysis on Protected Order Flow Sandwich Attacks. https://github.com/liobaheimbach/No-Place-to-Hide-An-Analysis-on-Protected-Order-Flow-Sandwich-Attacks, 2026. Accessed: 2026-09-21.

[50] Lioba Heimbach and Roger Wattenhofer. Eliminating Sandwich Attacks with the Help of Game Theory. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pages 153–167. ACM, 2022. doi:10.1145/3488932.3517390.

[51] Lioba Heimbach and Roger Wattenhofer. SoK: Preventing Transaction Reordering Manipulations in Decentralized Finance. In Maurice Herlihy and Neha Narula, editors, *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*, pages 47–60. ACM, 2022. doi:10.1145/3558535.3559784.

[52] Paul Janicot and Alex Vinyas. Private MEV Protection RPCs: Benchmark Study, 2025. URL: https://arxiv.org/abs/2505.19708, arXiv:2505.19708.

[53] Jina AI. Reader API. https://jina.ai/reader/, 2024. Service endpoint: https://r.jina.ai/; accessed: 2026-08-21.

[54] Jito Labs. Jito Labs suspends the mempool offered through the Jito Block Engine. https://x.com/jito_labs/status/1766228889888514501, 2024. Accessed: 2026-08-17.

[55] Jito Labs. jito-labs/searcher-examples: mempool subscription and its removal. https://github.com/jito-labs/searcher-examples, 2024. Commit `fc04568` (2022-07-26) ships a bot calling `subscribe_pending_transactions`; commit `367b7fe` "Remove Mempool from CLI" is authored 2024-03-08. Accessed: 2026-08-17.

[56] Jito Labs. Jito Relayer and Block Engine. https://jito-labs.gitbook.io/mev/searcher-services/recommendations, 2024. Accessed: 2026-08-17.

[57] Jupiter. Ultra Swap API Overview. https://developers.jup.ag/docs/ultra, 2026. Accessed: 2026-08-16.

[58] Alireza Kavousi, Duc Viet Le, Philipp Jovanovic, and George Danezis. Blindperm: Efficient MEV mitigation with an encrypted mempool and permutation. In Andrei Arusoaie, Emanuel Onica, Michael Spear, and Sara Tucci Piergiovanni, editors, *29th International Conference on Principles of Distributed Systems, OPODIS 2025, Iaşi, Romania, December 3-5, 2025*, volume 361 of *LIPIcs*, pages 36:1–36:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2025.36, doi:10.4230/LIPICS.OPODIS.2025.36.

[59] Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-Fair Consensus in the Permissionless Setting. In *9th ACM ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS 2022*, pages 3–14. ACM, 2022. doi:10.1145/3494105.3526239.

[60] Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489. ACM, 2023. doi:10.1145/3576915.3616658.

[61] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *Annual International Cryptology Conference*, pages 451–480. Springer, 2020.

[62] Aggelos Kiayias, Nikos Leonardos, and Yu Shen. Ordering transactions with bounded unfairness: Definitions, complexity and constructions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 34–63. Springer, 2024.

[63] Klaus Kursawe. Wendy, the Good Little Fairness Widget: Achieving Order Fairness for Blockchains.

In *2nd ACM Conference on Advances in Financial Technologies, AFT 2020*, pages 25–36. ACM, 2020. doi:10.1145/3419614.3423263.

[64] Jordan Leech. Solana Foundation removes certain operators from delegation program over malicious sandwich attacks. https://www.theblock.co/news/ecosystems/2024-06-10-solana-foundation-removes-certain-operators-from-delegation-program-over-malicious-sandwich-attacks-299244, 2024. The Block. Accessed: 2026-08-21.

[65] Rujia Li, Xuanwei Hu, Qin Wang, Sisi Duan, and Qi Wang. Transaction fairness in blockchains, revisited. *IEEE Transactions on Dependable and Secure Computing*, 2025.

[66] Zihao Li, Jianfeng Li, Zheyuan He, Xiapu Luo, Ting Wang, Xiaoze Ni, Wenwu Yang, Xi Chen, and Ting Chen. Demystifying DeFi MEV Activities in Flashbots Bundle. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 165–179. ACM, 2023. doi:10.1145/3576915.3616590.

[67] Davide Mancino and Davide Rezzoli. Sandwiched and Silent: Behavioral Adaptation and Private Channel Exploitation in Ethereum MEV. *CoRR*, abs/2512.17602, 2025. URL: https://doi.org/10.48550/arXiv.2512.17602, doi:10.48550/ARXIV.2512.17602.

[68] Marinade. MIP.9: Blocklisting malicious validators from the Stake Auction Marketplace to combat sandwich attacks. https://forum.marinade.finance/t/mip-9-blocklisting-malicious-validators-from-stake-auction-marketplace-to-combat-sandwich-attacks/1716, 2025. Accessed: 2026-08-21.

[69] Mempool.guru. Mempool.guru. https://mempool.guru/, 2023. Accessed: 2023-09-26.

[70] MEVBlocker. MEVBlocker Documentation. https://docs.mevblocker.io/, 2026. Accessed: 2026-08-05.

[71] Peyman Momeni, Sergey Gorbunov, and Bohan Zhang. FairBlock: Preventing blockchain front-running with minimal overheads. Cryptology ePrint Archive, Paper 2022/1066, 2022. URL: https://eprint.iacr.org/2022/1066.

[72] Monad Labs. Monad Documentation: Local Mempool and Transaction Forwarding. https://docs.monad.xyz, 2025. Accessed: 2026-06-25.

[73] Monad Labs. RaptorCast: Monad's Block Propagation Protocol. https://docs.monad.xyz, 2025. Accessed: 2026-06-25.

[74] Barnabé Monnot. Unbundling pbs: Towards protocol-enforced proposer commitments (pepc). Ethereum Research, 2022. URL: https://ethresear.ch/t/unbundling-pbs-towards-protocol-enforced-proposer-commitments-pepc/13879.

[75] Offchain Labs. The Arbitrum Sequencer. https://docs.arbitrum.io/how-arbitrum-works/sequencer, 2026. Accessed: 2026-08-05.

[76] OP-GETH Github. OP-GETH Tx Pool Leak Github. https://github.com/ethereum-optimism/op-geth/pull/118, 2026. Accessed: 2026-08-16.

[77] Optimism Foundation. OP Stack: Sequencing and the Transaction Pool. https://docs.optimism.io/, 2026. Accessed: 2026-08-05.

[78] Burak Öz, Christof Ferreira Torres, Christoph Schlegel, Bruno Mazorra, Jonas Gebele, Filip Rezabek, and Florian Matthes. Cross-Chain Arbitrage: The Next Frontier of MEV in Decentralized Finance. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(3):1–33, 2025. doi:10.1145/3771566.

[79] Burak Öz, Jonas Gebele, Parshant Singh, Filip Rezabek, and Florian Matthes. Playing the MEV Game on a First-Come-First-Served Blockchain. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2024, Dublin, Ireland, May 27-31, 2024*, pages 220–224. IEEE, 2024. doi:10.1109/ICBC59979.2024.10634397.

[80] Vabuk Pahari and Andrea Canidio. How Exclusive Are Ethereum Transactions? Evidence from Non-Winning Blocks. *CoRR*, abs/2509.16052, 2025. URL: https://doi.org/10.48550/arXiv.2509.16052, arXiv:2509.16052, doi:10.48550/ARXIV.2509.16052.

[81] Vabuk Pahari, Johnnatan Messias, and Christof Ferreira Torres. There Will Be Spam: Characterizing State-Invariant Transactions and Speculative MEV. *CoRR*, abs/2607.24172, 2026. URL: https://doi.org/10.48550/arXiv.2607.24172, arXiv:2607.24172, doi:10.48550/ARXIV.2607.24172.

[82] Santiago Palladino, Francisco Giordano, and Hadrien Croubois. Eip-1967: Proxy storage slots. Ethereum Improvement Proposals, 2019. URL: https://eips.ethereum.org/EIPS/eip-1967.

[83] Paradigm. cryo: Easy extraction of blockchain data to parquet, csv, json, or python dataframes, 2023. GitHub repository. URL: https://github.com/paradigmxyz/cryo.

[84] Jonathan Passerat-Palmbach. SoK: Encrypted mempools through the MEV lens. Cryptology ePrint Archive, Paper 2026/1643, 2026. URL: https://eprint.iacr.org/2026/1643, doi:10.4230/LIPIcs.AFT.2026.30.

[85] Photon. Photon. https://photon-sol.tinyastro.io, 2026. Accessed: 2026-08-16.

[86] Ethereum Improvement Proposals. Eip-7732: Enshrined proposer-builder separation. Ethereum Improvement Proposals, 2024. URL: https://eips.ethereum.org/EIPS/eip-7732.

[87] Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. An Empirical Study of DeFi Liquidations: Incentives, Risks, and Instabilities. In *ACM Internet Measurement Conference, IMC 2021*, pages 336–350. ACM, 2021. doi:10.1145/3487552.3487811.

[88] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 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*, pages 198–214. IEEE, 2022. doi:10.1109/SP46214.2022.9833734.

[89] sandwiched.me. Sandwich Attacks on Solana. https://sandwiched.me, 2025. Accessed: 2026-06-25.

[90] Hasret Ozan Sevim and Christof Ferreira Torres. Signals and Spoils: Speculative Oracle Extractable Value in the Era of Cross-Chain Interoperability. *arXiv preprint arXiv:2606.03434*, 2026.

[91] Solana Foundation. MEV Protection with Jito Dont-Front. https://solana.com/docs/defi/mev-protection, 2026. Accessed: 2026-08-13.

[92] Ozan Solmaz, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. Optimistic MEV in Ethereum Layer 2s: Why Blockspace Is Always in Demand. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies, AFT 2025, Pittsburgh, PA, USA, October 8-10, 2025*, volume 354 of *LIPIcs*, pages 28:1–28:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.AFT.2025.28, doi:10.4230/LIPICS.AFT.2025.28.

[93] Chrysoula Stathakopoulou, Signe Rüsch, Marcus Brandenburger, and Marko Vukolic. Adding Fairness to Order: Preventing Front-Running Attacks in BFT Protocols using TEEs. In *40th International Symposium on Reliable Distributed Systems, SRDS 2021*, pages 34–45. IEEE, 2021. doi:10.1109/SRDS53918.2021.00013.

[94] Sora Suegami, Shinsaku Ashizawa, and Kyohei Shibano. Constant-Cost Batched Partial Decryption in Threshold Encryption. Cryptology ePrint Archive, Paper 2024/762, 2024. URL: https://eprint.iacr.org/2024/762.

[95] Justin Sun. Statement on front-running transactions on the TRON network. https://x.com/justinsuntron/status/1830984536244789641, 2024. Accessed: 2026-08-21.

[96] Thomas Thiery, Francesco D'Amato, Julian Ma, Barnabé Monnot, Terence Tsao, Jacob Kaufmann, and Jihoon Song. Eip-7805: Fork-choice enforced inclusion lists (focil). Ethereum Improvement Proposals, 2024. URL: https://eips.ethereum.org/EIPS/eip-7805.

[97] Christof Ferreira Torres, Ramiro Camino, and Radu State. Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 1343–1359. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/torres.

[98] Christof Ferreira Torres, Albin Mamuti, Ben Weintraub, Cristina Nita-Rotaru, and Shweta Shinde. Rolling in the Shadows: Analyzing the Extraction of MEV Across Layer-2 Rollups. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 2591–2605. ACM, 2024. doi:10.1145/3658644.3690259.

