OpenZeppelin | security

# Mantle Node, Batcher, Proposer, and Tooling Incremental Audit

MANTLE

March 15, 2024

# Table of Contents

# Summary

| | |
|---|---|
| Type | Layer 2 |
| Timeline | From 2024-02-12<br>To 2024-02-28 |
| Languages | Go |

| | |
|---|---|
| Total Issues | 12 (3 resolved) |
| Critical Severity Issues | 0 (0 resolved) |
| High Severity Issues | 0 (0 resolved) |
| Medium Severity Issues | 1 (1 resolved) |
| Low Severity Issues | 9 (0 resolved) |
| Notes & Additional Information | 2 (2 resolved) |

# Scope

We audited three pull requests with the `op-node`, `op-batcher`, `op-proposer`, `op-chain-ops`, and `op-service` components being in scope:

Pull request #72 from the base commit at 68f2533 and the head commit at bb0ff70. In scope were the following files:

```
mantle-v2
├── op-batcher
│   ├── batcher
│   │   ├── batch_submitter.go
│   │   ├── channel_manager.go
│   │   ├── config.go
│   │   ├── driver_da.go
│   │   └── driver.go
│   ├── common
│   │   ├── types.go
│   │   └── utils.go
│   ├── flags
│   │   └── flags.go
│   └── metrics
│       ├── metrics.go
│       └── noop.go
├── op-chain-ops
│   ├── cmd
│   │   ├── check-migration
│   │   │   └── main.go
│   │   ├── op-migrate
│   │   │   └── main.go
│   │   ├── rollover
│   │   │   └── main.go
│   │   └── withdrawals
│   │       └── main.go
│   ├── crossdomain
│   │   ├── encoding.go
│   │   ├── hashing.go
│   │   ├── legacy_abi.go
│   │   ├── legacy_withdrawal.go
│   │   ├── message.go
│   │   ├── migrate.go
│   │   ├── params.go
│   │   ├── withdrawal.go
│   │   ├── withdrawals.go
│   │   └── witness.go
│   ├── eof
│   │   └── eof_crawler.go
│   ├── ether
```

```
│   │   ├── addresses.go
│   │   ├── cli.go
│   │   ├── migrate.go
│   │   └── storage.go
│   ├── genesis
│   │   ├── check.go
│   │   ├── config.go
│   │   ├── db_migration.go
│   │   ├── genesis.go
│   │   ├── layer_one.go
│   │   ├── layer_two.go
│   │   └── setters.go
│   ├── immutables
│   │   └── immutables.go
│   └── util
│       └── util.go
├── op-node
│   ├── chaincfg
│   │   └── chains.go
│   ├── cmd
│   │   └── genesis
│   │       └── cmd.go
│   ├── eth
│   │   ├── sync_status.go
│   │   └── types.go
│   ├── flags
│   │   └── flags.go
│   ├── metrics
│   │   └── metrics.go
│   ├── node
│   │   ├── config.go
│   │   └── node.go
│   ├── rollup
│   │   ├── da
│   │   │   └── datastore.go
│   │   ├── derive
│   │   │   ├── attributes.go
│   │   │   ├── calldata_source.go
│   │   │   ├── channel_bank.go
│   │   │   ├── channel_in_reader.go
│   │   │   ├── deposit_log.go
│   │   │   ├── engine_consolidate.go
│   │   │   ├── engine_queue.go
│   │   │   ├── error.go
│   │   │   ├── frame.go
│   │   │   ├── l1_block_info.go
│   │   │   ├── l1_retrieval.go
│   │   │   ├── l2block_util.go
│   │   │   ├── payload_util.go
│   │   │   ├── pipeline.go
│   │   │   └── system_config.go
│   │   ├── driver
│   │   │   ├── driver.go
│   │   │   └── state.go
│   │   ├── sync
│   │   │   ├── config.go
```

```
│   │   │   └── start.go
│   │   └── types.go
│   ├── service.go
│   ├── service_mantle.go
│   ├── sources
│   │   └── sync_client.go
│   └── withdrawals
│       └── utils.go
└── op-service
    ├── crypto
    │   └── signature.go
    └── txmgr
        └── cli.go
```

