Discrimination of Toxic Flow in Uniswap V3: Part 4

CrocSwap

Follow

--

1

Listen

Share

This post is a new installment in an

[ongoing series

](/)by

[@0xfbifemboy

](https://twitter.com/0xfbifemboy)on Uniswap liquidity pools, concentrated liquidity, and fee dynamics. It is the fourth of multiple posts in a subsequence which aims to focus on the characterization of toxic flow in ETH/USDC swap data and potential implementations of price discrimination or flow segmentation mechanisms.

Introduction

Thus far, we have covered a great deal of territory in our research on Uniswap V3 order flow:

- Part 1 distinguished between profitable and unprofitable wallets, finding that a small proportion of toxic wallets generated the majority of the liquidity pool's losses
- Part 2 refined a classification of nontoxic vs. toxic wallets and analyzed the time decay of toxic wallets' trading alpha
- Part 3 characterized the toxicity of order flow depending on the source of the swap (order router, custom contract, etc.)

In a certain sense, we have been progressively stepping to higher levels of abstraction as we progress. We first looked at individual swappers; then, we attempted to categorize those swappers into broad groups based on observed behavior; after that, rather than looking at individual swappers, we then examined the order flow coming through particular contracts. At each level of abstraction, there are valuable insights that are not visible at other levels; for example, when we examine the behavior of individual wallets, we find that toxicity is associated with swap frequency and notional size, as well as the regularity of swaps coming from a given wallet. Such observations are not redundant with observations about order flow from different sources; instead, they are additive, together forming a complex tapestry of market structure with fine details.

There are still many questions about wallet behavior or aggregator order flow that remain unresolved. However, before we address those in more detail, it is natural to ask if we may step back even further to a yet higher level of abstraction. Fascinatingly, it turns out that if we look at the block builders

which assemble the transactions in each block, the identity and historical data associated with each block builder reveal some further intricacies in the nature of Uniswap swap flow. While the overall picture remains quite complex, our findings illustrate some details of a presently underexplored component of the overall DeFi ecosystem.

Overview of block building

Prior to the https://ethereum.org/en/upgrades/merge/, in which the Ethereum network switched from Proof-of-Work to Proof-of-Stake, new blocks on Ethereum were assembled by miners, who both received and ordered transactions. Skilled "searchers" of miner extractable value (MEV; now known as maximal extractable value post-Merge) were able to pay miners an extra fee in return for assembling transactions in a certain order that allowed them to derive profits greater than the fee paid. Since the Merge, however, which obsoleted the role of miners in block production, the structure of this ecosystem has changed significantly.

Currently, rather than miners producing blocks of transactions based on their Proof-of-Work computations, validators (who stake 32 ETH) are randomly selected to be block proposers, who send out blocks to be validated by other validators in the network. Validators, much like miners, may not necessarily possess any special expertise in building profitable blocks and extracting value from the reordering of transactions; instead, this role is left to the block builder,

who assembles an entire block of ordered transactions, along with a validator payment, and submits this block to the block

proposer. (For simplicity, we are here eliding the role of the relayer, which relays blocks from block builders to the current proposer.)

We notice immediately that there is a marketplace for competition among block builders: whoever can extract the most MEV from transaction reordering at any given time will be able to supply the highest validator payment, leading to the validation and finalization of their proposed block. Consequently, we might expect that blocks built by notable or skilled builders contain a high amount of MEV extraction. In contrast, if a block is built by a relatively uncompetitive block builder, we would expect that the block has a smaller amount of MEV. Arbitrage opportunities which manifest as toxic flow for the Uniswap ETH/USDC pool are sensitive to transaction ordering and speed of inclusion, and hence we anticipate that there should be a natural association between the amount of toxic flow observed in any given block and the extent of overall MEV extraction, which itself should correlate with the identity of different block builders.