[99] Trojan. Trojan. https://trojan.com, 2026. Accessed: 2026-08-16.

[100] TRON DAO. TRON Whitepaper: DPoS Consensus and Transaction Ordering. https://tron.network, 2024. Accessed: 2026-06-25.

[101] Hristina Vasileva. Jito bans 15 additional validators after data emerges of widespread sandwich attacks. https://www.cryptopolitan.com/jito-bans-15-additional-validators-after-data-emerges-of-widespread-sandwich-attacks/, 2025. Cryptopolitan. Accessed: 2026-08-21.

[102] Wenhao Wang, Aditya Saraf, Lioba Heimbach, Kushal Babel, and Fan Zhang. Blockspace Under Pressure: An Analysis of Spam MEV on High-Throughput Blockchains. *CoRR*, abs/2604.00234, 2026. URL:

https://doi.org/10.48550/arXiv.2604.00234, arXiv:2604.00234, doi:10.48550/ARXIV.2604.00234.

[103] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. Cyclic Arbitrage in Decentralized Exchanges. In *Companion Proceedings of the Web Conference 2022, WWW 2022*, pages 12–19. ACM, 2022. doi:10.1145/3487553.3524201.

[104] Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. A Flash(bot) in the Pan: Measuring Maximal Extractable Value in Private Pools. In Chadi Barakat, Cristel Pelsser, Theophilus A. Benson, and David R. Choffnes, editors, *Proceedings of the 22nd ACM Internet Measurement Conference, IMC 2022, Nice, France, October 25-27, 2022*, pages 458–471. ACM, 2022. doi:10.1145/3517745.3561448.

[105] Fei Wu and Burak Öz. To wait or to probe: Arbitrage competition on high-throughput blockchains, 2026. URL: https://arxiv.org/abs/2606.00720, arXiv:2606.00720.

[106] Saisi Xiong and Jie Chen. Efficient Batched IBE from Lattices in the Standard Model. Cryptology ePrint Archive, Report 2025/2158, 2025. URL: https://eprint.iacr.org/2025/2158.

[107] Anatoly Yakovenko. Gulf Stream: Solana's Mempool-less Transaction Forwarding Protocol. https://solana.com/news/gulf-stream--solana-s-mempool-less-transaction-forwarding-protocol, 2019. Accessed: 2026-08-16.

[108] Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. SoK: MEV Countermeasures. In *Proceedings of the 2024 Workshop on Decentralized Finance and Security, DeFi 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 21–30. ACM, 2024. doi:10.1145/3689931.3694911.

[109] Haoqian Zhang, Louis-Henri Merino, Vero Estrada-Galiñanes, and Bryan Ford. Flash Freezing Flash Boys: Countering Blockchain Front-Running. In *42nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops 2022*, pages 90–95. IEEE, 2022. doi:10.1109/ICDCSW56584.2022.00026.

[110] Yunhao Zhang, Srinath T. V. Setty, Qi Chen, Lidong Zhou, and Lorenzo Alvisi. Byzantine Ordered Consensus without Byzantine Oligarchy. In *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020*, pages 633–649. USENIX Association, 2020. URL: https://www.usenix.org/conference/osdi20/presentation/zhang-yunhao.

[111] Liyi Zhou, Kaihua Qin, and Arthur Gervais. A2MM: Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges. *CoRR*, abs/2106.07371, 2021. URL: https://arxiv.org/abs/2106.07371, arXiv:2106.07371.

[112] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc Viet Le, and Arthur Gervais. High-Frequency Trading on Decentralized On-Chain Exchanges. In *42nd IEEE Symposium on Security and Privacy, SP 2021*, pages 428–445. IEEE, 2021. doi:10.1109/SP40001.2021.00027.

A Data Collection

We provide a detailed description of our data collection pipeline in the following.

A.1 EVM Chains

We collect swap data from five EVM chains by decoding the respective log events, i.e., Ethereum, Base, Arbitrum, Monad, and Tron. Our coverage spans Uniswap V2, V3 and V4 together with their identical forks, i.e., SushiSwap, PancakeSwap and SunSwap, as well as Aerodrome V1 on Base, which gives us the largest automated market maker on each chain [28]. We collect all this over each chain's RPC interface, along with the transaction metadata, fees, block builders, and per-sender transaction histories our analysis relies on.

For Ethereum pool availability and back-run analysis, we construct a catalog of pool creation and registration events. Using an Ethereum RPC and cryo [83], we scan 22 creation event topics covering Uniswap V1–V4-style deployments, Balancer V2 and V3, several Curve factory generations, DODO, Mavrick, Sushiswap Trident, Kyberswap, and Bancor V3. Uniswap V1 is scanned from its factory deployment at block 6,627,917. All others are scanned starting from 10,000,835, the canonical Uniswap V2 factory deployment block. The collection ends on June 30th, 2026. Of the 737,484 collected rows, 733,257 describe two-token pool creations, 581 describe multi-token pools, and 3,646 remain unresolved or excluded. Our pool-count and back-run analysis use only two token pools.

A.1.1 Mempool Labeling on Ethereum

For Ethereum, we use the Flashbots public mempool archive [35] to assess whether a swap is public or private. We label a swap **public** when it was seen in the public mempool prior to inclusion, **private** when it was not and the archive has good data quality in the relevant hour, and **unknown** when it is absent during an hour with bad data quality. An hour has bad data quality when the archive holds fewer than half the transactions we expect for it, i.e., the median count for the same hour of day across the $\pm$10-day window around it. To guard against false private classifications caused by gaps in the public archive, we additionally cross-check transactions labeled private against an internal Flashbots transaction collection. Any transaction observed there before inclusion is removed from the private set. Throughout, we treat only the swaps we label private as protected order flow. We note that this is the conservative choice, as it leaves every unknown swap outside the protected order flow and keeps us from labeling a public swap as private. The Flashbots archive begins in August 2023, so for the first weeks of our window we rely on mempool sightings obtained from the mempool.guru project instead [69]. Appendix D reports the resulting split.

Data Source	Coverage Period
MEV-Share	
Dune (`dune.flashbots.dataset_protect_transactions`)	2022-02 – 2026-05
Flashbots Data Browser	2022-10 – 2026-03
Historical event-stream API	2024-02 – 2026-07
MEVBlocker	
Dune (`mevblocker.raw_bundles`)	2023-04 – 2026-07
Blink	
Bundle data from BuilderNet	2025-08 – 2026-07
Merkle	
Dune (`dune.flashbots.merkle`)	2025-01 – 2026-03

Table 11: Data sources used to label Ethereum sandwich victims by OFA provider, and the coverage period of each.

A.1.2 OFA Labeling on Ethereum

For each identified victim transaction on Ethereum, we determine whether it used an OFA solution. OFA routing is an RPC-level decision made by the submitting user wallet or bot, so it cannot in general be recovered from blockchain data alone: a transaction sent to an OFA and one broadcast directly to the public mempool are, once included in a block, indistinguishable at the protocol level. Some OFA operators nonetheless publish auxiliary datasets—query interfaces, historical archives, or partial-disclosure hint streams—that let us reconstruct routing after the fact, though several of these channels are no longer actively maintained and their coverage windows do not always extend to the present. Table 11 summarizes, for each provider, the data source we rely on and the period it covers.

For MEV-Share specifically, three such sources are available: a Dune-indexed record submissions, Flashbots' public data browser S3 archive[3], and the event-stream API[4]. We find that the three sources provide complementary coverage, with each capturing victims that are absent from the others. We therefore take the union of all three rather than relying on any single source, which would otherwise undercount MEV-Share coverage. For MEVBlocker, we rely on the bundle data CoW Protocol uploads to Dune. For Blink, there are no public records of its order flow, so we instead use BuilderNet's internal bundle data, provided by Flashbots. For Merkle, we use bundle data on Dune uploaded by Flashbots, which covers a limited window.

Coverage Limitations. Our OFA attribution is incomplete. Coverage differs across providers and over time. In particular, our Blink data consists of bundles observed by BuilderNet and is available only from August 2025 onward, so we cannot identify earlier Blink submissions or bundles that were not received by BuilderNet. More generally, the external datasets used for all four OFAs may omit transactions even within their

[3] Flashbots data browser: https://flashbots-data.s3.us-east-2.amazonaws.com/index.html

[4] MEV-Share event stream: https://docs.flashbots.net/flashbots-mev-share/searchers/event-stream

nominal coverage periods. We therefore treat OFA labels as positive evidence of exposure rather than exhaustive routing ground truth where the absence of a label does not imply that a transaction did not pass through an OFA.

A.2 Reorged Blocks

We query Xatu [32] for Ethereum `chain_reorg` events during our study period. These events are observer-local, so several nodes may report the same reorg. Grouping identical $(slot, oldHead, newHead)$ tuples yields 63,842 distinct transitions. These cover only what Xatu's nodes observed, so the network-wide count may be higher. From each old head, we follow parent roots to the eventually finalized canonical chain. We exclude 71 roots that ultimately become canonical and compare each remaining execution-block hash with the canonical hash at the same height. This leaves 14,398 unique, eventually non-canonical execution payloads, of which 14,397 have complete ancestry. For all but one, the available Xatu metadata allow us to trace the corresponding beacon ancestry back to a finalized-canonical ancestor. For the remaining root, the parent-chain metadata stop before reaching such an ancestor.

We retrieve their transaction lists from Dora [10], using Jina [53] as transport fallback. We validate each body's execution-block hash, height, parent hash and transaction count against Xatu. We recover 9,493 bodies (65.9%), containing 1,831,716 transaction-inclusion occurrences. We were not able to obtain the remaining 4,905 bodies with publicly available data sources.

A.3 Solana

We collect swap data from Solana by decoding each transaction's instructions and token-balance deltas, as Solana offers no event logs. Our coverage spans Raydium (V4, CLMM, CPMM), Orca Whirlpools, Meteora (DLMM, DAMM v2), Pump.fun (bonding curve, PumpSwap), SolFi v1, and Lifinity v2, which gives us the largest automated market makers on Solana [28]. We collect all this through per-slot RPC calls, along with the slot leaders, the SPL token transfers, the transaction fees, the Jito tips, the application fees, and the routing programs.

B Per-Chain Detection Details

The heuristics of Section 4 share one pipeline. This appendix records the parameters and adaptations that differ by chain.

Amount Match (H1). On the EVM chains we set the tolerance to $\varepsilon = 0.01$ on both sides, and a transaction that trades in both directions of the pool at once cannot serve as a leg. Solana runs detection with a relaxed lower bound, i.e., $A_{b,in} \geq 0.5 A_{f,out}$, so that we do not miss a back leg cashed out over several transactions. We still require the exit to reach 99% of the front output. It reaches it in one of two ways. Either the back legs inside the window sum to at least 99%, or the wallet sells at least 99% of the position later on, i.e., a staged exit.

Victims (H1). A victim is any swap in the direction of the front leg by another wallet, and not itself a candidate leg, that sits strictly between the first front leg and the first back leg. A bot can wrap one victim in two nested round trips on the same pool. We then credit only the inner pair, so a bot never counts the same victim twice. Two different bots that each sandwich the same victim both count, since that is competition for one target. Solana is stricter. There a victim must trade at least 1% of the front leg's size and execute at a worse price than the attacker's front leg, and we ignore SOL-paying fronts below 0.01 SOL as dust.

