This post was originally posted on 2022-08-04

Author: Hasu, @hasu

[

1200×800 180 KB

](https://collective.flashbots.net/uploads/default/original/1X/31586da68bae6645e3b2a35869ce95100448aebc.jpeg)

Over the last few days, mev-boost has been a hot discussion topic in the community. We want to take this opportunity to highlight and educate the design and trust assumptions that make mev-boost possible as a merge-ready PBS solution.

First, it's important to note that the reports of new issues or trust assumptions being discovered are false. Everything discussed in this post has been known properties of mev-boost and discussed in the initial mev-boost research post. Each issue has a known and accepted mitigation which is being implemented in consensus clients. The merge and mev-boost are both on track for launch at the scheduled timeline. You can keep an eye on the readiness of each consensus client and node operator on the mev-boost landing page over at boost.flashbots.net

With this said, let's take a deep look at the mev-boost design, trust assumptions and why they were chosen, and how we mitigate liveness risks to Ethereum.

Block withholding attacks

To bring you up to speed, mev-boost is middleware that validators can run to outsource their block construction to a market of block builders. In the middle of builders and validators sit "relays", which are in charge of facilitating smooth commerce between the two parties. The relay protects the builder from leaking any information about the block to the validators and ensures that even small validators can participate in the builder market. At the same time, the relay protects the validator from receiving blocks that are invalid, overstate their bid to the validator, or are missing entirely. (More on this later in "Why was this design chosen?

")

Relays can connect to one or many builders, and we expect that there will be both variants. A relay connecting to many builders will aggregate their bids (fun fact: in a previous iteration, we called them builder aggregators or builder pools). The relay can see all the blocks submitted by the builders to confirm their validity and how much they pay to the validator. The relay then only submits the highest valid bid to the validator to sign.

Before validators can receive any bids from relays, they need toset up mev-boost and add relays to their mev-boost config. mev-boost is effectively just a relay aggregator or a local relay of relays. It will serve the validator the winning bid from all relays. A validator can connect to a small number of relays that aggregate all the builders, and many will probably do that. Other validators might connect to many relays.

If mev-boost has no relays in the config OR all relays are offline, then the beacon node (BN) will always fall back to constructing a block from the public mempool. So we know the risk is not all relays going offline. So what is the risk? It is that:

- The entire network connects to the same relay (not necessarily exclusively), AND
- That relay is the highest bidding relay, so its block is selected by validators, AND
- The relay sends the block header for signing (= it is not offline) but after receiving the validator signature, does not publish the final block to the network.

In this case, the same relay would keep suggesting blocks to validators, and these validators would keep signing them, and then no block would be published. The result is a series of empty slots. The network not making blocks is typically called a liveness issue

. (It is different from a DDOS attack because the affected nodes still fulfill all of their other network duties like publishing attestations, gossiping messages, etc.)

This fault can happen via a bug or a malicious relay. In the bug case, we expect the relay to realize their problem and fix it quickly or go offline entirely. So the more relevant case to protect against is a malicious relay launching a deliberate withholding attack

.

It is important to emphasize that a withholding attack

can immediately cascade into a liveness issue

for the entire network. That is because a malicious relay can lie about its bid to guarantee it is always selected by all validators that register with it. For example, imagine a malicious relay bidding an artificially high number that is higher than that of honest relays.

This attack is free for the relay because it never publishes the block and pays the validator, and yet all affected validators miss their slot for proposing.

Importantly, we are not concerned about 10% of the network connecting to a faulty relay, but only when a faulty relay is so popular that it creates systemic risk for the network. To recycle a famous saying, if 10% of validators miss their slot, it is their problem. If 100% of validators miss their slot, it is Ethereum's problem.