Our hypothesis is straightforward enough; however, how do we actually collect data on block builders? Many analyses of Ethereum network activity tend to be scant on implementation details; however, we feel that it is valuable to the reader, as well as to overall clarity, to supply a more explicit description of our data retrieval process.

First, to illustrate, let us consider an arbitrarily selected block, such as block16787456:

We immediately see that the “Fee Recipient” is an address labeled by Etherscan asbuilder0x69. There is also an “Extra Data” field which contains a hexadecimal value encoding the string “@builder0x69”. The name “builder0x69” naturally suggests that this is an identifier for a specific block builder.

The astute reader may wonder: Should the “Fee Recipient” really be the same as the block builder? In fact, is it not typically the validator who receives the fees paid by transactions in the block? The answer is given by the very last transaction in the block, which is a transfer of Ether from builder0x69 to the Lido Execution Layer Rewards Vault:

It turns out that when blocks are relayed by FlashbotsMEV-Boost, the data which is sent to the block proposer both includes (1) setting the fee recipient to be the block builder and (2) a final transaction which transfers a payment of Ether from the block builder to the validator:

This behavior conforms to a Flashbots proposal for a standard specification for the payment of block builders to proposers. However, we should be careful to note that not all blocks fit this specification! Even though MEV-Boost-relayed blocks make up the majority of post-Merge Ethereum blocks by far, a number of other relayers also exist, and do not necessarily follow the same protocol. To illustrate, mev-boost.org shows that there is a fair amount of diversity in the relayer ecosystem:

Indeed, taking a look at block5958608, which was relayed by the BloXroute Max Profit relay, we see that the transaction fees go directly to the validator in this case (i.e.,

the Lido vault contract):

Unfortunately, this makes the analysis of block builders somewhat problematic: we can see that there is no “clean” way to identify the builder of a given block from the data contained within that block alone. Generally, builders seem to “sign off” in the extra data field, and there is no reason why one builder might impersonate another with this field; however, it is ultimately an imperfect categorization of blocks according to their builders. Although mev-boost.org supplies an explicit mapping between accepted blocks and the public keys of their builders, the data are only available through a limited API, and appear relatively laborious to acquire for the full set of all post-Merge blocks.

Therefore, for simplicity, while understanding that this is but an imperfect description of each block builder, we will simply use the extra data field supplied by each block as an identifier for the block builder itself. Some block builders are individual actors; others yet receive and compile externally submitted bundles of transactions; while the overall picture is quite complex, we will nevertheless be able to discern some fascinating details here.

Initial exploration of block builder data

Our dataset of blocks and their extraData

fields spans from block 15537393, the first post-Merge Ethereum block on September 15, 2022, to block16780152, which was proposed on March 8th, 2023. Approximately 75% of all post-Merge blocks explicitly specify some sort of non-empty extraData

and, in general, the percentage of blocks which specify no extraData

whatsoever has declined to below 10% in recent weeks, consistent with the development of a mature block builder ecosystem:

For simplicity, let us continue our analysis with a relatively simple question: What is the average swap PnL across the blocks built by any given block builder? First, we limit ourselves to post-Merge swap data and to swaps with notional size greater than $10,000 USD. (The latter condition helps prevent excessive bias from small, random swaps that exert strong influence on the average PnL; recall that blocks may have very few ETH/USDC swaps in them.) Within each block, then, we compute the weighted mean of swap PnL according to Binance 5-minute markouts and weighted by the amount of ETH swapped.

Next, we summarize across the value supplied in each block's extraData

field, taking the weighted mean of each block's average swap PnL weighted by the total amount of ETH swapped in each block. Finally, we limit ourselves to extraData

identifiers which are present in at least 1,000 different blocks. While imperfect, this is a reasonable first-pass attempt at constructing a metric that answers our question of interest.