Window and Types (H1). The EVM window spans five blocks between the front and the back leg. On Solana a leader produces four consecutive slots, so we bound the gap in *leader rotations* instead, i.e., a back leg pairs with a front leg up to two leaders earlier, which lets an attack span three leaders in total. In both regimes a sandwich is **tight** when it consists of exactly three consecutive transactions in one block, or slot: the front leg, the victim, and the back leg. It is **within-block wide** when it stays inside one block with more transactions between the legs, and **cross-block wide** otherwise.

Entity Resolution (H3). On the EVM chains we resolve every leg and victim to the account that signed it, and we retain bots of three kinds. A *single sender* signs all of its legs itself. A *recurring sender pair* splits them across two accounts, one for the front and one for the back transaction, that call the same address in every sandwich. A *bot contract* is the receiver address of both transactions and the trader of the swap log, in every profitable sandwich we attribute to it.

We count only the sandwiches our detection links, and every candidate must clear the count and the rate floor. Note that on Ethereum we test the rate on protected flow alone. The density is the number of distinct attributed leg transactions divided by all swaps the accounts or the contract make in our study window. We attribute overlapping sandwiches to the candidate contract. A sender or a pair keeps its remaining sandwiches as a separate bot only when that remainder clears every floor. We label a pair by the bot address, while its membership and density stay pair-based.

On Solana the bot is the `trader` wallet for **vanilla** sandwiches. For **evasive** sandwiches the front wallet transfers the bought token to the back wallet inside the front transaction. We attribute the sandwich to the front wallet.

Victim Visibility (Ethereum). On Ethereum we additionally label the visibility of a transaction in the mempool (cf. Appendix A). A protected order flow sandwich then has at least one private victim transaction. We apply the full H3 filter to this subset, i.e., at least 100 protected sandwiches, at least 60% of them profitable, and a density of at least 25%. Density

counts protected-sandwich legs over all study-window swap rows of the bot. We further require a private-victim share of 66%, where the numerator counts only the sandwiches whose victims are all private and the denominator counts all of the bot's sandwiches.

C Threshold Sensitivity

The selection rule carries four floors, i.e., a total sandwich count of 100, a profit rate of 60%, a density of 25%, and on Ethereum a private-victim share of 66%. In the following, we report how the retained population moves when each floor varies. Table 12 sweeps density against the count floor for every chain, Table 13 sweeps the profit rate and the private-victim share, and Table 14 varies the amount-match tolerance ε.

Count. The count floor gives the two ratio gates evidence to work with. An address with two profitable sandwiches out of two scores a perfect rate and a high density on what may well be coincidence. Lowering the floor to 50 leaves the Ethereum private scope and Tron unchanged, adds 37 addresses on public Ethereum, admits two 74-sandwich bots on Base, and adds 3,361 bots on Solana that together carry only 0.2 million further sandwiches. Raising it cuts real bots, e.g., at 500 Ethereum loses one of its seven bots and Tron four of its twelve. We set the floor at 100 because it sits exactly between these regimes, i.e., lowering it only adds marginal addresses and raising it starts to cost real bots. The two 74-sandwich Base bots are the price, i.e., they clear every other gate and plausibly attack, and we exclude them to guard against coincidence.

Profit Rate. The profit-rate floor asks whether a bot mostly succeeds. Ethereum retains eight entities at 0% and 25%, and seven at 50–75%, wile Tron retains twelve throughout that range (Table 13). On Base, 144 entities clear the count and density floors at 0%, 14 at 25%, 5 at 50%, and the four reported entities at 60%, whose rates run 93.9–100%. At 75%, a fifth sender qualifies because its overlapping contract no longer passes candidacy. On Solana bots spread continuously across the range, i.e., 10,248 pass at 0% and 2,925 at 90%, and no threshold separates two populations. We place the floor at 60%, where the low-profit Base candidates are excluded while all reported EVM entities remain.

Density. The density floor separates dedicated sandwich bots from high-frequency traders whose two-way flow matches the sandwich structure incidentally. The Ethereum set is stable from 25% to 60% and admits one further entity at 10%. Tron loses a 10,438-sandwich entity with a density of 31.7% once the floor reaches 33%, and Base loses the 239-sandwich campaign bot with a density of 51.5% at 60%. On Solana the floor does most of the selection and cuts 14,808 candidate bots to 8,631 at the operating point. We set the floor at 25% because every entity we verified by hand clears it with a wide margin.

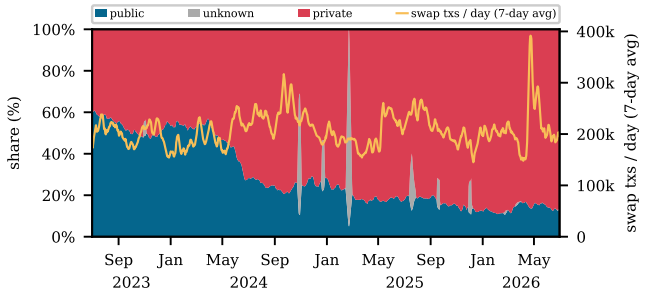


Figure 10: Ethereum mempool visibility per day. Shaded area, share by visibility (left axis, stacked to 100%). Line, daily swap count (right axis) smoothed over seven days.

On the EVM chains, the two lowest genuine densities in our study are the 31.7% entity on Tron and the 51.5% entity on Base named above.

Private-Victim Share. The private-victim share floor applies to Ethereum alone, where we observe victim visibility, and it separates the bots that specialize on protected order flow from public mempool bots with incidental private victims. This floor binds, i.e., 31 entities pass every other gate without a share requirement, 15 remain at 50%, and 7 at 66% (Table 13). The floor further sits on a plateau. The set of seven is unchanged up to 80% and the retained entities carry shares of 89.8–96.8%, so two thirds selects the specialized population.

Amount-Match Tolerance. The tolerance $\varepsilon = 0.01$ acts at detection time and defines which round trips count as a sandwich, whereas the four floors filter the detected entities afterwards. Table 14 re-filters the selected EVM sandwiches without rerunning detection or entity selection. Tightening the stored amount-match condition tenfold removes about 0.2% of these attacks at most on the EVM chains. Solana runs with the relaxed bound of Appendix B and is not part of this sweep.

Taken together, every reported EVM bot clears the count and rate floors with a margin, and these floors exist to exclude degenerate candidates. The tolerance barely moves the counts. The two floors that select, i.e., the density on all chains and the private-victim share on Ethereum, both sit on plateaus of the sweep, below every verified bot and above the excluded population.

D Ethereum Mempool Visibility

For Ethereum, our analysis of protected order flow sandwiches covers only swaps that did not enter the public mempool prior to inclusion in a block. Figure 10 shows how the visibility of swaps evolves over our window. The public share starts at 60% in mid 2023 and falls below 20% by mid 2026. We note that our mempool data has periods of poor quality, i.e., the public sources do not reliably record what was gossiped in those hours. We mark the transactions of such hours as

Density ≥	SWs ≥50	SWs ≥100	SWs ≥500	SWs ≥1000
10%	9 / 31,095	8 / 31,018	6 / 30,248	4 / 28,457
25%	7 / 30,607	**7 / 30,607**	6 / 30,248	4 / 28,457
33%	7 / 30,607	7 / 30,607	6 / 30,248	4 / 28,457
50%	7 / 30,607	7 / 30,607	6 / 30,248	4 / 28,457
60%	7 / 30,607	7 / 30,607	6 / 30,248	4 / 28,457

(a) Ethereum (Private Scope)

Density ≥	SWs ≥50	SWs ≥100	SWs ≥500	SWs ≥1000
10%	278 / 2.74M	237 / 2.74M	125 / 2.71M	96 / 2.69M
25%	232 / 1.79M	**195 / 1.79M**	101 / 1.77M	76 / 1.75M
33%	208 / 0.76M	174 / 0.76M	87 / 0.74M	66 / 0.72M
50%	163 / 0.58M	134 / 0.57M	64 / 0.56M	46 / 0.55M
60%	131 / 0.47M	105 / 0.47M	49 / 0.45M	36 / 0.45M

(b) Ethereum (Public Scope)

Density ≥	SWs ≥50	SWs ≥100	SWs ≥500	SWs ≥1000
10%	14 / 38,755	13 / 38,694	8 / 37,421	6 / 36,022
25%	12 / 38,567	**12 / 38,567**	8 / 37,421	6 / 36,022
33%	11 / 28,129	11 / 28,129	7 / 26,983	5 / 25,584
50%	11 / 28,129	11 / 28,129	7 / 26,983	5 / 25,584
60%	11 / 28,129	11 / 28,129	7 / 26,983	5 / 25,584

(c) Tron

Density ≥	SWs ≥50	SWs ≥100	SWs ≥500	SWs ≥1000
10%	51 / 10,085	39 / 9,144	2 / 1,448	0 / 0
25%	6 / 2,037	**4 / 1,889**	2 / 1,448	0 / 0
33%	6 / 2,037	4 / 1,889	2 / 1,448	0 / 0
50%	4 / 1,889	4 / 1,889	2 / 1,448	0 / 0
60%	3 / 1,650	3 / 1,650	2 / 1,448	0 / 0

(d) Base

Density ≥	SWs ≥50	SWs ≥100	SWs ≥500	SWs ≥1000
10%	21,642 / 35.4M	14,808 / 34.9M	5,623 / 32.8M	3,539 / 31.3M
25%	11,992 / 28.3M	**8,631 / 28.0M**	3,648 / 26.9M	2,345 / 26.0M
33%	9,178 / 24.5M	6,449 / 24.3M	2,843 / 23.5M	1,822 / 22.7M
50%	5,016 / 18.1M	3,582 / 18.0M	1,508 / 17.6M	970 / 17.2M
60%	3,086 / 15.7M	2,297 / 15.6M	883 / 15.3M	558 / 15.0M

(e) Solana

Table 12: Retained population under variation of the density and sandwich-count floors, by chain. Note that public Ethereum only uses swap-log identities. Every cell is bots / sandwiches (M for millions), with the chosen operating point (density ≥ 25%, at least 100 sandwiches) in bold. The profit-rate floor is held at 60% throughout and Ethereum's private-victim share at its operating point of 66%. Table 13 varies those two. Arbitrum and Monad are omitted, as neither retains a bot anywhere on the grid.

unknown, unless they were observed, and keep them outside the protected scope throughout the analysis, which is the conservative choice.

E Ethereum Public Mempool Sandwiches

To illustrate the difference between protected order flow sandwiches and the public mempool sandwiches studied in prior

Profit Rate ≥	0%	25%	50%	60%	75%	90%
Ethereum (Private)	8	8	7	**7**	7	4
Tron	12	12	12	**12**	12	11
Base	144	14	5	**4**	5	4
Solana	10,248	10,153	9,162	**8,631**	6,545	2,925
Private-Victim Share ≥ **(Ethereum Only)**	**0%**	**50%**	**66%**	**80%**	**90%**	**95%**
Ethereum (Private)	31	15	**7**	7	6	3

Table 13: Retained population under variation of the two floors Table 12 holds fixed, with density and sandwich count at their operating point. Cells show the number of bots and the chosen floor is in bold.