Pull request #89 from the base commit at 5e3886c and the head commit at 76959dd. In scope were the following files:

```
mantle-v2
└── op-node
    └── rollup
        └── derive
            ├── deposit_log.go
            └── l1_block_info.go
```

Pull request #98 from the base commit at 0f0861b and the head commit at 365f02a. In scope were the following files:

```
mantle-v2
└── op-chain-ops
    ├── genesis
    │   └── config.go
    └── immutables
        └── immutables.go
```

# System Overview

Mantle V2 is a layer 2 (L2) scaling solution for Ethereum that uses fraud proofs instead of validity proofs for its security. The protocol aims to provide low transaction fees and high throughput while maintaining full EVM compatibility. Mantle V2 is built on top of Ethereum using the OP Stack and therefore shares many similarities with Optimism. The `op-node`, `op-batcher`, and `op-proposer` components collectively comprise the consensus layer that keeps the chain running and glues all the other layer 1 and layer 2 components together (though there is no consensus as per the Ethereum understanding). Adding `op-geth` to this set, which is the execution layer, makes up the Mantle network.

The `op-node` component is responsible for deriving the layer 2 blockchain which means that it listens for specific layer 1 events and layer 2 transactions, and keeps the chain running by continuously grouping this data into batches and passing them to the execution client in order to produce layer 2 blocks. The `op-batcher` component, in turn, listens to `op-node` for fresh batches, groups the batches into channels, compresses them, splits them into frames, and posts frames to the data availability layer.

The `op-proposer` component listens to `op-node` for the so-called layer 2 output roots which are Merkle roots. These are then posted to the layer 1 so that the smart contracts on layer 1 can receive messages from layer 2. This includes ERC-20 token bridging as well as general message passing. The `op-chain-ops` component is a tool that facilitates the administration of the chain. The `op-service` component contains common utilities used by other parts of the codebase.

# Summary of Changes

Presented below is a summary of changes to the in-scope components grouped by pull request and component.

# Pull Request #72

### op-node

- MantleDA is now used as a data availability provider. Prior to that, Ethereum calldata was used.
- The base fee parameter was introduced. It is set in the `SystemConfig` contract on Ethereum and then passed to the execution layer as part of the first transaction of each L2 block along with other `SystemConfig` parameters.
- The `requestsChannelBufferSize` constant was increased from 128 to 1024 and the logic around handling the corresponding channel was changed.
- The `unsafeL2PayloadsChannelBufferSize` constant was increased from 10 to 4096 and the logic around handling the corresponding channel was changed.
- Several improvements were merged from the upstream repository.

### op-batcher

- MantleDA is now used as a data availability provider. Prior to that, Ethereum calldata was used.

### op-proposer

- No changes

### op-chain-ops

- Support for the `GasPriceOracle` contract was added.
- Support for MNT as a native token was added.

### op-service

- The updates to the `op-service` component encompass support for the Key Management Service (KMS) from the Google Cloud Platform (GCP) in Ethereum operations. In case the Cloud HSM is expected to be used, new configuration parameters have been introduced to streamline the process.

# Pull Request #89

### op-node

- The new version of the deposited transaction event was added. In particular, the marshal and unmarshal functionality was added.

**op-batcher, op-proposer, op-chain-ops, and op-service**

- No changes.

## Pull Request #98

**op-chain-ops**

- The `L1_MNT_ADDRESS` constant was added.

**op-node, op-batcher, op-proposer, and op-service**

- No changes.

# Security Model and Trust Assumptions

MantleDA was treated as a black-box during this engagement as we were not provided with access to the documentation, tests, or the development instance of the system. Similarly, it was also assumed that the in-scope code uses MantleDA correctly.

Having said that, MantleDA is an emerging technology and thus has a higher chance of having bugs. The current way to recover from a MantleDA outage or malfunctioning is to stop the sequencer, switch data availability to Ethereum calldata in the configuration file, and start the sequencer again. This means there is no automatic fallback mechanism and, in case such an outage happens, the Mantle blockchain loses liveness until the sequencer is restarted with the new config.

