Introduction

On April 2nd, a malicious Ethereum network participant stole $20M from aMEV searcher by exploiting a vulnerability in the mev-boost-relay

(see Flashbots’ post-mortem). In the following days, developers addressed the bug by releasing five patches to the broader mev-boost

ecosystem. These patches, alongside existing network latencies and validator strategies, resulted in a brief period of instability in the Ethereum network due to an increased rate of reorged blocks on April 6th. Reorgs are bad for network health because they decrease the rate of block production and reduce settlement assurances.

In this post, motivated by the attack against the searcher and the temporary instability of the network, we explore the interplay between mev-boost

& consensus, unpack subtleties of Ethereum’s Proof-of-Stake mechanism, and enumerate some of the possible paths forward.

What is mev-boost & why is it important?

mev-boost

is a protocol designed by Flashbots and the community to mitigate the negative effects of Maximal Extractable Value (MEV) on the Ethereum network.

There are 3 actors in mev-boost

:

1. Relays

– mutually-trusted auctioneers that connect proposers to block builders.

1. Builders

– sophisticated entities who construct blocks in order to maximize MEV for themselves and the proposers.

1. Proposers

– Ethereum Proof-of-Stake validators.

The rough sequence of events per-block is:

1. A builder creates a block by receiving transactions from users, searchers, or other (private or public) orderflow.
2. The builder submits the block to a relay.
3. The relay validates that the block is valid and calculates how much it pays the proposer.
4. The relay sends a "blinded" header and a payment value to the proposer of the current slot.
5. The proposer evaluates all the bids they've received and signs the blinded header associated with the highest payment.
6. The proposer sends this signed header back to the relay.
7. The block gets published by the relay using their local beacon nodes and returned to the proposer. The rewards are distributed to the builder & proposer through transactions in the block and the block reward.

The relay is a mutually-trusted party that facilitates the fair exchange of block space (from the proposer) and transaction sequencing for MEV extraction (from the builder). The relay protects builders from MEV stealing, in which proposers copy builder transactions to take MEV for themselves instead of allocating it to the searcher/builder who discovered it. The relay protects proposers by (a) confirming the validity of the builder blocks, (b) processing hundreds of blocks per-slot on behalf of the proposer, and (c) ensuring the accuracy of the proposer payment.

mev-boost

is critical protocol infrastructure because it enables democratic access to MEV for all proposers without requiring trusted relationships with builders or searchers, which contributes to Ethereum's long-term decentralization.

Ethereum's Fork-Choice Rule & mev-boost

Before we dive into the attack and responses, we examine Ethereum's Proof-of-Stake (PoS) mechanism and the associated fork-choice rule. A fork-choice rule allows the network to reach consensus about the head of the chain. From Ethereum Reorgs After The Merge:

A fork choice rule is a function, evaluated by the client, that takes as input the set of blocks and other messages that have been seen, and outputs to the client what the "canonical chain" is. Fork choice rules are required because there may be multiple valid chains to choose from (eg. if two competing blocks with the same parent get published at the same time).

A lesser-known aspect of the fork-choice rule is its relationship to time, which has major implications for block production.

Slots & sub-slot periods

In Ethereum PoS, time is partitioned in 12 second increments called slots

. The PoS algorithm randomly assigns a validator the permission to propose

a block for that slot; this validator is referred to as the proposer. In the same slot, other validators are assigned the task of attesting (voting) for the block that is the head of the chain according to their local view by applying the fork-choice rule. The 12 second slot is subdivided into three phases, each consuming 4 seconds.

The events that take place in a slot are listed below using t=0

to denote the beginning of the slot.

The most critical moment in a slot is the attestation deadline

at t=4

. If an attesting validator has not seen a block by the attestation deadline, they will instead vote for the previously accepted head of the chain (according to the fork-choice rule). The earlier a block is proposed, the more time it has to propagate, and thus the more attestations it accumulates (because more validators see it by the attestation deadline).

From the network-health perspective, the best time for a block to be published is t=0

(as dictated by the spec). However, since the value of the block increases monotonically with time, proposers have the incentive to delay the publication of their blocks to allow more MEV to accumulate. See Timing games in Proof-of-Stake and this discussion for further details.