	ε=0.001	ε=0.0025	ε=0.005	ε=0.01
Ethereum (private)	7 / 30,575	7 / 30,584	7 / 30,587	**7 / 30,607**
Tron	12 / 38,486	12 / 38,511	12 / 38,530	**12 / 38,567**
Base	4 / 1,889	4 / 1,889	4 / 1,889	**4 / 1,889**

Table 14: Post-filter sensitivity to the amount-match tolerance ε within the selected EVM population. Cells are represented bots / retained sandwiches and the chosen value is in bold.

work, we turn to the latter in the following. Figure 11 plots the daily number of *public* Ethereum sandwiches and their split between tight and within-block wide. Only 0.2% of public sandwiches span a block boundary. Within-block wide attacks are common at first, and from late 2024 onward the majority are tight. The daily count drops sharply over the same period. Further, the wide attack classification is misleading as the associated sandwiches generally take the following structure:

Index	Role	Pool
170	Front-Run	*Both pools across one transaction*
171	Victim	WETH/CTM
172	Victim	USDT/H
173	Back-Run	*Both pools across one transaction*

The attacker sends one front transaction and one back transaction that serve two pools at once, i.e., a bundle of two sandwiches, likely to save gas. Detection records one sandwich per pool, and each of them looks wide because the other pool's victim sits between its own legs. As the daily sandwich count falls, the opportunities to sandwich several pools in one block fall with it, which could explain why the wide pattern disappears.

Comparing this with Figure 2a, two things stand out. Public sandwiches are far more numerous than those on protected order flow, and a much larger share of them is tight. An attacker there sees the victim and bundles both legs around it and likely submits this bundle to an MEV auction, so the higher precision is what we would expect.

Figure 12a shows the distance from the front transaction to the victim and from the victim to the back transaction. The

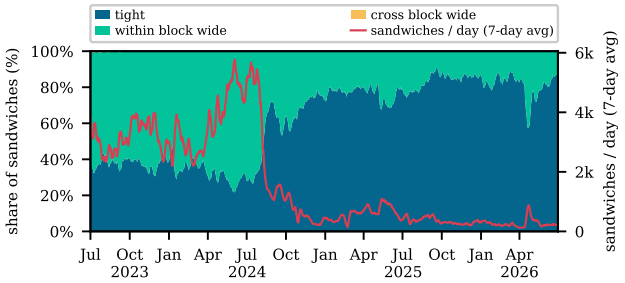
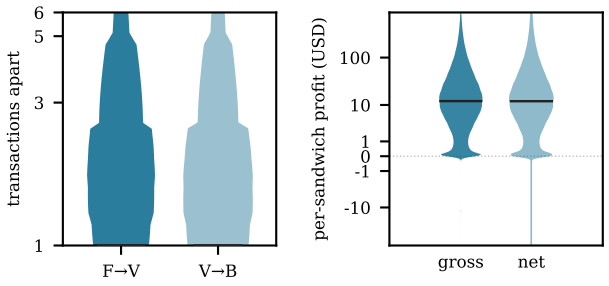


Figure 11: Public sandwich attacks per day on Ethereum. Shaded area, attack-type composition (left axis, stacked to 100%). Line, daily count (right axis). Both smoothed over seven days.



(a) (b)

Figure 12: Ethereum public-victim sandwiches (1,786,785 attacks, 195 bots). a Transaction-index distance from the front leg to the first victim (F→V) and from that victim to the back leg (V→B). b Per-sandwich profit, gross and net of gas.

two sides are symmetric, in line with the bundled example above, and the median is one transaction, i.e., the legs sit directly around the victim. Protected order flow sandwiches spread far wider (cf. Figure 3). Per-attack profit looks much the same across the two, at a public median of around $10 (Figure 12b) against a comparable median on protected flow (cf. Figure 15).

The difference lies in the share of attacks that turn a profit. Table 15 reports the per-bot profitability ratio for the Ethereum public bots, at 98.9% across all of them, against 91.9% on protected flow. Public attacks act on full visibility of the victim and land exactly, whereas attacks on protected flow work from an incomplete picture (about the position of the transaction in the block) and carry the corresponding risk.

Finally, the number of bots differs dramatically, at 195 on the public mempool against seven bots run by a single entity on protected flow. The volume difference likely accounts for part of this, i.e., 1.79 million public attacks against 30,607 protected ones.

Bot	Sandwiches	Profitable	Density
0x6b7...a80	985,291	98.7%	29.3%
0x000...8b8	218,842	100.0%	71.9%
0x473...000	70,691	100.0%	50.3%
0x429...e63	64,332	91.2%	49.2%
0x000...3d1	52,632	100.0%	65.4%
0x00f...c43	28,857	98.4%	34.7%
0x8af...a62	24,819	100.0%	81.0%
0x6e0...0d6	18,369	100.0%	86.4%
0xe44...b6c	17,271	100.0%	28.8%
0x000...b40	15,087	99.9%	36.4%
All 195 Bots	**1,786,785**	**98.9%**	—

Table 15: Profitability ratio of Ethereum public-victim real bots (top ten by sandwich count). Profitable, share of the bot's sandwiches with a positive token-level round trip. Density, sandwich-leg share of the bot's swaps.

Slot	Role	Swapping Account	Amount	Transaction
329,378,888	Front-Run	FLg7qbsM	0.750000 SOL	4Rdtny5T
329,378,888	Transfer	FLg7qbsM → 7KGtS9vd	130.15M tok	(same tx)
329,378,889	Victim	F1M6SgP9	0.010000 SOL	2SnDsFPC
329,378,890	Back-Run	7KGtS9vd	0.750830 SOL	4QrrxiYy

Table 16: Cross-trader sandwich on Raydium CPMM pool 2AMKkPfx. The front buy is executed by FLg7qbsM and the back sell by 7KGtS9vd, which returns the full 130,151,136 tokens the front acquired. The front transaction itself hands those tokens to the back trader in a third instruction, which is the on-chain link between the two accounts. Profit 0.000830 SOL.

F Sandwich Attack Structure on Solana

In the following, we illustrate the shapes a sandwich attack takes on Solana with one example each, other than the vanilla sandwich attacks we also observe on the other chains.

Separate Account. Table 16 shows an evasive attack, where the front and the back leg run on separate accounts and the front transaction hands the bought tokens to the back trader, which is the only on-chain link between the two. It nets 0.000830 SOL.

Split Leg. Table 17 shows a split back leg, where two sells unwind one buy and their summed input matches the front output, for 0.122 SOL.

Staged Exit. Table 18 shows a staged exit, where the attacker returns 71.4% of the position in the front's own slot and sells the remainder over the following 120 slots, closing 99.99% of it for a net 20.767 SOL.

We note that the first two examples most likely take this shape to escape detection, as each defeats a detector that pairs one buy with one sell on a single account within one slot or across neighboring slots [89]. The staged exit is a split back

Slot	Role	Signer	Amount	Transaction
311,083,629	Front-Run	FnyFZ6v8	37.13M tok	44htaJt3
311,083,629	Victim	HNCpJeAq	0.032 SOL	5EzgxMZm
311,083,630	Back-Run 1	FnyFZ6v8	18.57M tok	5UH8CYfx
311,083,632	Back-Run 2	FnyFZ6v8	18.57M tok	64SBkK61

Table 17: Split back leg on Pump AMM pool 6HKFa8i1. The front buys 37,131,550 tokens for 2.180 SOL and each back leg sells exactly half of them, 18,565,775, so the two sells together return the full position. Profit 0.122 SOL.

Slot	Role	Signer	Amount	Transaction
426,772,197	Front-Run	2mqrindM	6.716 SOL	HDwUyZyD
426,772,197	Victim	9mjtNH9v	64.0M tok	47bpB79Z
426,772,197	Victim	5brv79eF	205.7M tok	brrxoSyy
426,772,197	Victim	9UBhK7j1	128.9M tok	4yJL4pEb
426,772,197	Back-Run 1	2mqrindM	9.595 SOL	nUmuvpxz
426,772,197	Back-Run 2	2mqrindM	9.218 SOL	5ekUS4oC
426,772,206	Back-Run 3	2mqrindM	1.269 SOL	2ruuYaMK
426,772,215	Back-Run 4	2mqrindM	1.066 SOL	59EBu3wd
426,772,226	Back-Run 5	2mqrindM	1.055 SOL	ssZ5cfZa
426,772,237	Back-Run 6	2mqrindM	0.826 SOL	5v2ACmwp
426,772,251	Back-Run 7	2mqrindM	0.529 SOL	2QTEp2DN
426,772,266	Back-Run 8	2mqrindM	0.490 SOL	DjNA6uJD
426,772,280	Back-Run 9	2mqrindM	0.534 SOL	5zDHGTEs
426,772,299	Back-Run 10	2mqrindM	0.492 SOL	4pbd4FbK
426,772,317	Back-Run 11	2mqrindM	2.409 SOL	2Xk8o74J

Table 18: Staged exit by 2mqrindM. The first two back legs land in the front's own slot and return 71.4% of the position. The remaining nine decay over 120 slots until 99.99% is sold, for 27.483 SOL out and 20.767 SOL net.

leg that takes far longer to complete, and selling the position slowly may equally serve a better price.

Figure 13 shows the daily share of Solana sandwiches that rely on evasive mechanisms. The share rises sharply around the closure of the Jito mempool, most likely to avoid detection. From mid-2025, when the validators stop being the main source of exposure, split legs become significant and reach around 6% of sandwiches by the end of our measurement period. Attackers likely use them to escape detection by the validators building the blocks. Note that we observe no such mechanisms at scale on the other chains.

G Attack Structure and Profit

In the following, we take a closer look at attack structure and profit. Recall that protected order flow sandwich legs land less precisely than public mempool sandwiches on Ethereum. Figure 3 shows that many transactions sit between the front-running leg, the victim, and the back-running leg. Only the swaps in the attacked pool matter, as everything else leaves the attack untouched. It is important to note that also swaps prior to the sandwich attack in the same pool matter, as the sandwich attack might encounter a different state than expected. As we cannot reliably determine what state the transaction expected, we focus on swaps in between the front and back leg in the same pool.

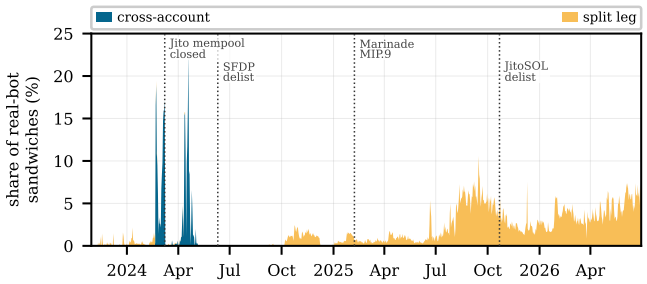


Figure 13: Daily share of sandwiches on Solana using an evasion mechanic: legs submitted by different accounts (cross-account) or a leg split over several transactions. Dotted rules mark interventions.

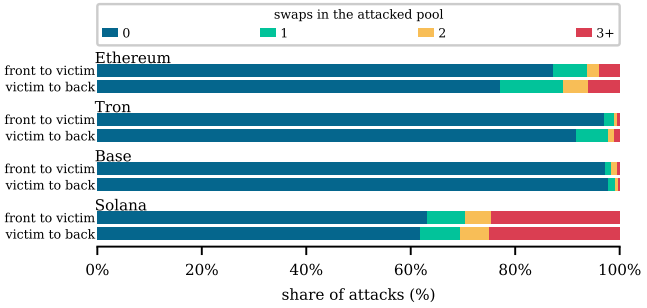


Figure 14: Swaps in the attacked pool between the front leg and the victim, and between the victim and the back leg, as a share of each chain's attacks. Counts of three or more are pooled into the last bucket.

Figure 14 reports the number of such swaps between the legs, counted on each side of the first victim separately. We note that we take the first swap in the attacked direction as the victim, where in principle it could be the second or the third. Any swap in between means the attack does not execute as planned, and the interference can leave it unprofitable.