## Privileged Roles

The `op-batcher` component uses a private key to interact with the MantleDA contract which defines which MantleDA blobs should be used to derive the layer 2 chain.

# Medium Severity

## M-01 Data With Id 1 Cannot Be Retrieved From Mantle DA

The `RetrievalFramesFromDa` function of OP-Node implements the logic for retrieving frames from Mantle DA. It requires a `dataStoreId` value, which undergoes a check to verify if it is equal to or smaller than 0. As the `calldata_source` calls `RetrievalFramesFromDa` with the `dataStoreId` decreased by 1, attempting to retrieve data from Mantle DA with a `dataStoreId` equal to 1 becomes impossible.

Consider removing the check from the `RetrievalFramesFromDa` function to permit the processing of a `dataStoreId` with a value of 0. In addition, since the `dataStoreId` is of type `uint32` and so cannot be smaller than 0, the check becomes unnecessary.

***Update:*** *Resolved in pull request #117.*

# Low Severity

## L-01 `RequestL2Range` Does Not Return Error if Channel Is Full

The `RequestL2Range` function queues a range of L2 blocks and returns early if the channel is full. However, it does not return an error in this case which means that the partial data will be processed.

Consider returning an error in case the channel is full in order to make callers aware.

***Update:*** *Acknowledged, not resolved. The Mantle team stated:*

> *Not a valid issue.*

## L-02 Witness Data Reader Skips the Last Line if There Is No New Line

The `ReadWitnessData` function of OP-Chain-Ops utilizes `bufio`'s `NewReader` to iterate through the file and retrieve all entries. To read each line, the function employs `ReadString`, which reads until a new line is encountered. However, this approach poses an issue – if the last line of the file lacks a new line character (as is the case for text file `witness.txt`), the reader will trigger an `EOF` error, causing an early break from the loop failing to read the last line.

Consider redesigning the function logic in such a way the last line will be read correctly even if it does not end with a new line.

***Update:*** *Acknowledged, not resolved. The Mantle team stated:*

> *Acknowledged, only used for upgrading, will not fix it.*

## L-03 Missing Type Conversion

The `NewWithdrawal` function of OP-Chains-Ops returns a `Withdrawal` struct with filled values. The `Data` field, which is of type `hexutil.Bytes`, is assigned `data` of type `[]byte`. However, the assigned value should first be converted to the correct type using the `hexutil.Bytes` function.

Consider converting the `data` parameter to the `hexutil.Bytes` type.

***Update:*** *Not resolved. The Mantle team stated:*

> *Not a valid issue.*

## L-04 Tests Panic and Fail

The proposed changes are not thoroughly covered by the test suite. In addition, there are multiple tests that fail, making it difficult to confirm the correctness of the implementation. The following components require additional testing:

- op-node
- op-batcher
- op-chain-ops

Consider reviewing the test suite for the above mentioned components to improve code quality.

***Update:*** *Not resolved.*

## L-05 Lack of Input Validation

The `BaseFee` parameter of type `big.Int` is checked for not being `nil`. However, a `big.Int` can be a negative number which does not make sense for `BaseFee` given that it should always be positive.

Consider checking the `BaseFee` parameter both for not being `nil` and being a positive number just like other `big.Int` parameters (e.g., chain ids are validated to be greater than 0 and not equal to 0).

***Update:*** *Acknowledged, will resolve.*

## L-06 The Mantle DA Status Is Not Cleared on OP-Batcher Start

Upon starting the OP-Batcher, the state is cleaned but the Mantle DA status is not.

Consider calling the `clearMantleDAStatus` function to clear the Mantle DA status.

***Update:*** *Acknowledged, not resolved.*

## L-07 Sleep Is Used to Wait for Channel Readiness

The sleep is used to wait `0.1 ms` for `sequencerCh` and `stepReqCh` to be ready. While this might mitigate the issue in the testing environment, it might behave differently in the production environment where load might be different and thus the sleep time might not provide the desired effect.