If we order block builders by their PnL, looking first at those which are the most unprofitable for the liquidity pool (and hence most profitable for informed traders), we immediately notice the presence of a block producer with a generic, but consistent, extraData

identifier of geth^go1.19.1...linux

(the default signature supplied by go-ethereum) and an average swap PnL of -6.17 basis points (i.e.,

quite unprofitable for the liquidity pool, but profitable for the block builder). What is the identity of this block builder? Choosing random blocks built by this builder until we arrive at an MEV-Boost relayed block, we find that the address of this builder is always 0x473780deaf4a2ac070bbba936b0cdefe7f267dfc

:

It is particularly interesting to note that in essentially every case that we manually verified, the blocks built by this address include, at the very top of the block, interactions with the contract 0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

. For example, at the top of block16479013, we see one swap of AAVE for UNI and one swap of LINK for UNI, suggesting that (1) the contract 0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

is engaging in some sort of statistical arbitrage trading strategy and (2) the block builder 0x473780deaf4a2ac070bbba936b0cdefe7f267dfc

is run by the same organization controlling the contract0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

, prioritizing extraction of its own MEV strategies, which are sufficiently profitable to allow this block builder to consistently supply the highest validator payment.

If we examine the distribution of average PnL per block across all

post-Merge blocks, we find, as one might expect, that the distribution is quite strongly centered at zero, meaning that most blocks are, on average, only slightly profitable or unprofitable for the liquidity pool:

Now, if we examine same distribution for blocks built by0x473780deaf4a2ac070bbba936b0cdefe7f267dfc

, we find that the distribution is just slightly

skewed toward the direction of negative PnLs (unprofitability for the pool):

This is consistent with our proposed hypothesis in which the MEV bot0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

engages in statistical arbitrage at large size in a manner which is on average

profitable for the wallet and unprofitable for the Uniswap liquidity pool. (You may feel that, visually, the difference between the two distributions does not seem so great; yet such a slight shift in the PnL distribution is responsible for so many millions of trader profit and liquidity pool loss!)

Curiously, in the course of our research, we noticed that this block builder abruptly stopped building blocks one month ago; as far as we can tell, the last block they built was block 16522347, on January 30th, 2023. Did they simply decide to stop building blocks? We do not believe so. Most active block builders in our data started building blocks relatively quickly after the Merge; however, there are two specific block builders that stand out as starting relatively 'late', namely manta-builder

and rsync-builder.xyz

. In particular, manta-builder

is associated with the second lowest (unprofitable for the liquidity pool) PnL across all highly active block builders, second only to geth^go1.19.1...linux

itself! Furthermore, when we examine the blocks built by manta-builder

, we notice a very similar pattern: at the top of every block built by manta-builder

, such as block 16487128, there are transactions made to the MEV bot0xf8b721bff6bf7095a0e10791ce8f998baa254fd0

.

Are the two MEV contracts 0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

(associated with the block builder which has signature geth^go1.19.1...linux

) and 0xf8b721bff6bf7095a0e10791ce8f998baa254fd0

(associated with manta-builder

) connected? On first sight, it is difficult to identify any relationship between the two. However, if we use Panoramix to decompile the bytecode of the two contracts:

- Decompilation of 0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94)
- Decompilation of 0xf8b721bff6bf7095a0e10791ce8f998baa254fd0

We find that a remarkable number of their function signatures overlap, suggesting that they are in fact related to the same controlling entity! For example, the function signatures unknowne148cd50(uint128 _param1, uint256 _param2)

, unknown6dbf2fa0(uint128 _param1, uint256 _param2, array _param3)

, and unknown048867dc()

are present in both decompilations with very similar, if not identical, internal logic; for code of this complexity, it is unthinkable that this could be the result of pure coincidence. It is also straightforward to verify that the decompilations of other MEV contracts, such as 0xa69babef1ca67a37ffaf7a485dfff3382056e78c

, do not share any

of the same congruences. Although there has beenlimited speculation on the identity of manta-builder

, we believe we are the first to publicly identify an association with a prior (and very active) block builder.