Ethereum, Tron and Base see no swap in between for more than 70% of attacks, with Tron and Base above 85%, and more than three is rare. Solana differs. Only 55.4% of its attacks run free of interference, and 69.6% carry at most three swaps on either side. Attacks on protected order flow are thus least exact on Solana.

Interference, i.e., any trade in the attacked pool other than the first victim between the legs, further separates the failed attacks from the successful ones. On Ethereum, 96.2% of unprofitable attacks have a swap in the attacked pool between their legs, against 22.4% of profitable ones. Tron shows the same gap at 91.0% against 6.5%, and Base at 88.7% against 2.0%.

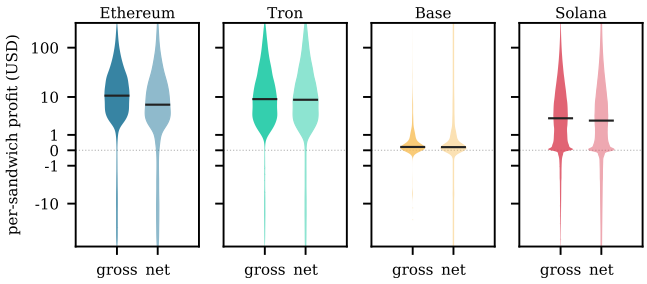


Figure 15: Per-sandwich profit distribution per chain, gross and net of gas.

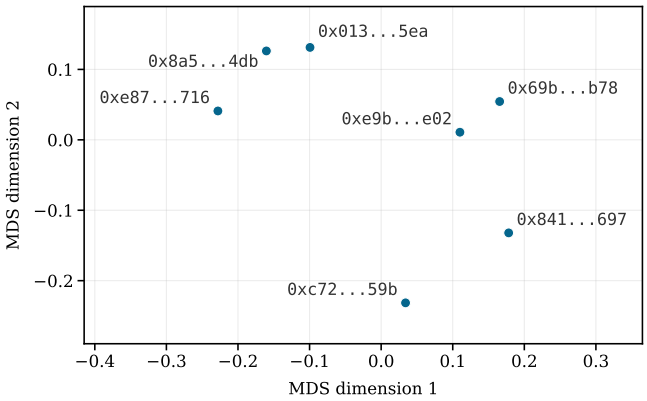


Figure 16: Identified protected order flow sandwich bot smart contracts on Ethereum, clustered based on bytecode and function selector similarity.

Figure 15 further shows the profit distribution per chain, gross and net of gas. Ethereum and Tron look alike, at a gross profit of around $10 per attack. Base sits far below at $0.16. Solana is smaller as well at $3.22, and its distribution is considerably wider.

H Entity Clustering

We cluster bots performing sandwich attacks across the three EVM chains, Ethereum, Base, and Tron, where we observe sandwich attacks against protected order flow.

H.1 Ethereum

For Ethereum, we cluster the destination addresses of the identified sandwich transactions based on the bytecode associated with each address. We employ a two stage clustering approach. In the first stage, we cluster the bytecode of each sandwich contract. We first remove the Solidity CBOR metadata and disassemble the remaining bytecode into EVM opcodes. We then separate executable code from embedded data by removing all `PUSH` instructions and their associated data bytes. The resulting opcode sequences are subsequently partitioned into 3 grams, and DBSCAN is applied using Jaccard distance, with a similarity threshold of 75%. In the second stage, we leverage EVMole [20] to extract the function selectors embedded in each sandwich bot's bytecode. Within each bytecode based cluster, we then measure the similarity between contracts based on the overlap of their function selector sets. Specifically, we employ a 75% set overlap similarity threshold and again apply DBSCAN to identify contracts implementing similar functionality.

The first, bytecode based stage yields two clusters: one containing three contracts and one containing four contracts. Applying the second stage merges the two clusters, resulting in a single final cluster, as shown in Figure 16. Based solely on the bytecode analysis, we can therefore infer that all seven identified bots likely belong to the same entity. To validate our clustering results, we examine the bots' deployment history. The contract `0xc72...59b` was deployed by `0xa46...04a`, which was itself funded by `0x928...2d9`. Notably, `0x928...2d9` also deployed contracts belonging to the other six sandwich bots, including `0x841...697`, for example. This funding and deployment relationship provides additional evidence that the seven bots are controlled by the same entity.

H.2 Base

For Base, we apply the same methodology as for Ethereum. However, two of the four identified sandwich bot smart contracts are proxies implementing the EIP-1967 Transparent Proxy pattern [82]. For `0xc9c...41b`, we identify 16 distinct implementations, while `0xfcf...a26` has 5 distinct implementations. We retrieve the bytecode of all identified implementations and initially cluster the implementations associated with each proxy. We then incorporate these clusters into our aforementioned two-stage clustering procedure.

Figure 17 shows that the two bots `0xc9c...41b` and `0xfcf...a26` have distinct bytecode but are are assigned to the same cluster via their function selectors, whereas the remaining two bots, `0x000...b3e` and `0x14d...b75`, are assigned to distinct clusters. We manually validate these results by examining the deployment history of the four bots. Our analysis shows that `0x000...b3e` and `0x14d...b75` were deployed by distinct accounts, while `0xc9c...41b` and `0xfcf...a26` were both deployed by `0xe59...300`. These findings confirm our clustering results and indicate that three distinct entities performed sandwich attacks against protected order flow on Base.

H.3 Tron

For Tron, our clustering is different as all but one bot do not use a custom contract implementation but instead interact with pools directly from a EOA. Our clustering identifies nine of the twelve to be controlled by the same entity as indicated in Table 7. For this we have various forms of evidence that

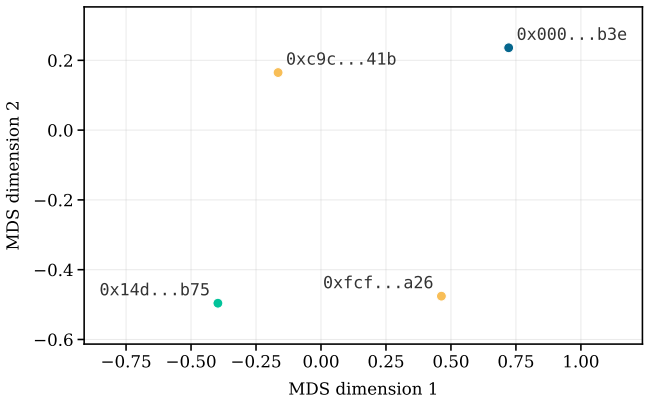


Figure 17: Identified protected order flow sandwich bot smart contracts on Base, clustered based on bytecode and function selector similarity.

respectively link a subset of them: *vanity addresses*, *funding*, and *pool partition*.

Vanity Addresses. The entity generated vanity addresses for four of the nine bots, all carrying the `0x1f0a` prefix. Three further wallets matching the prefix of the first by chance has probability 16^{-12}, i.e., roughly 1 in 3×10^{14}. Note that MEV bots sometimes create recognizable addresses on purpose.

Bot	First Funded	TRX	Funder
`0xc83...28b`	—	—	(deployed contract)
`0x35e...9a1`	24-09-08	2,499	`0x0b4...e24`
`0x728...6e9`	25-02-21	3,000	`0xaa9...b1a`
`0x1f0...600`	25-03-22	600	`0xaa9...b1a`
`0x8ad...267`	25-03-18	1,500	`0xaa9...b1a`
`0x1f0...f9e`	25-03-23	1,600	`0xaa9...b1a`
`0x1f0...bd4`	25-03-19	1,300	`0xaa9...b1a`
`0x1f0...e9f`	25-03-26	600	`0xaa9...b1a`
`0x34f...978`	25-03-20	1,000	`0xaa9...b1a`
`0xfe9...ef4`	24-12-04	1,000	`0xeb4...864`
`0x9f0...70d`	25-03-23	1,000	`0xaa9...b1a`
`0xb2c...713`	25-02-16	800	`0xaa9...b1a`

Table 19: First inbound native TRX transfer to Tron persistent bots. `0xaa9...b1a` is the common funder of nine bots.

Funding. The nine addresses we attribute to a single entity, shaded in Table 7, received their first inbound native transfer from `0xaa9...b1a` within 38 days of each other. These nine cover all four `0x1f0a` wallets and five wallets without that prefix. The four `0x1f0a` wallets were funded within eight days, but two wallets without the prefix were funded in between them. We note that the funding wallet paid out to 15 addresses in total, nine of them these entities. It therefore does not act as a general funding source such as an exchange, which makes the shared origin meaningful.

Pool Partition. The nine bots partition the pools they attack, sharing no pool across any of their 36 pairs (cf. Figure 18). This partition suggests coordination between the entities through a joint operator.

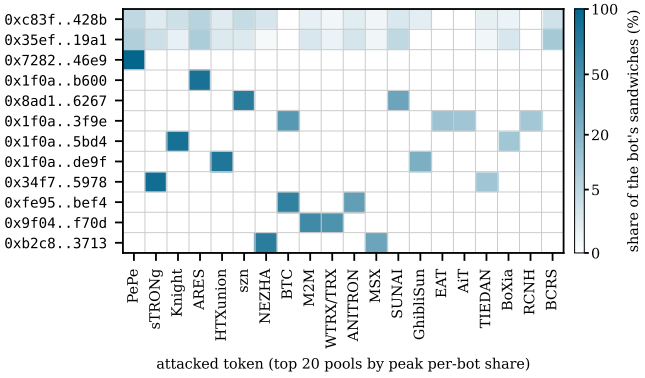


Figure 18: Tron sandwich attacks by bot and pool.

I Origin Details

This appendix contains the per-chain figures supporting Section 6.

I.1 Ethereum

I.1.1 Ruling Out Block Builders

Builders receive private transactions in plaintext, making the block-building stage a natural candidate for the exposure behind protected sandwiches on Ethereum. We test this by comparing each builder's sandwich share with its block share.

Builder	Blocks (%)	25/11	25/12	26/01	26/02	26/03	26/04
Titan	47.7	1.14	1.17	1.20	1.14	1.13	1.24
BuilderNet (Flashbots)	13.6	1.13	1.17	1.13	1.14	1.10	1.35
Quasar	15.0	1.05	0.96	0.87	0.88	0.99	0.67
BuilderNet (Nethermind)	5.7	1.04	1.16	1.23	1.26	1.28	1.11
BuilderNet (Beaver)	5.2	1.10	1.19	1.31	1.21	1.18	0.50
beaverbuild	1.4	1.29	0.67	1.29	1.35	1.37	0.42
Sandwiches		**1,103**	**959**	**1,338**	**2,187**	**2,581**	**6,139**

Table 20: Ethereum builder sandwich share divided by block share, per month. A ratio of one indicates no over- or under-representation.

Table 20 reports, for each builder, the ratio between its share of protected sandwiches and its share of blocks, restricted to the months with a substantial number of attacks. A ratio of one means the builder carries exactly as many sandwiches as its block share implies. No builder exhibits the clear over-representation that would be expected from a builder-specific exposure, in contrast to the substantial excess ratios observed for Solana leaders (cf. Section 6.4.1). Titan sits between 1.13 and 1.24 throughout while building 47.7% of blocks, and the remaining builders move around one in both directions.

I.1.2 OFA Attribution

Table 21 reports the full distribution of observed OFA label combinations among sandwich victims, together with their single-victim share. The largest individual category is Blink-only, accounting for 20.9% of all victims, followed by the MEV-Share + MEVBlocker-only combination at 13.4%. Overall, 33.2% of all victims appear in multiple OFA datasets, corresponding to 53.0% of attributed victims.

