Preamble

Requests for proposals are not to be taken as prescriptive or exhaustive. The community is encouraged to submit proposals that build upon the ideas presented in this document. The scope of the project may change based on the proposal received. The primary intent of this document is to provide a starting point from which to achieve the goals outlined, and what is ultimately implemented may differ from the initial proposal.

Background

MEV Blocker, co-developed by CoW DAO, is a leading private RPC endpoint notable for its wide searcher coverage and excellent MEV refund performance.

It effectively prevents malicious MEV attacks such as front-running and sandwiching, while non-harmful MEV opportunities like backruns are auctioned to a permissionless set of searchers.

90% of these proceeds are then refunded to the origin of the transaction or a specified refundRecipient

.

Following CiP-40, MEV Blocker introduced stringent rules for builders connecting to its transaction flow.

Non-compliance leads to penalties.

The comprehensive rules are available atCoW Docs, emphasizing maximizing user refunds and ensuring the inclusion of multiple MEV Blocker backruns over external ones.

Goal

The objective is to develop an open-source tool that confirms the maximization of user refunds for each included MEV Blocker transaction and flags rule violations.

Specification

- Transaction Identification:
- Determine the position p

of the MEV Blocker transaction within a block.

- Track all bundles related to this transaction that were broadcasted within the first 10 seconds prior to the block time (configurable).
- Determine the position p

of the MEV Blocker transaction within a block.

- Track all bundles related to this transaction that were broadcasted within the first 10 seconds prior to the block time (configurable).
- Bundle Simulation and Sorting:
- Simulate each bundle with the target transaction at position p

.

- If successful, note the payment to the builder; if not, disregard the bundle.
- Order bundles by descending payment value.
- Simulate each bundle with the target transaction at position p

.

- If successful, note the payment to the builder; if not, disregard the bundle.
- Order bundles by descending payment value.
- State Management:
- Record the block state s

post-application of the target transaction at position p

.

- For each sorted bundle, simulate only the backrun segment at position p+1

.

- If successful, update s

with the new state post-simulation, adjust p

, and log the backrun transaction hash; if not continue with next bundle.

- Record the block state s

post-application of the target transaction at position p

.

- For each sorted bundle, simulate only the backrun segment at position p+1

.

- If successful, update s

with the new state post-simulation, adjust p

, and log the backrun transaction hash; if not continue with next bundle.

- Verification:
- Verify the inclusion of each logged transaction hash within the target block, followed by a refund transaction to the specified recipient.
- If a backrun is omitted, re-simulate it at the block's end to determine if it was replaced by a less optimal transaction for the user.
- Verify the inclusion of each logged transaction hash within the target block, followed by a refund transaction to the specified recipient.
- If a backrun is omitted, re-simulate it at the block's end to determine if it was replaced by a less optimal transaction for the user.

Ressources

All MEV Blocker transactions and bundles are posted with a slight delay in the mevblocker.raw_bundles

Dune table (cf. sample query).

Simulation of multiple transactions (e.g. all transaction of a block leading up to p + bundle[s]) is available using the non-standard [trace_callMany

](https://www.quicknode.com/docs/ethereum/trace_callMany) RPC call (supported by e.g. Nethermind, Reth, Erigon).

Alternatively, more advanced simulation tools such as Temper or Tenderly can be used for counter-factual simulation of specific transaction at specific position in historic blocks.

For more background information on MEV Blocker generally, please seeOverview | CoW Protocol Documentation.

Deliverables

- A script for periodic execution (e.g., via cron job) to check and alert violations for specific periods, or
- A real-time system that monitors new blocks and processes them upon availability of necessary Dune data.

Additionally:

- Development tools such as a Docker container, .env

file setup, etc., for production deployment.

- Detailed documentation covering the operational logic of the tool and execution instructions.