Historically, a proposer could publish a block well after the attestation deadline (even close to the end of the slot), as long as the next validator observed the block before they built their block for the subsequent slot. This is a result of the child block inheriting the weight of the parent block and the fork-choice rule terminating at a leaf node. Thus there was no downside to delaying block publication. In order to help push rational behavior (delaying the block publication) towards honest behavior (on-time publishing), “honest reorgs” were implemented.

Proposer boost & honest reorgs

Two new concepts were introduced into the consensus clients that have critical implications for the attestation deadline.

1. proposer boost

(PR) – attempts to minimize reorgbalancing-attacks by granting the proposer a fork-choice “boost” equivalent to40% of the full attestation weight. Importantly, this boost only lasts for the duration of the slot.

1. honest reorgs

(PR) – takes proposer boost and allows honest proposers to use it to forcibly reorg blocks that have attestation weight below 20%. This is implemented in Lighthouse and Prysm (as of v4.0 – the Capella release). This change is optional because it is a local decision made by the proposer, and does not impact the attesting validators’ behavior. As a result, there was no coordinated effort to roll it out in all clients simultaneously, nor was it associated to any specific hard-fork.

Note that honest reorgs are avoided in some special cases:

1. during epoch boundary blocks
2. if the chain is not finalizing
3. if the head of the chain is not from the slot prior to the reorged block

Condition 3 ensures that honest reorgs only ever remove a single block from the chain, which acts as a circuit breaker to allow the chain to continue producing blocks during periods of extreme network latency. This also reflects the proposer's reduced confidence in their view of the network, because they can no longer be certain that their proposer-boosted block will be seen as canonical.

The diagram below demonstrates how honest behavior changes to implement the reorg strategy.

In this scenario, let b1

represent a late block. Due to the lateness, b1 only has 19% of the attesting weight for slot n

. The remaining 81% of the attesting weight is allocated to the parent block HEAD

, because many attesters did not see b1

by the attestation deadline.

Without honest reorgs

, the proposer for slot n+1

sees b1

as the head of the chain and builds a child block b2

. The proposer makes no effort to reorg b1

, despite it only having 19% attesting weight. During slot n+1

, b2

has proposer boost, and assuming it was delivered on-time, b2

will become canonical by accumulating a majority of the attestations for that slot.

With honest reorgs

, the situation is much different. Now the proposer of slot n+1

sees that the 19% attestation weight on b1

is below the reorg threshold, so they build a block with HEAD

as the parent of b2

, and forcibly reorg b1

. When we reach the attestation deadline of slot n+1

, honest attesters will compare the relative weights of b1

(19%) vs b2

(40% from proposer boost). All the clients implement proposer boost, thus b2

will be seen as the head of the chain and will accumulate the slot n+1

attestations.

Relay & beacon node fixes in response to the unbundling attack

During the April 2nd unbundling attack, the proposer exploited a relay bug by sending an invalid signed header back to the relay. During the following days, the relay and the core-dev teams released a number of software patches to mitigate the risk of a repeat attack. The five major changes were the following:

1. Relay changes:
2. Check the DB for known malicious proposers (only ever used in prod by theultra sound relay, and has since been removed).
3. Check if the relay has already delivered a full block to the p2p network during that slot.
4. Introduce a uniform random delay in the range 0-500ms before the publication of the block (removed from all relays).

5. Check the DB for known malicious proposers (only ever used in prod by theultra sound relay, and has since been removed).

6. Check if the relay has already delivered a full block to the p2p network during that slot.

7. Introduce a uniform random delay in the range 0-500ms before the publication of the block (removed from all relays).

8. Beacon node changes (only for relay beacon nodes):

9. Validate the beacon block before

broadcasting it.

1. Check the network for equivocations before publishing a block.

2. Validate the beacon block before

broadcasting it.

1. Check the network for equivocations before publishing a block.

2. Check the DB for known malicious proposers (only ever used in prod by theultra sound relay, and has since been removed).

3. Check if the relay has already delivered a full block to the p2p network during that slot.

4. Introduce a uniform random delay in the range 0-500ms before the publication of the block (removed from all relays).

5. Validate the beacon block before

broadcasting it.

1. Check the network for equivocations before publishing a block.

The combination of these changes led to consensus instability that was exacerbated by the fact that a large percentage of validators are now using the honest reorg strategy described above.