OFA Combination	Victims	% of Total	% of Attributed	Single-Victim
Unattributed	14,702	37.26%	–	43.53%
Blink	8,259	20.93%	33.36%	89.85%
Share + Blocker	5,280	13.38%	21.33%	88.58%
Share + Blink	3,542	8.98%	14.31%	88.14%
Share	1,727	4.38%	6.98%	29.53%
Blocker + Blink	1,712	4.34%	6.91%	81.43%
Blocker	1,553	3.94%	6.27%	67.42%
Share + Blocker + Blink	1,430	3.62%	5.78%	92.87%
Share + Merkle	611	1.55%	2.47%	90.83%
Share + Blocker + Blink + Merkle	232	0.59%	0.94%	94.83%
Share + Blink + Merkle	205	0.52%	0.83%	96.59%
Merkle	104	0.26%	0.42%	56.73%
Share + Blocker + Merkle	71	0.18%	0.29%	92.96%
Blink + Merkle	21	0.05%	0.08%	90.48%
Blocker + Blink + Merkle	11	0.03%	0.04%	54.55%
Blocker + Merkle	1	0.00%	0.00%	100.00%
Total	**39,461**	**100.00%**	–	

Table 21: Observed OFA label combinations among victims of successful and unsuccessful private sandwich attempts. *Share* and *Blocker* denote MEV-Share and MEVBlocker, respectively. Each row reports the exact combination of OFA labels observed for a victim (e.g., *Blink* denotes Blink only), together with the share for which it was the sole victim of the sandwich. *Unattributed* denotes victims matched by none of the four datasets and is excluded from the *% of Attributed* column, which totals 24,759 victims.

Figure 19 shows the weekly evolution of victims observed only in MEV-Share or only in MEVBlocker. The temporal concentration of these single-OFA exposure categories is discussed in Section 6.1.3.

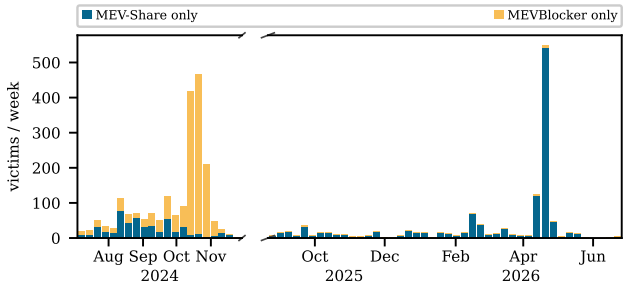


Figure 19: Weekly evolution of sandwich victims observed exclusively in MEV-Share or MEVBlocker.

I.1.3 Sandwiching Under the Risk of Back-Runs

A back-run and a sandwich feed on the same price displacement, so a back-run executed before the sandwicher's unwind takes away profit from that unwind. We quantify the tension from both sides which is relevant for the sandwiches we observe on OFA users (cf. Section 6.1.2). A two-pool benchmark derives when a sandwich survives such a back-run, and a counterfactual over the observed victims measures how often a profitable direct back-run exists at all.

When Not to Sandwich? In a within-block wide or a cross-block sandwich, an intervening back-runner may arbitrage away part of the victim and front-run-induced price distortion before the sandwicher unwinds, thereby reducing or even eliminating the sandwich's profit. Thus, a non-tight sandwich should decide under what conditions it's likely to succeed and, based on that, how to optimize it. We demonstrate this case in the simplest settings, where the sandwicher can decide based on the victim transaction, which can be back-run as well as sandwiched, the optimal front-run amount that would succeed under the optimal intervening back-runner. For simplicity, we assume that the victim, back-runner, and sandwicher are different identities that decide independently.

We consider a continuous, fee-only two-pool V2 benchmark. A victim swaps X for Y through pool 1, and the only alternative source-pool-disjoint route between the same tokens is pool 2. Let (x_i, y_i) denote the X and Y reserves of pool i, let f_i be its fee rate, and define $\gamma_i = 1 - f_i$. Before the front-run, we assume that the pools satisfy the gas-free marginal no-arbitrage condition

$$\gamma_1\gamma_2 \le \frac{x_1 y_2}{x_2 y_1} \le (\gamma_1\gamma_2)^{-1}.$$

For an $X \to Y$ swap with input δ, define

$$\Phi^{X\to Y}_{\delta,\gamma}(x,y) = \left(x+\delta, \frac{xy}{x+\gamma\delta}\right),$$

with the reverse transition defined analogously.

We consider a sandwicher $\mathcal{S}$, a victim $\mathcal{V}$, and a back-runner $\mathcal{B}$, and condition on the realized same-block ordering

$$\mathcal{S}_{\text{front}} \prec \mathcal{V} \prec \mathcal{B} \prec \mathcal{S}_{\text{unwind}}.$$

The back-runner reads the live post-victim reserves at execution time and computes its trade on-chain [92, 102, 105]. For the theoretical optimum, we treat token amounts as continuous and ignore the back-runner's gas costs and capital constraints.

Let $\delta_{\mathcal{S}}$ and $\delta_{\mathcal{V}}$ denote the X inputs of the front-run and victim transactions through pool 1. After these transactions, its state is

$$\Phi^{X\to Y}_{\delta_{\mathcal{V}},\gamma_1}\left(\Phi^{X\to Y}_{\delta_{\mathcal{S}},\gamma_1}(x_1,y_1)\right) = (\tilde{x}_1, \tilde{y}_1).$$

The back-runner cycles $X \to Y$ through pool 2 and then $Y \to X$ through pool 1. Let $\delta_{\mathcal{B}}$ denote the back-runner's input to pool 2. This trade produces

$$q_{\mathcal{B}}(\delta_{\mathcal{B}}) = \frac{\gamma_2 y_2 \delta_{\mathcal{B}}}{x_2 + \gamma_2 \delta_{\mathcal{B}}}$$

units of Y. Define

$$C = \gamma_1\gamma_2\widetilde{x}_1 y_2, \qquad D = x_2\widetilde{y}_1, \qquad E = \gamma_2(\widetilde{y}_1 + \gamma_1 y_2).$$

The back-runner's gross X-denominated profit is

$$\Pi_{\mathcal{B}}(\delta_{\mathcal{B}}) = \frac{\gamma_1\widetilde{x}_1 q_{\mathcal{B}}(\delta_{\mathcal{B}})}{\widetilde{y}_1 + \gamma_1 q_{\mathcal{B}}(\delta_{\mathcal{B}})} - \delta_{\mathcal{B}} = \frac{C\delta_{\mathcal{B}}}{D + E\delta_{\mathcal{B}}} - \delta_{\mathcal{B}}.$$

Writing $[z]_+ = \max(z, 0)$, its maximizing input is

$$\delta^*_{\mathcal{B}} = \frac{[\sqrt{CD} - D]_+}{E}, \qquad q^*_{\mathcal{B}} = \frac{\gamma_2 y_2 \delta^*_{\mathcal{B}}}{x_2 + \gamma_2\delta^*_{\mathcal{B}}}.$$

A positive-size gross-profitable back-run exists exactly when $C > D$. After this back-run, the pool states are

$$(\bar{x}_1, \bar{y}_1) = \left(\frac{\widetilde{x}_1\widetilde{y}_1}{\widetilde{y}_1 + \gamma_1 q^*_{\mathcal{B}}}, \widetilde{y}_1 + q^*_{\mathcal{B}}\right)$$

and

$$(\bar{x}_2, \bar{y}_2) = \left(x_2 + \delta^*_{\mathcal{B}}, \frac{x_2 y_2}{x_2 + \gamma_2\delta^*_{\mathcal{B}}}\right).$$

The front-run leaves $\mathcal{S}$ holding $\delta'_{\mathcal{S}} = \gamma_1\delta_{\mathcal{S}} y_1/(x_1 + \gamma_1\delta_{\mathcal{S}})$ units of Y. We consider a standard back-run-unaware sandwicher that unloads all of this inventory through pool 1 after $\mathcal{B}$. It receives $\delta''_{\mathcal{S}} = \gamma_1\delta'_{\mathcal{S}}\bar{x}_1/(\bar{y}_1 + \gamma_1\delta'_{\mathcal{S}})$ units of X, giving gross profit

$$P_{\mathcal{S}} = \delta''_{\mathcal{S}} - \delta_{\mathcal{S}}.$$

For every fixed $\delta_{\mathcal{S}} > 0$,

$$P_{\mathcal{S}} > 0 \iff \gamma_1^2 y_1(\bar{x}_1 - \delta_{\mathcal{S}}) > \bar{y}_1(x_1 + \gamma_1\delta_{\mathcal{S}}).$$

Here, $(\bar{x}_1, \bar{y}_1)$ depends on $\delta_{\mathcal{S}}$ through the victim state and $\mathcal{B}$'s best response. For a fixed front-run, any executed positive-size back-run reduces $\bar{x}_1$, increases $\bar{y}_1$, and therefore strictly lowers the proceeds of this fixed source-pool unwind. It may, but need not, eliminate the sandwich's gross profit.

Let $\mathcal{D}_{\mathcal{S}}$ denote the feasible front-run sizes imposed by the victim's slippage constraint and the sandwicher's available capital, and let $K_{\mathcal{S}}(\delta_{\mathcal{S}})$ denote the sandwicher's X-denominated execution and financing costs. Under this benchmark, the sandwicher should abstain when

$$\sup_{\delta_{\mathcal{S}} \in \mathcal{D}_{\mathcal{S}}} \{P_{\mathcal{S}}(\delta_{\mathcal{S}}) - K_{\mathcal{S}}(\delta_{\mathcal{S}})\} \leq 0.$$

We leave this one-dimensional optimization and alternative back-run-aware unwind strategies outside our analysis.

Back-Run Analysis. The benchmark asks what a back-run costs the sandwicher. Our empirical counterfactual asks the prior question, whether the victim alone creates a profitable direct back-run at all, and, whenever the pre-victim state is conclusive, whether the victim created or enlarged the opportunity. It does not estimate the sandwicher's residual profit.

We restrict this analysis to the 31,259 victim transactions whose observed front-run occurs in the same block as the victim. For a victim in block b_e, the state at the end of block $b_e - 1$ excludes both transactions and therefore provides a baseline uncontaminated by the observed front-run. We exclude the remaining 8,202 victims because their earlier-block front-run is already reflected in this state. We do not require the sandwicher's observed unwind to share the victim block because it is not executed in our victim-only counterfactual.

Activity Convention. We determine pool activity from on-chain state at the end of block $b_e - 1$: positive reserves for V2, positive current-range liquidity for V3, positive `StateView` liquidity for V4, and corresponding protocol-specific checks for other venues. Unavailable state remains unknown rather than inactive. We include the source pool by construction. In four V3 cases, it is counted despite reporting zero current-range liquidity because the observed same-block front-run successfully accesses initialized liquidity. We exclude these four baseline states from numerical modeling.