Consider refactoring the code to avoid using sleep and instead using more reliable mechanisms to sync the channels.

***Update:*** *Not resolved. The Mantle team stated:*

> *Not a valid issue.*

## L-08 Missing Connection Timeout for Contacting Mantle DA

The implemented connection to Mantle DA is missing a defined timeout option. This might lead to issues when servers accept connections but fail to respond to calls. There are the following occurrences of connections to Mantle DA:

- Connection to Mantle DA in `getFramesByDataStoreId` of OP-Node
- Connection to Mantle DA Indexer in `getFramesFromIndexerByDataStoreId` of OP-Node
- Connection to Mantle DA Disperser in `callEncode` of OP-Batcher

Consider adding a timeout mechanism to the above listed connections.

***Update:*** *Acknowledged, will resolve.*

## L-09 Unencrypted Connection to Mantle DA

The implemented connection to Mantle DA is unencrypted. In a production environment, it is generally recommended to encrypt the connection. There are the following occurrences of unencrypted connections to Mantle DA:

- Connection to Mantle DA in `getFramesByDataStoreId` of OP-Node
- Connection to Mantle DA Indexer in `getFramesFromIndexerByDataStoreId` of OP-Node
- Connection to Mantle DA Disperser in `callEncode` of OP-Batcher

Consider using encrypted connection instead.

***Update:*** *Acknowledged, will resolve.*

# Notes & Additional Information

## N-01 Typographical Errors

Throughout the codebase, several typographical errors were found:

- The constant name `TxConfirmDataSubmiited` should be `TxConfirmDataSubmitted`.
- The comment `openend` should be `opened`.
- This `comment` should be `L1_MANTLE_TOKEN` not `L1_MANTLE_TOEKN`.

Consider addressing the above typographical errors.

***Update:*** *Resolved in pull request #124.*

## N-02 Code Clarity

Throughout the codebase, several instances of redundant and unclear code were identified:

- The `NewMantleDataStore` function defined in `datastore.go` always returns `error` as `nil`. Thus, it is not necessary to return it and check it later.
- The `NewMantleDataStoreConfig` function returns a config and an error. However, the error is always `nil` which makes returning an error unnecessary.
- The `MockDataStoreConfig` variable is not used anywhere and should be removed.
- The `marshalDepositVersion1` function is not used anywhere and can be removed.
- The data from reply is retrieved twice in the `getFramesByDataStoreId` function. Consider calling `GetData` once and then use it in `log.Debug` and the return statement.
- The data from reply is retrieved twice in the `getFramesFromIndexerByDataStoreId` function. Consider calling `GetData` once and then use it in `log.Debug` and the return statement.
- The two `if` statements check for the `BaseFee` not being equal to `0`. Consider moving the second `if` statement inside the first `if` statement block and removing the unnecessary check.

- The `ds.cfg.MantleDaSwitch` `if` statement could be easier to read if it used an `if` statement with only true and false branches where the true branch contained code for handling MantleDA and the `else` branch contained code for Ethereum `calldata`.
- One of the significant updates to the codebase was addition of the MNT value which changed the protocol to handle `MNTValue` and `ETHValue`. The correct encoding starts with the `MNTValue` parameter followed by `ETHValue` parameter. However, in multiple places, the structures are initilaized in the reverse order. While this does not cause an issue since the parameter names are used, it does hinder readability. Consider changing the order of the `MNTValue` and `ETHValue` parameters in the `Decode` and `WithdrawalTransaction` functions.

***Update:*** *Resolved in pull request #125.*

# Conclusion

The audit uncovered one issue of medium severity, in addition to several other issues of a lower severity. Various recommendations have been provided to enhance the quality and documentation of the codebase. We also strongly recommend that Mantle implements more extensive QA procedures and tests before going live to prevent potentially undiscovered vulnerabilities from being exploited. This is especially crucial in areas of the code where testing was limited, such as the integration with Mantle DA. The Mantle team was very supportive throughout the audit period and answered questions in a timely manner.