Unforeseen consequences

Each of the 5 changes mentioned above introduced latency into the hot path of relay block publication, which increased the probability that relay blocks would be broadcast after the attestation deadline. The figure below shows the five checks in sequence and how the introduced latency could cause the block publication to exceed the attestation deadline.

Before these checks were implemented, a signed header arriving substantially later than t=0

, e.g. t=3

, would typically present no issues. The relay had a very low overhead, and thus would publish the block well before the attestation deadline at t=4

.

However, with the introduction of the latency from the five patches, the relay could now be partially responsible for a late broadcast. Let’s look at a hypothetical block publication below.

The relay receives the signed header from the proposer at t=3

. By t=4

, the relay is still executing checks and thus the broadcast happens after the attestation deadline. In this case, the combination of the proposer sending the signed header late and the relay introducing some additional latency resulted in the missed attestation deadline. Without honest reorgs, these blocks would have very likely made it on chain

. The honest proposers for the subsequent slot would not intentionally reorg the block for being late as we saw in Figure 2. With honest reorgs however, missing the attestation deadline means this block will be reorged by the next proposer.

As a result, in the days following the attack, the number of forked blocks increased dramatically.

The Metrika 2 week data shows that in the worst case, 13 blocks (4.3%) were reorged in an hour, which was ~5x more than normal. As the relays rolled out various changes, the sharp increase in the number of forked blocks became clear. Thanks to a great community effort from relay operators and the core-devs, once the impact was understood, many of the changes were rolled back and the network returned to a healthy state.

As of today, the most useful changes were the beacon node block validation and equivocation checks that take place before broadcasting. Malicious proposers can no longer execute the attack by sending an invalid header to the relay and also must ensure that the relay beacon nodes do not see the equivocating block before publishing. Despite this, the relay remains exposed to the more general equivocation attack as presented in Equivocation attacks in mev-boost and ePBS.

So what should we do?

In this post we highlighted how mev-boost

works and how critical it is for Ethereum consensus. We also double-clicked on some of the lesser-known aspects of Ethereum's fork-choice rule surrounding timing. Using the unbundling attack and the developers' response as a case study, we underscore the potential fragility of the timing-related aspects of the fork-choice rule, and its impact on the network's stability.

Given that, the research community should evaluate what is an "acceptable" amount of reorgs and consider the exposure to equivocation attacks more generally to determine if mitigations should be implemented.

Additionally, there are multiple future directions being actively explored:

1. Implementing "headlock" to protect mev-boost

against equivocation. This would also require changes in the consensus client software and likely a spec change to extend the attestation deadline.

1. Increasing the number and visibility of bug bounty programs for the mev-boost

software.

1. Expanding simulation software to explore how sub-slot timings can impact network stability. This could be used to evaluate how adjusting the attestation deadline could reduce reorgs.
2. Optimizing the block publication path on the relay to reduce unnecessary latency. This is already being explored.
3. Recognizing that mev-boost

is core-protocol functionality and absorbing it into the consensus clients, aka enshrined-PBS (ePBS). Two-slot ePBS is vulnerable to equivocation attacks, so implementing "headlock" remains an option.

1. Adding more hive and/or spec tests informed by issues around latency and the attestation deadline.
2. Encouraging relay client diversity by building additional implementations of the relay spec.
3. Considering an adjustment to the slashing penalties for equivocation, while keeping in mind that even a full 32 ETH slashing may not be enough to dissuade malicious behavior in the presence of extremely large MEV opportunities.
4. Revisiting the sub-slot timings and considering and adjustment of the block propagation phase (e.g. moving the attestation deadline from t=4

to t=6

).

Overall, we are excited by the renewed energy around MEV and the mev-boost

ecosystem. Through the unbundling attack and mitigations, we have come to understand the critical relationship between latency, mev-boost

, and the consensus mechanism; we hope that the protocol continues to harden in response.

If these topics are exciting and interesting to you, please reach out toGeorgios (email) or Mike (email).

Many thanks to Bert Miller, Danny Ryan, Alex Stokes, Francesco D'Amato, Michael Sproul, Terence Tsao, Frankie, Joachim Neu, Chris Hager, Matt Garnett, Charlie Noyes, and samczsun for their feedback on this article and Achal Srinivasan for designing the figures.