Counterfactual Execution. For each event e, we identify the sandwiched source pool, the victim's swap direction $A_e \to B_e$, and every active alternative pool containing the same token pair. From the `Swap` log in the victim's receipt, we decode the exact amount of A_e entering the source pool. Starting from the source pool's state at the end of block $b_e - 1$, we apply this input only to the source pool and leave the alternatives unchanged. We recompute the victim's output because its observed output was produced after the front-run and therefore reflects a different state.[5]

For every active alternative pool p, we attempt to simulate a direct cycle that trades A_e for B_e through p and then trades the resulting B_e back to A_e through the source pool. For V2/V2 routes, we compute the exact integer profit-maximizing input separately in the pre- and post-victim states. For routes containing V3 or V4, we search candidate inputs using exact integer swaps with initialized-tick traversal and retain the most profitable input found. A route is *victim-created* when the pre-victim state is certified to admit no gross-profitable exact-input trade and the post-victim search finds a profitable input. A route that was already profitable before the victim is *victim-expanded* when the victim strictly increases its profit. For V2/V2 routes we compare the exact pre- and post-victim optima, and for concentrated-liquidity routes we replay the profitable pre-victim input after the victim. When the pre-victim search is inconclusive, we retain any post-victim opportunity we find but leave its victim-effect classification open. When the post-victim search is inconclusive, the route stays

[5] This is an observed-input source-leg replay rather than a full transaction replay: without the front-run, an exact-output or multihop transaction could produce a different source-leg input. Apart from the observed source-pool swap, we condition only on confirmed chain state through block $b_e - 1$, thereby omitting transactions earlier in block b_e and information that a searcher might obtain from OFA order flow or other pending transactions. These restrictions isolate the victim's marginal effect under a reproducible counterfactual.

inconclusive rather than being recorded as unprofitable.

We evaluate only this victim-induced direction. An $A_e \to B_e$ victim makes B_e relatively scarcer in the source, potentially making it profitable to buy B_e from an alternative and sell it into the source. We do not analyze arbitrage in the reverse direction because its execution before the source-pool unwind would move the source further in the victim's direction and improve, rather than erode, the proceeds of that unwind.

Execution and Validation. V2 routes use exact integer arithmetic with historically verified fees. We exclude 1,138 candidate routes across 804 victims whose fee semantics cannot be verified. We model pools created by the official Uniswap V3 factory and official hookless, static-fee Uniswap V4 pools using their historical price, liquidity, and tick state. We exclude other V3-style pools and hooked or dynamic-fee V4 pools because their exact execution semantics require additional implementation-specific modeling. Native ETH is treated as WETH for token matching and valuation.

The victim receipt must contain exactly one source-pool swap in the recorded direction. We reject a route if source decoding is ambiguous or its historical state or fee cannot be established. We also reject it if the victim changes the candidate alternative, since applying the victim only to the source would then combine incompatible pool states. One exact positive post-victim execution is sufficient to establish an opportunity. We conclude that no profitable direct route exists only when every active direct alternative has a conclusive post-victim result and none is profitable. Unsupported or post-victim-inconclusive routes prevent a negative conclusion. Pre-victim inconclusiveness suppresses only the victim-created or victim-expanded label. It does not invalidate an exact positive post-victim result. This conclusion concerns only the modeled direct A_e–B_e alternatives and does not imply that no profitable multihop route exists.

Gas and Valuation. Gross profit is initially denominated in the victim's input token. When that token is WETH, conversion to ETH is exact. When the victim's output token is WETH, we approximate the input token's ETH value using the source pool's pre-victim reserve ratio. We do not assign a gas-adjusted conclusion to non-WETH pairs. We assume 150,000 gas for V2/V2 routes, 200,000 for V2/V3 routes, and 250,000 for V3/V3 routes or any route containing V4. Let $G_{e,p}$ denote this route-specific gas usage, g_e the victim transaction's effective gas price in wei per gas, and $\widehat{\Pi}^{\text{WETH}}_{e,p}$ the route's gross profit expressed in wei. We price the hypothetical back-run at one wei per gas above the victim's realized effective gas price:

$$\widehat{\Pi}^{\text{net}}_{e,p} = \widehat{\Pi}^{\text{WETH}}_{e,p} - G_{e,p}(g_e + 1).$$

This is a profitability screen rather than an ordering model or an exact net-profit estimate: actual gas usage may differ, and the calculation omits builder payments, financing costs (e.g., flashloans), and failed attempts.

Measure	Victim Transactions
Same-block cohort	31,259
At least one active direct alternative	9,886
At least one conclusive post-victim route	7,977
All active alternatives conclusive post-victim	6,374
Post-victim gross-positive route	3,905
At least one victim-created route	2,272
At least one victim-expanded route	1,859
Gross-positive and ETH-evaluable	3,863
Profitable after the gas screen	484

Table 22: Same-block victim-only direct-back-run results.

Results. Among the 9,886 victims with an active direct alternative, at least one post-victim route has a conclusive result in 7,977 cases (80.7%), while every active direct alternative has a conclusive post-victim result in 6,374. The event-level result remains inconclusive for 2,504 victims, primarily because the victim changes an alternative, the V2 fee cannot be verified, execution requires an unsupported protocol, noncanonical V3 implementation, or hooked or dynamic-fee V4 pool, or the post-victim search is inconclusive. The latter includes 56 candidate routes for which the post-victim marginal return is positive but no profitable exact integer input is found.

At zero gas cost, we find a gross-profitable route for at least 3,905 victims (12.5% of the cohort). Of these, 2,272 have at least one victim-created route and 1,859 have at least one victim-expanded route. Overall, 3,895 victims have at least one victim-created or victim-expanded route. 236 exhibit both classifications on different alternatives. Of the remaining ten gross-positive victims, four have an inconclusive pre-victim search and therefore receive no victim-effect classification, while six have only pre-existing routes for which we find no verified expansion.

Profit is ETH-valued for 3,863 positive cases and is strongly right-skewed: the median, 90th percentile, and 99th percentile of the best detected gross profit are 2.63×10^{-5}, 1.37×10^{-3}, and 1.52×10^{-2} ETH. The gas screen removes 3,379 of these 3,863 opportunities (87.5%), leaving 484 profitable cases, or 1.5% of the cohort. Thus, based on our methodology, detected direct opportunities occur for 12.5% of the cohort before gas and 1.5% under our gas screen. For routes containing V3 or V4, every reported positive result is supported by exact integer swap execution, but the search may miss another profitable input or understate its maximum profit.

Future work should derive back-run-aware sandwich strategies and extend the empirical analysis to multihop routes.

I.2 Tron

Tron implements first-come-first-served transaction ordering in the client, so a block producer running a modified ordering policy is a natural candidate for the observed sandwiches on protected order flow. We test this by comparing each producer's share of sandwiches with its share of blocks.

Figure 20 shows each producer's share of sandwiches rel-

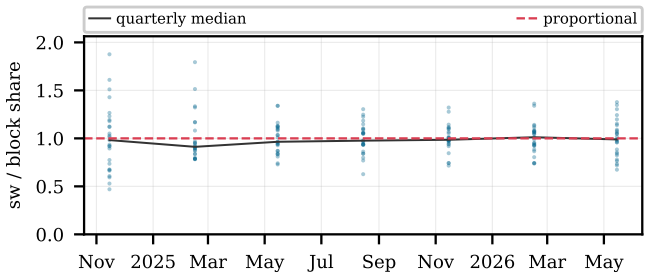


Figure 20: Sandwich production per Tron block producer, relative to the blocks it produces. Each UTC calendar quarter, a producer's share of that quarter's sandwiches divided by its share of that quarter's blocks. A ratio of 1.0 indicates sandwiching in proportion to block production. Quarters with fewer than 300 sandwiches are omitted, as are producers making up less than 2% of a quarter's blocks.

ative to its share of blocks. The ratio ranges from 0.47 to 1.88, with 90% of leader-quarters between 0.72 and 1.34, and no producer sustains an elevated share across quarters. Attacks thus land almost uniformly across producers, making producer-specific misordering an unlikely explanation for the observed sandwiches.

I.3 Base

In the following, we detail the behavior of the four sandwich attack bots on Base, which form three entities.

`0x000...b3e`. The evidence for this bot is most consistent with pre-inclusion visibility, either through provider-local transaction-pool exposure or an equivalent private pending-order-flow source. Its activity is confined to 13 to 28 August 2023, which overlaps Base's disclosures that some shared RPC offerings exposed transactions from their local pending pools, along with the corresponding Base and op-geth mitigations [6–8, 76] (cf. Figure 7). Our detector output holds 202 candidates, 200 of which have a positive gross return. Among the 199 same-block candidates, the back-run reproduces the victim transaction's complete fee values in every case, i.e., the transaction type, the effective gas price, `maxFeePerGas`, and `maxPriorityFeePerGas`. These copies span 128 distinct fee tuples, which rules out a single fixed fee configuration.

`0x14d...b75`. Our detector flags 239 sandwich sequences for this bot, 234 of which follow one pattern. The victims and the tokens they trade share their funding. In 232 of the 234 cases the target token's deployer was funded directly by `0xc43...078`, and 233 of the 243 victim transactions originate from three EOAs funded by `0xb01...42b`. Both hubs trace upstream to one funding root, `0xea7...259`. Every occurrence follows the same loop, i.e., `0xc43...078` funds a token deployer, the deployer creates the token and seeds the pool, `0x14d...b75` front-runs, the buyer EOAs execute their buys, and `0x14d...b75` follows with the back-run. Table 9 shows one cycle of the loop. The token creators and the victims therefore operate as one infrastructure. We observe no on-chain funding link between `0x14d...b75`, or its deployer and beneficiary `0xaea...8fd`, and the victim-side funding lineage. We therefore do not attribute it to the same operator and classify it as an opportunistic actor exploiting an automated campaign across multiple blocks rather than a conventional intra-block sandwich bot.

`0xc9c...41b` and `0xfcf...a26`. These two contracts are successive deployments of one managed execution system. Both are ERC-1967 proxies controlled by `0xe59...300`, they share upgrade infrastructure, and they reuse execution EOAs. All 21 EOAs sending `0xfcf...a26` front-run legs had previously executed through `0xc9c...41b`. Their operating periods do not overlap, which points to an operational migration rather than two independent bots (cf. Section H.2).

The detector identifies 1,448 candidates, 750 for `0xc9c...41b` and 698 for `0xfcf...a26`, of which 1,402 have a positive gross profit before gas and 1,445 sit within one block. 18 `0xc9c...41b` candidates contain two victim transactions , which yields 1,463 transaction-level fee comparisons. In 1,274 of them (87.1%) the effective-gas-price deltas are $(g_F - g_V, g_B - g_V) = (+1, 0)$ wei. The relation holds for all 698 `0xfcf...a26` comparisons and for 576 of the 765 `0xc9c...41b` comparisons.

The victim flow is highly structured. We tie 1,109 of the 1,466 distinct victim transactions (75.6%) to seven funding-linked cohorts spanning 64 wallets, linked through direct funding, a common dispersal transaction, or multi-hop ancestry. The four largest cohorts contribute 481 transactions from ten wallets tracing to `0xea7...259`, 284 from eight DUAL wallets funded together with two dispersal transactions (one funds them with ETH for gas `0x76a...929`, the other with DUAL token `0xf73...ac7`), 129 from four wallets funded directly by `0xe6e...201`, and 121 from fifteen wallets tracing to `0x24f...999`. Three smaller cohorts add 61, 20, and 13 transactions, the last being 13 AMETA buyers whose five immediate funding branches converge at `0xfbf...fec`. The two largest cohorts carry 69.0% of the funding-linked flow and the four largest 91.5%. Funding-linked flow is also more common under `0xfcf...a26` (632/698, 90.5%) than under `0xc9c...41b` (477/768, 62.1%).