The blocks manta-builder

yield an average PnL of -3.19 basis points for the Uniswap ETH/USDC liquidity pool, i.e.,

slightly less toxic than the blocks built by geth^go1.19.1...linux

. If we examine the distribution of average block PnLs for bocks built by manta-builder

, as we did before, the distribution is qualitatively rather similar:

The difference in PnL, then, is perhaps simply just due to higher or lower price volatility in a given time period rather than to any change in the underlying strategy employed by the controller of these MEV bots.

Examination of other block builders

We have delved fairly deeply into the identity and activity of a single block builder; clearly, we could spend quite a while if we were to exhaustively profile the behavior of each prominent builder! The fact that different builders have their own unique idiosyncrasies (some may essentially be purpose-made for a specific group's bundles, whereas others may be associated with the submission of bundles from independent searchers, and so on) further complicates any systematic analysis.

It is, at the very least, interesting to look at the average PnL (across built blocks) from each block builder who has built at least 1,000 blocks:

The mysterious builder who signs off as both geth^go1.19.1...linux

and manta-builder

, which we discussed above, easily tops the list, but other block builders are also robustly associated with flow toxicity, such as rsync-builder.xyz

or beaverbuild.org

. On the other hand, some block builders are actually associated with nontoxic

swap flow, such as blocks built bybuilder0x69

, whose blocks seem to generally deliver several basis points' worth of profit to the Uniswap ETH/USDC liquidity pool.. (Why the extraData

field has multiple distinct variants for the builder0x69

signature remains unclear to me.) builder0x69

supplies an endpoint for independent searchers, and as such blocks that they build do not exhibit any obvious consistency in terms of which MEV bots' transactions land at the top of the block. (Compare, for example, blocks 16494139 and 16494137.)

We may also wonder to what extent different block builders are specialized

toward the task of building blocks which extract value from the Uniswap ETH/USDC pool. In the above table, ethusdc_blocks

denotes the number of blocks built by each block builder, all_blocks

denotes the total

number of blocks built by that block builder (including blocks which have no ETH/USDC swaps), and ratio

is equal to the former divided by the latter. We see that approximately 20% of the blocks built by geth^go1.19.1...linux

or manta-builder

include ETH/USDC swaps on Uniswap: a noticeably higher proportion than any other block builder on this list. One likely interpretation, therefore, is that the manta-builder

entity focuses on statistical arbitrage of major trading pairs; in comparison, blocks built by other block builders may include arbitrages on a broader range of longer-tail assets.

Similarly, we might ask: Among blocks which contain at least one ETH/USDC swap on Uniswap, which block builders are the most prominent, and how much pool PnL can be attributed to each block builder?

(A clarifying note on methodology: Previously, we specified that average normalized PnL in basis points was computed per-block excluding overly small swaps, to avoid the inclusion of blocks with a single, small. In the calculation of total notional PnL, we simply include all swap data regardless of size.)

In this table, pnl

is the total PnL, in United States dollars, attributable to blocks built by a specific block builder, and ratio_ethusdc

is the percentage of post-Merge blocks with ETH/USDC swaps built by each block builder. (Note that ratio_ethusdc

does not sum to 100% because there are many blocks built by block builders not on this list, either because they did not specify any extraData

or because they did not build more than 1,000 blocks.)

Despite making up under 3% of post-Merge blocks with ETH/USDC swaps, blocks built by geth^go1.19.1...linux

and manta-builder

are responsible for a sum total of -1.2 million in PnL for the Uniswap ETH/USDC liquidity pool! This is remarkable, considering that the aggregate PnL

of the liquidity pool in the same time duration is approximately +1 million (liquidity providers are actually making money). If we restrict further to swaps routed through either 0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

or 0xf8b721bff6bf7095a0e10791ce8f998baa254fd0

, we find that such swaps are attributable to a pool PnL of nearly -2.2 million dollars, even though they only make up 1% of all post-Merge swaps!