Mapping the solution space

So, let's say a popular relay is withholding blocks; how does the system recover? From first principles, the system recovers when any

of the below steps happen:

1. Validators remove the faulty relay from their mev-boost config (or turn off mev-boost entirely), OR
2. Other relays start outbidding the faulty relay, OR
3. The faulty relay goes offline entirely, OR
4. The faulty relay starts publishing blocks again

Looking at the "defenses" against the faulty relay, it becomes clear that cases 3-4 are within the control of the faulty relay, and case 2 is in the control of the other relays. But as validators, we are only interested in solutions that let us remove relays once we realize they are faulty.

In this solution space, there are two categories: local

and global solutions

. The local solutions are the most simple — the validator (or their mev-boost) can track the most recent performance of a relay. If a relay misrepresented payments or didn't publish blocks, the validator can automatically remove it. But this solution has a problem of local information.

The solution protects a single validator from a bad relay, but the next validator doesn't know about the bad relay. A local solution may be enough for Coinbase, Lido, Binance, Kraken, etc., because they control so many validators that their "local" is a good approximation for the network's "global" anyway — but it doesn't help smaller validators. These validators need to know how the relays performed on the slots of previous other validators

.

It seems that we need a global solution.

In a global solution, validators look at the global history of the network, not just their own, to remove relays. The community is building two global solutions ahead of the merge.

The first is a relay monitor operated by a trusted third party. This monitor watches the global performance of relays and can send a message a la "remove relay x now" to validators. So if any relay misbehaves, all validators who connect to the relay monitor will have their configs updated. Are there any new risks with this approach? We recall from earlier that an empty mev-boost config means that validators fall back to local block production. So the relay monitor can only temporarily remove relays from a validator's config, not add any new ones, or cause the validator to miss any slots.

The second solution, introduced in the call by Alex Stokes, is the circuit breaker. It works similarly to a relay monitor but does not rely on a third-party service. The circuit breaker is part of the beacon node. It looks at the local view of the network, and if the node sees x out of y missed slots in a row, it will disconnect from the builder network and fall back to producing blocks locally. There is some nuance about picking a good number for x because a small number would allow a large validator to miss slots on purpose to trigger the circuit breaker for the rest of the network and turn off their mev-boost. A large number could lead to many missed slots in a row.

The above and further, less-mature solutions are being discussed in this GitHub issue. Eventually, Enshrined PBS will do away with relays entirely, but this is still a few years away.

Why was this design chosen?

Due to the commit-reveal scheme used in mev-boost, relays can fail to release blocks after making the winning bid. This attack vector has been known from the start and it is a calculated choice. Why did we choose it?

The alternative would have been to continue with Flashbots' pre-PoS design that we call "Stage 1 PBS". In Stage 1 PBS, block builders send full blocks to validators in cleartext, and validators have to open a DOS-sensitive RPC to block builders.

The advantage of the Stage 1 design has been that validators can always look at the block to verify that the block is valid and how much it pays the validator. If no builder sends a block in time, the validator can fall back to local block construction and there is no risk of ever missing a slot.

But the disadvantage is that block builders need to trust validators not to steal MEV from them, and validators need to trust block builders not to DoS attack them. The unfortunate consequence is that only trusted validators and builders can participate in the PBS market.

So, while the old design has nice properties against block withholding attacks, it would have completely cut off solo stakers from MEV extraction. After public discussions with the Ethereum community and other key stakeholders, this cost was deemed too high and we settled in favor of a design that can include them.

Summary

I hope this post helps you frame the trust assumptions of mev-boost, why they were chosen, and what mitigations will be in place to avoid liveness risk to Ethereum. The relay monitor will be ready for the merge. The circuit breakers are being developed and some consensus clients already have them implemented. Together with validators practicing good relay hygiene (only using trusted relays until further mitigations are in place, setting up downtime monitors for their slots), I think we are in a good position for the merge.

References

- mev-boost architecture
- mev-boost homepage
- Why run mev-boost?
- liveness mitigations GitHub issue
- relay monitor
- circuit breaker proposal