Summary & Impact

mev-boost had a bug in decoding the getPayload

request body, which would fail if a deposit is included in the SignedBlindedBeaconBlock

sent to the relay. We are aware of three instances where this bug caused a missed slot.

The issue was resolved in [mev-boost v1.3.1

](https://github.com/flashbots/mev-boost/releases/tag/v1.3.1).

Timeline

Friday, 16.9.

- Flashbots was notified at 9am UTC by Enrico del Fante (from the Teku team) about a report of a failed getPayload

call to mev-boost.

msg="could not decode payload (signed blinded beacon block)" error="json: unknown field "proof""

- The Flashbots team, together with Ethereum core dev community, began investigating and identified the issue shortly after.
- The cause of the error was that the [Deposit

type](https://github.com/flashbots/go-boost-utils/blob/main/types/builder.go#L79-L90), which can be included in the BeaconBlockBody

and BlindedBeaconBlockBody

types, was missing the container structure with the proof

field.

- The issue was initially fixed in go-boost-utils PR #38 and mev-boost v1.3.0 was released at 1pm UTC.
- A follow-up issue in the proof decoding was discovered and reported at 5pm UTC by Stefan Bratanov (from the Teku team).
- The final solution was implemented in go-boost-utils PR #40 with the correct proof decoding. A big help was the inspiration by github.com/attestantio/go-eth2-client (thanks @jgm) and the Prysm codebase to double-check and verify the implementation.
- mev-boost v1.3.1 was released with the final solution at 8pm UTC

Corrective and Preventative Measures

- We've improved automatic testing on both go-boost-utils and mev-boost, by using additional static analysis and linting tools as well as additional test vectors.
- We've compared the implementation, encoding, and hashing algorithms from go-boost-utils with those of github.com/attestantio/go-eth2-client and Prysm, and can confirm that they are compatible.
- We've incorporated test-vectors from ethereum/consensus-spec-tests to both go-boost-utils and mev-boost. These test vectors include all possible fields (including deposits, slashings, etc.). These tests are run on every commit and pull request, which would catch any regressions if they were to happen.
- We've started an additional release checklist with steps taken before each release of mev-boost and go-boost-utils to provide additional security.
- We're continuing to invest into our security processes, both internally and with the help of external security advice.
- All mev-boost releases go through a signoff process with multiple parties, including node operators testing the latest release candidate on test networks.
- We're continuing to work closely with the core dev community on hardening mev-boost.

Shoutout to @tbenr, @stefanbratanov, @lightclient, @terencechain, @tersec, @justintraglia (and many others) for supporting the investigation and implementation, reviewing the solution as well as for further improvements and testing on

the various codebases.