A very brief aside: in a previous analysis,, we estimated total Uniswap LP losses to approach -40 million dollars based on short-term Binance price markouts; as such, the reader may find it curious that the post-Merge period actually showed net positive

profitability for LPs, even if only to the tune of +1 million. Rather intriguingly, it seems that since the time that post was written,

the performance of the Uniswap pool has significantly improved, as we can see from this updated graph of cumulative pool PnL:

This is perhaps attributable to low price volatility since the Merge, at least relative to the year-long period prior to the Merge,

which yields fewer opportunities for CEX/DEX arbitrage and, consequently, a generally higher PnL for Uniswap liquidity providers.

In the specific case of MEV bots0x57c1e0c2adf6eecdb135bcf9ec5f23b319be2c94

and 0xf8b721bff6bf7095a0e10791ce8f998baa254fd0

, there is a clear one-to-one mapping from block builder to toxic MEV bot, and analysis on the level of block builders does not necessarily get us much more insight than looking at wallets or MEV contracts alone. However, in the case of blocks built by builder0x69

, one might plausibly argue that seeing a block built by builder0x69

is a fairly credible signal of nontoxic swap flow, at least on a historical basis.

Of course, even within blocks built by the same block builder, the predicted toxicity of swap flow still depends heavily upon the originating wallet as well as on the contract that a given swap interacts with (as we have previously observed). Generally, these may even be dominant factors. Nevertheless, especially for the cases of swaps coming from ambiguous sources (e.g.,

fresh wallets), the identity of a block builder may still be quite valuable at the margin with regard to the accurate classification of swap flow as toxic versus nontoxic.

If we look at blocks that are either built by the manta-builder

entity (including both of its addresses) or by builder0x69

(including all manifestations of its extraData

signature) and examine the distribution of notional ETH/USDC swap sizes, we notice that while both distributions are bimodal, manta-builder

's blocks are particularly enriched in the rightmost peak of larger swaps:

This is likely caused by the inclusion of its own MEV swaps at the top of the block. In contrast, if we plot the same distributions but excluding

swaps from the two manta-builder

-associated MEV contract addresses we profiled above, the distributions of notional swap sizes appear much more similar:

The concordance of notional swap size distributions after excluding manta-builder

's own statistical arbitrage suggests that both block builders include nontoxic flow, but only manta-builder

is successfully capitalizing, at scale, upon statistical alphas that predict price movement in the next 5-10 minutes. (Note that the same pattern is reflected in the distributions of markout PnLs, both on a per-swap and a per-block level; however, in those cases, the difference is much less visually clear.) Given that builder0x69

supplies a public endpoint for bundle submission, this is perhaps not too surprising of a result: if you are in possession of proprietary, high-value trading signals, you will likely prefer to build your own blocks so that you can ensure prioritization of your transactions above others and capture as much value as possible.

Conclusion

To summarize: we have made the following observations:

- A small number of block builders are responsible for building most blocks on Ethereum
- Certain block builders are associated with very toxic Uniswap swap flow, whereas others are associated with very nontoxic swap flow
- Two specific block builders, namely geth^go1.19.1...linux

and manta-builder

, are almost certainly operated by the same entity, who specializes in statistical arbitrage and delivers highly toxic swap flow to Uniswap liquidity providers

- The blocks built by builder0x69

are, in contrast, quite nontoxic

Overall, we find evidence that identifying the builder which builds a given block can, in some cases, provide information about the toxicity of swaps contained within that block. This is context-dependent; for example, if most of the toxicity associated with a block builder originates from the same MEV bots whose swaps are prioritized block builder, then examining block builder identities adds little information (although it may satisfy our curiosity). However, in other cases, where block builders collate various transactions from different sources, this may be a relatively credible sign of nontoxicity.

-0xfbifemboy