Flashbots Discord member @nrv

analyzed the mev-boost-relay codebase and disclosed three vulnerabilities between April 23 and April 26th, 2023.

1. Race condition in getPayload
2. JSON parser differential for validator registrations
3. Minor: validator registration timestamp overflow

We immediately investigated the disclosures and resolved the issues between April 24 and April 28. Special thanks to @nrv

for the detailed disclosure and ongoing support, as well as for reviewing the draft of this post

@nrv

has gracefully requested that the bounty rewards be donated to Médecins Sans Frontières.

If you find a security vulnerability, on this project or any other initiative related to Flashbots, please let us know by sending an email to security@flashbots.net.

Race condition in getPayload

Details

- Assuming two perfectly timed getPayload requests, each with a different signed blinded beacon block
- Both requests simultaneously arrive at the check whether the block has already been delivered at the same time
- During loading the response and updating the Redis state with the latest delivered payload, there was a small time window during which both requests might have succeeded. That time window should be about the duration of the two Redis requests, which should typically be less than 10ms.
- Successful execution would have resulted in the attacker obtaining the payload for both bids, and would incur a guaranteed slashing because two different signed beacon blocks were submitted.

Resolution

- Use optimistic locking for getPayload requests, in order to remove the race condition
- Use transaction with optimistic lock to eliminate race by metachris · Pull Request #365 · flashbots/mev-boost-relay · GitHub

Impact

- We were able to confirm through the Flashbots relay logs that this has not been exploited.

JSON parser differential for validator registrations

Details

- mev-boost-relay used both jsonparser and the stdlib json parser to decode validator registrations.
- But their implementation differs in which field was used when a particular field is present multiple times in the raw JSON bytes.
- This would allow a validator to pass the validator-registration checks with different values than would be used later on when saving the registration!

For instance, consider the following JSON payload:

{"Field": "a", "Field": "b"}

One library would return "a"

and the other library would return "b"

Resolution

- registerValidator with fast JSON decoding only by metachris · Pull Request #369 · flashbots/mev-boost-relay · GitHub

Impact

- We were able to confirm through the Flashbots relay logs and database that this has not been exploited.

Minor: validator registration timestamp overflow

Details

- If a validator registration contained an incorrect huge value for the timestamp, it would inadvertently pass registration checks and could register a validator with a timestamp in the past.
- time.Unix

is dangerous because it can allow for an integer overflow:

sec = 0x7fffffffffffffff

func Unix(sec int64, nsec int64) Time { ... return unixTime(sec, int32(nsec)) }

func unixTime(sec int64, nsec int32) Time { return Time{uint64(nsec), sec + unixToInternal, Local} // unixToInternal = 62135596800 }

type Time struct { ... ext int64 // This overflows into 0x8000000E7791F6FF.. which is a negative value }

Resolution

- registerValidator with fast JSON decoding only by metachris · Pull Request #369 · flashbots/mev-boost-relay · GitHub

Impact

- We were able to confirm through the Flashbots relay logs and database that this has not been exploited.