Summary

On April 3rd, 2023, a malicious proposer exploited the ultra sound relay through a vulnerability in the open sourced mev-boost-relay implementation maintained by Flashbots to steal ~$20M from multiple sandwich bots. The attack was possible because of a vulnerability in the majority of mev-boost relays (mev-boost-relay, Dreamboat) detailed below which has since been patched. In following up to this event, a related timing attack was identified and mitigated, although more research is needed to understand if this mitigation is worth its negative externalities.

mev-boost is an open-source proposer-builder separation (PBS) protocol that allows proposers to sell their blockspace to an open market of specialized actors called builders. Builders compete to create the most valuable block possible, mostly by aggregating many individual MEV searchers' bundles of transactions.

mev-boost works through a commit and reveal scheme where proposers commit to blocks created by builders without seeing their contents, by signing block headers. Only after a block header is signed are the block body and corresponding transactions revealed. A trusted third party called a relay facilitates this process. mev-boost is designed to allow block builders to send blocks that contain valuable MEV to validators without having to trust them. Removing the need for builders to trust validators ensures that every validator has equal access to MEV regardless of their size and is critical for ensuring the validator set of Ethereum remains decentralized. For more background on mev-boost, please see the mev-boost GitHub repository. For more background on mev-boost relays, please see mev-boost-relay, Optimistic Relay, mev-relay, or Dreamboat. There are also closed source relay implementations, e.g. Manifold Finance's.

The attack on April 3rd, 2023 was possible because the exploited relay revealed block bodies to the proposer so long as the proposer correctly signed a block header. However, the relay did not check if the block header that was signed was valid. In the case that the block header was signed but invalid, the relay would attempt to publish the block to the beacon chain, where beacon nodes would reject it. Crucially, regardless of whether the block was rejected by beacon nodes or not, the relay would still reveal the body to the proposer.

Having access to the block body allowed the malicious proposer to extract transactions from the stolen block and use them in their own block where it could exploit those transactions. In particular, the malicious proposer constructed their own block that broke the sandwich bots' sandwiches up and effectively stole their money.

Usually, proposers publishing a modified block would not only equivocate but their new block would have to race the relay block - which has a head start - to acquire attestations for the fork choice rule. However, in this case, the relay was not able to publish a block because the proposer returned an invalid block header. Therefore, the malicious proposer's new block was uncontested and they won the race automatically. This has been addressed by requiring the relay to successfully publish a block, thereby not sharing invalid blocks with proposers. The mitigations section covers this and future looking details at more length.

Furthermore, in following up on this incident, a potential further attack was discovered and acted on. A malicious proposer with two slots in a row could ask a relay for a block too late in its slot. The relay would still respond with the block body even if a proposer's slot was almost over and the proposer would miss their slot. Note that this attack also allows a proposer with a single slot to grief the relay as well by intentionally missing its slot and causing the builder's block to be revealed even if it won't be able to internalize the MEV generated by this.

A potential mitigation to this attack is to introduce a cutoff timing into the proposer's slot whereafter this time (e.g. 3 seconds) the relay will no longer return a block to the proposer. Relays began to roll out this mitigation in the evening of April 3rd UTC time with a 2 second cutoff, and notified other relays to do the same. After receiving credible reports of honest validators missing their slots the suggested timing cutoff was increased to 3 seconds. At the time of writing, this cutoff seems to be acceptable and not causing undue missed slots, but further investigation is needed to determine if this mitigation is worth the cost on honest proposers. This issue is elaborated further in the mitigations section.

It is critical that Ethereum has an MEV marketplace that is democratic and efficient. Vulnerabilities like those detailed in this post undermine the integrity of the MEV marketplace and the experience of its users. We call on the security, research, and open source communities to join us in hardening mev-boost and future enshrined PBS designs.

Timeline

- On March 16th 2023, the attacker, whose account was funded by ETH withdrawn from Aztec, registered a validator in epoch 187922.
- Around block 16964664 the attacker sent multiple transactions to the public mempool performing swaps with infinite slippage on pools with relatively high amounts of liquidity. These were crafted to bait sandwich bots into making specific trades on very high leverage.
- Several bots picked up these mempool transactions and attempted to sandwich them by constructing bundles that were sent to the builder0x69 builder.
- builder0x69 used these bundles to build a block that was sent to the ultra sound relay.
- At 11:29pm UTC on April 2nd 2023, the attacker requested a block header from the ultra sound relay.
- The attacker crafted an invalid block header by setting the state root and parent root to 0. They signed this block header and sent it to the ultra sound relay.
- The ultra sound relay, after receiving the invalid block header, attempted to publish the block header and