Funding ancestry alone does not demonstrate common control. Its significance comes from the execution regularities that accompany it. All 284 sandwiched DUAL transactions trade the same WETH/DUAL pool, arrive on a five-to-six-block cadence, set zero minimum output, and use the same effective gas price in 280/284 cases. The four wallets funded by `0xe6e...201` execute their 129 sandwiched swaps through the Aerodrome Router using the same function, one-hop USDC/LIQ route, pool, self-recipient form, zero minimum output, and fee tier, on a 92–95-block cadence. The `0x24f...999`, `0xeda...4ba`, and `0xb73...b96` cohorts use the public Uniswap V2 Router, so router reuse alone tells us

little there. What matters is the conjunction of cohort-level funding links with regular execution, and every transaction within each cohort uses the same gas price. These regularities are consistent with coordinated or automated systems.

The 0xea7...259 cohort also connects this activity to the launch factory previously targeted by 0x14d...b75. Its ten buyer wallets make 481 purchases across 71 newly created tokens. On a separate branch, 0xea7...259 funds 0xc43...078, which supplies 13.02 ETH to each token creator. Each creator deploys a token at nonce zero, provides liquidity, later removes it, and returns the ETH to 0xc43...078. The median intervals from funding to deployment, deployment to liquidity, and liquidity to first purchase are 8, 7, and 61 blocks. Roughly fourteen months earlier, 0x14d...b75 targeted the same funding and launch lifecycle in 232 of its 234 core cases. The two campaigns share no token addresses, and 0x14d...b75 has a separate observed funding lineage, so the connection is the reoccurrence of a predictable token launch cycle rather than a shared operator.

These findings suggest prediction and pre-positioning are plausible for structured cohorts, but not always. Of 0xfcf...a26's 698 victim transactions, 667 use one of three dominant effective-gas-price settings: 6.0, 6.3, or 51 million wei. The remaining 31 transactions come from 22 senders and use 24 distinct effective gas prices, nine call targets and selectors, and 14 pools. Thirty lie outside the seven funding-linked cohorts, while one belongs to the 0xea7...259 cohort. Nevertheless, everyone is bracketed with the exact $(+1,0)$ relation (cf. Figure 21). Alignment, this specific to the transaction, is difficult to derive from a static schedule and remains consistent with either pre-sequencing transaction visibility or a shared off-chain backend. The on-chain evidence neither distinguishes between these mechanisms nor establishes common control between the funding cohorts and the executor.

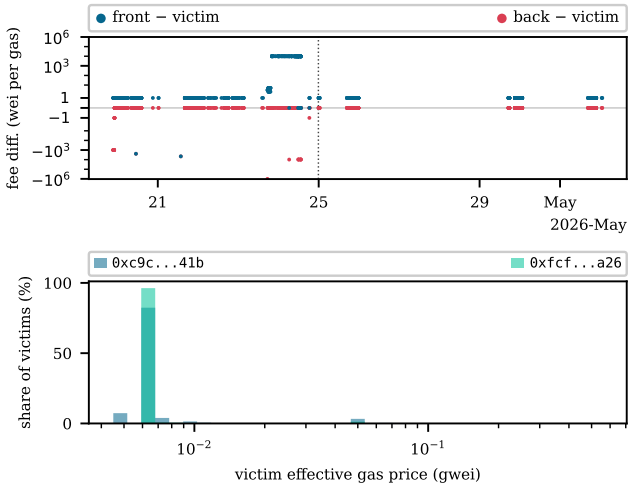


Figure 21: Distribution of differences of fees set by attacker and victim as well as full fee distribution of the victim transactions for bots 0xc9c...41b and 0xfcf...a26.

Multi-victim cases. Multi-victim cases are rare in our Base dataset, with most sandwiches involving a single candidate victim. In the 23 multi-victim cases, both ordinary purchases and atomic arbitrage transactions can be labeled as victims. These cases require a more nuanced definition of a *victim*: an atomic arbitrage can remain profitable and still be harmed if the sandwich front-run reduces its profit. Using the arbitrage detection methodology of Solmaz et al. [92], we identify ten atomic arbitrage transactions among the victims in these cases. Such instances may be more common on high-throughput, low-cost L2s, such as Base, where inexpensive execution supports optimistic on-chain searching [92, 105].

I.4 Solana

I.4.1 Leader Concentration

In the following, we detail how sandwich attacks distribute across Solana's leaders (cf. Section 6.4.1).

Figure 22 tracks each leader's share of sandwiches relative to its share of blocks on a weekly basis, with the ten most over- and the ten most under-represented leaders shown daily. The two groups start far apart, and both move towards one over time. The suppressing leaders do so from the first quarter of 2025 and the over-represented ones follow from the first quarter of 2026. We note that these curves cover the extremes of the distribution alone.

Figure 23 plots the Lorenz curves per era, i.e., the cumulative share of sandwiches against the cumulative share of blocks produced, with leaders ordered from most to least attacked. Distance from the diagonal measures how much of the attack volume sits with few leaders. Under the Jito mempool the curve rises steeply, i.e., the top ten leaders alone hold 26.4% of the sandwiches on 15.7% of the blocks, and each era's curve sits closer to the diagonal than the last. By the Marinade MIP.9 era, the curve lies close to the diagonal, indicating that sandwiches are distributed across leaders approximately in proportion to the blocks they produce.

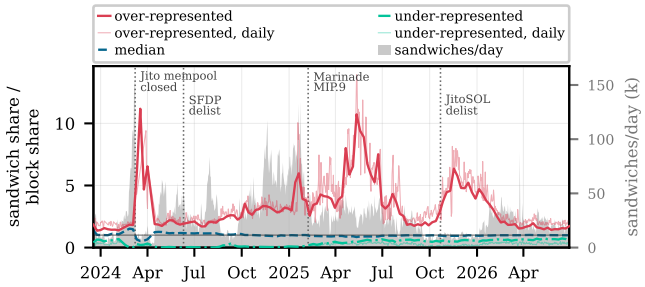


Figure 22: Sandwich production per Solana block producer, relative to the blocks it produces. Each week, a leader's share of that week's sandwiches divided by its share of that week's blocks, with the ten most over- and under-represented leaders also shown daily. Dotted rules mark interventions against sandwiching, shading daily attack volume.

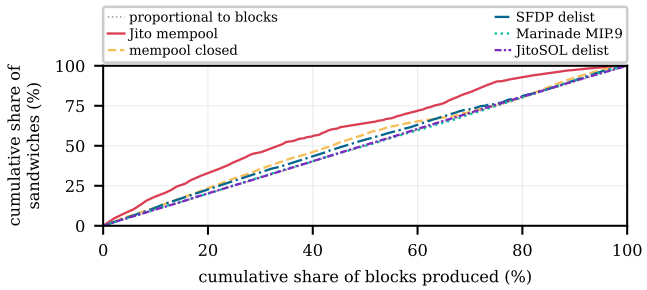


Figure 23: Cumulative share of Solana sandwiches against cumulative share of blocks produced, per intervention era. The diagonal is production in proportion to block share.

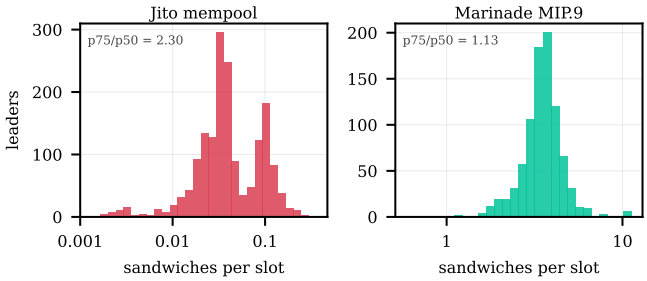


Figure 24: Per-leader sandwich rate, leaders with at least 5,000 blocks in the era. Jito's mempool was opt-in per validator, and the rate is correspondingly bimodal during the Jito mempool era. In later eras, the distribution becomes unimodal as the distinction between leader groups disappears.

Figure 24 shows the distribution of the per-leader sandwich rate in each era, restricted to leaders with at least 5,000 blocks. Under the Jito mempool the distribution carries two peaks, i.e., leaders fall into two behaviors, which follows from the mempool being opt-in per validator. The ratio of the 75th to the 50th percentile lies between 2 and 3 during the Jito mempool era and falls to 1.13 afterwards, indicating that per-leader sandwich rates become substantially more homogeneous.

I.4.2 Application Concentration

We further investigate the concentration of sandwich attacks across applications on Solana (cf. Section 6.4.2).

Figure 25 pairs each application's adoption of `jitodontfront` with the volume of its sandwiched transactions since the flag launched on 17 April 2025. Adoption is close to binary per application, since the application sets the flag rather than the user. Axiom sets it on 98.5% of its sandwiched victims and Padre on 99.4%, whereas Bloom, TradeWiz and Phantom never set it. Axiom dominates by volume, and three of the four most sandwiched applications, Axiom, GMGN, and Padre, set the flag on at least 90% of their victims. We recall that the flag only forces a flagged transaction to sit first in any Jito bundle, i.e., it rules out front-running inside a bundle and leaves every other path to the leader untouched. Setting the flag therefore does not by itself guarantee protection outside the Jito bundle path.

Table 23 repeats the excess-ratio analysis of Figure 9, restricting the numerator to single-victim sandwiches, for which there is no ambiguity over which enclosed transaction was targeted. The excess ratio decreases for every over-represented application, by roughly half for Axiom and Padre, but remains well above one. Jupiter Ultra rises to 2.0 in this subset, while unattributed flow remains below one in both analyses.

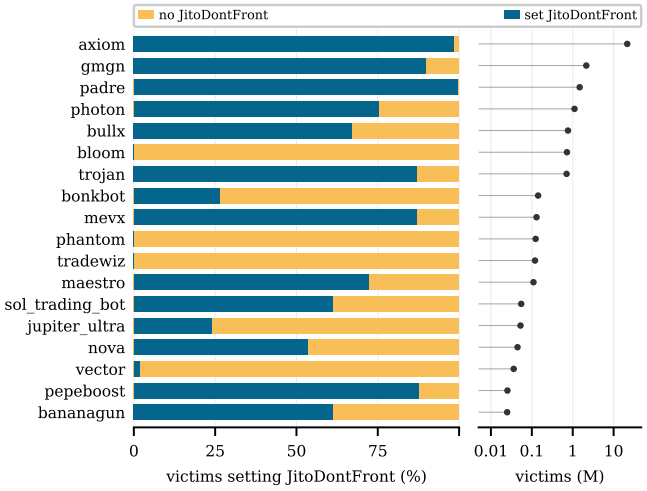


Figure 25: Application-level concentration of protected order flow sandwich victims on Solana, since `jitodontfront` launched (2025–04–17). For each application, we report the share of sandwiched victims carrying the flag (left) and the total number of sandwiched transactions (right). Three of the four applications with the largest victim volumes set the flag on at least 90% of their victims.

App	Quarters	Sole SWs	Excess	Excess (sole)
Axiom	6	619,987	18.9	8.1
Photon	10	1,375,685	11.1	7.2
BullX	9	472,404	10.7	5.4
GMGN	9	395,763	10.6	8.5
Padre	5	40,149	8.8	3.3
Trojan	10	728,267	5.6	5.0
BonkBot	11	827,141	5.4	3.9
Jupiter Ultra	12	1,628,708	0.7	2.0
Unattributed	12	10,498,623	0.5	0.7

Table 23: Excess ratio (victim share over swap share) per application on Solana, as the median over supported quarters, over all victims and over sole-victim sandwiches only (Sole SWs: number of such victims).