corresponding body. Local beacon nodes rejected this block as it was invalid.
- Nonetheless, the relay published the full block to the attacker.
- The attacker used the block to construct a block that drained the sandwich bots and proposed it in slot 6137846
- At 6:11am UTC, on April 3rd @punk3155 was first to write about it, published tweet.
- At 6:41am UTC, the ultra sound relay team notified the Flashbots team and a war room was established.
- At 7:47am UTC, the ultrasound relay team disables its relay.
- At 8:28am UTC, nrv fully reconstructed the attack and shared it in the Flashbots Discord.
- At 9:35am UTC, the Flashbots relay is disabled.
- At 10:06am UTC, a patch is released after testing on the Sepolia and Goerli testnets.
- At 10:53am UTC, relays began deploying the patch on mainnet.
- Around 7:40pm UTC, the ultra sound relay and Flashbots teams discover a potential attack where malicious proposers ask for a block too late into their slot. A warroom is established.
- Around 10:00pm UTC, samczsun notices that Prysm propagates blocks before validating them. This is confirmed with the Prysm team shortly thereafter.
- Around 10:20pm UTC, relays begin deploying a patch to not return blocks after 2 seconds into the proposer's slot and are notified to ensure their CL nodes validate blocks before propagating them.
- Around 10:00am UTC on April 4th, some relays begin noticing missed slots caused by the 2 second cutoff and up the cutoff to 3 seconds.

Mitigations

April 3rd production exploit - relay handles signed invalid block headers wrong

To fix the vulnerability exploited on April 3rd where the relay handled signed but invalid incorrectly, mev-boost relays were patched to not return the block body to proposers if it fails to successfully publish the block to the beacon nodes. See the patch for the mev-boost-relay here.

Note that this mev-boost-relay patch requires modification to the default behavior of CL nodes to work because CL nodes broadcast blocks before verifying them. Relays want the opposite behavior - verification before broadcasting - to prevent their CL nodes from broadcasting an invalid block. See an example patch here.

Another possible future mitigation is for mev-boost relays to not return the block body to proposers at all. This serves to further increase the head start of the relay in a potential block propagation race with a malicious proposer. Doing so needs to be coupled with a new mev-boost release because mev-boost by default expects block bodies to be returned.

However, even if they are not sent the block directly, malicious proposers can receive the block body earlier by running beacon nodes themselves. At the limit a malicious proposer may even "eclipse" relay beacon nodes entirely by controlling a sufficient number of nodes that they monopolize all connections to and from a relay's beacon nodes. To mitigate eclipse attacks in the short term mev-boost relays are looking to gossip blocks directly to professional block propagation networks, but this attack vector needs more study.

Proposers intentionally calling the relay too late to miss their slot but receive a block

To mitigate the impact of the potential attack where a malicious proposer intentionally asks a relay for a block too late a cutoff timing was introduced. With this cutoff mev-boost relays will only return a block body up until a certain time into the proposer's slot. Historical data was used to attempt to set the right cutoff point. Initially, 2 seconds was chosen but this was increased to 3 seconds after reports from honest validators that missed their slots. See the patch for the mev-boost-relay here.

This attack is difficult to manage because an honest proposer calling the relay late and a malicious proposer doing so are indistinguishable. There are many reasons for why an honest proposer would wait to call the relay late: they are geographically far from the relay, they are waiting to allow builders to optimize for MEV, etc. As of the time of writing, a cutoff of 3 seconds seems to not induce more missed slots from honest validators, but we continue to monitor and analyze the situation. More data and research is needed.

Looking forward

First, we'd like to note mev-boost and the mev-boost-relay both have bug bounty programs. A potentially expanded bug bounty program is being coordinated.

Second, we encourage searchers to think carefully about how much MEV their strategies expose, willingly or inadvertently, given the extremely adversarial nature of the MEV market and to take actions to mitigate potential attacks. In light of recent events, we ask searchers to reevaluate all risks in the marketplace: code risks, networking risks, reorg risks, smart contract risks, etc. and manage their risks appropriately.

Third, we call on the research and security community to carefully study vulnerabilities like these in the context of enshrined PBS, where variants may be applicable.

Finally, this is a call for your contributions, as a community, to ensure a healthy and robust PBS market today and into the

future:

- Developers: Contribute to mev-boost relay implementations (mev-boost-relay, mev-relay, or Dreamboat) and mev-boost.
- Security researchers: Audit mev-boost relay codebases and mev-boost, as well as contribute to designs for enshrined PBS.
- Researchers: Contribute to medium term research (e.g. the Optimistic Relay roadmap for enshrined PBS) and long term research (PEPC, enshrined PBS).
- Operators: consider running your own mev-boost relay.
- Searchers: study your contracts and be mindful of the MEV they may expose in light of various risks in the MEV marketplace.
- Others: Join MEV-Boost community calls.

Thank you

Thanks to @drakefjustin, the Flashbots team, @samczsun, @VitalikButerin, @mikeneuder, @alphamonad, @dannyryan, @dankrad, @barnabemonnot, @terencechain, @fradamt, @ralexstokes, @punk3155, @bbbb, and nrv, for their contributions in identifying, discussing and mitigating the vulnerabilities above. Thanks to @gakonst and @TimBeiko for reviewing a draft of this post-mortem.

This post-mortem is based on analysis taken place at the time of the writing. We will continue to monitor and update as more information and analysis comes to light.