Blockchain Protocol Security Analysis Report

Customer: Bitlayer

Date: 27/03/2024

We express our gratitude to the Bitlayer team for the collaborative engagement that enabled the execution of this Security Assessment of the project's consensus engine implementation.

Platform: Bitlayer (geth fork)

Language: Golang

Timeline: 26/02/2024 - 27/03/2024

Methodology: [Blockchain Protocol and Security Analysis Methodology]()

Review Scope

Repository	https://github.com/bitlayer-org/bitlayer-l2/
Commit	44ae502bb8c1032d06befd4166ae773a12a35c37
Remediation Commit	c2db1cc69feaa00b5e9cbd9625d547d5e5ab2fbc

Audit Summary

10/10	8/10	10/10	10/10
Security Score	Code quality score	Architecture quality score	Documentation quality score

Total 9.7/10

The system users should acknowledge all the risks summed up in the risks section of the report

2	1	0	1
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

Document

Name	Blockchain Protocol Code Review and Security Analysis Report for Bitlayer
Audited By	Yaroslav Bratashchuk
Approved By	Luciano Ciattaglia
Website	https://www.bitlayer.org/
Changelog	25/03/2024 - Preliminary Report; 27/03/2024 - Final Report

Table of Contents

System Overview

The Merlion consensus mechanism is a Proof of Stake Authority (PoSA) engine implemented for the Bitlayer project. It is designed to maintain network agreement with a set of validators who are responsible for creating new blocks and validating transactions. The protocol manages the validator set, handles their rotation based on stakes, and enforces rules to penalize inactive validators. It is built to be robust against unauthorized validators and ensures the integrity of block creation with cryptographic signatures. The engine integrates with Ethereum's existing infrastructure, like the EVM and state management, while adapting it for PoSA operations. This implementation provides an alternative to Ethereum's traditional Proof of Work system, emphasizing security and efficiency within the Bitlayer's forked Geth context.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's blockchain protocol project. Detailed scoring criteria can be referenced in the corresponding section of the Blockchain Protocol and Security Analysis Methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

The additions made by the Bitlayer team integrate seamlessly into the existing Ethereum codebase, reflecting a natural extension of the original architecture. These modifications are intuitively designed, such that they do not necessitate extensive in-code documentation, with their purposes and functions being readily apparent.

The provided documentation on the consensus rules is comprehensive, offering sufficient insight to facilitate a thorough understanding during the audit process.

Furthermore, the codebase, inclusive of tests, is well-commented, elucidating the rationale behind the code, which speaks to the thoughtful and deliberate development approach taken by the Bitlayer engineering team.

Code quality

The total Code Quality score is **8** out of **10**.

Our audit process did not surface any significant issues that warranted explicit mention in this report. However, minor typographical errors were observed in certain files (here and here) which could be addressed to polish the documentation.

The dependencies within the project are current, with the most recent updates from Geth integrated into the tip of the repository. This reflects a proactive approach to maintaining the codebase.

While the code cleanliness is generally good, there is an instance of an unused variable (here), along with some commented code that might be removed to avoid unnecessary clutter.

The code coverage stands out positively but presented an opportunity for enhancement, which was partially done during audit remediation period. We still advice to cover specific modifications related to the tracking of transaction call frames.

We noted some inconsistencies in the Staking ABI within the repository, detailed in a separate observation F-2024-1631 — a fix for which was provided during remediation period.

Other findings from our audit are based on opinions and do not influence our overall positive judgement of the codebase's quality.

Architecture quality

The total Architecture Quality score is **10** out of **10**.

The introduced Merlion consensus mechanism is a partial change and the overarching architecture of the project remains fundamentally unaffected. This strategic incorporation of a Proof of Stake Authority model enhances governance without disrupting the existing system's structure or scalability.

With focused enhancements in smart contract functionality and diligent management of dependencies, the project maintains its commitment to security and efficiency. While minor areas such as Staking ABI discrepancies and ensuring comprehensive test coverage present opportunities for refinement, these partial changes solidify Bitlayer’s foundation without altering the project's core architectural integrity.

Security score

Our review of the changes introduced in the Bitlayer project's codebase found no security issues, resulting in a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's blockchain protocol yields an overall score of **9.7**. This score reflects the combined evaluation of documentation, code quality, architecture quality, and security aspects of the project.

Findings

Vulnerability Details

C

Observation Details

F-2024-1631 - Inconsistency in Staking ABI between Repositories - Info

Description:	There is a discrepancy in Staking ABI definitions (in testdata) between its **bitlayer-contracts** and **bitlayer-l2** repositories. The `changeFoundationPool` method is absent in the **bitlayer-l2** ABI, and `valInfos` is missing in the **bitlayer-contracts** ABI. This misalignment leads to variations in bytecode, affecting the genesis file and the output of the mkalloc tool, potentially causing runtime issues.
Status:	Fixed

Recommendations

Recommendation:	Synchronize ABI Definitions: Update the **bitlayer-l2** repository to include the `changeFoundationPool` method and add `valInfos` to the **bitlayer-contracts** repository. Update Genesis File and Tool Outputs: Reflect the synchronized ABI definitions in the genesis file bytecode and verify correct mkalloc tool outputs. Implement CI Checks: Introduce CI checks to automatically verify ABI consistency across repositories, preventing future discrepancies (optional)

Evidences

Test data

Location:	https://github.com/bitlayer-org/bitlayer-l2/blob/master/consensus/merlion/testdata/staking_abi.json

F-2024-1660 - Suboptimal Test Coverage in Bitlayer Project - Info

Description:

Merlion module exhibits a test coverage of 83.3% across files, with statement coverage varying from 0% to 100%, as specified in the table below.

Specific concern raised with the `TestMerlion` suite, which was broken in the audit commit, but later fixed during remediation period, and it significantly improved coverage.

Moreover, additional modified components by the Bitlayer engineering team, such as `ActionLogger`, `TraceActionByBlockHash`, `TraceActionByBlockNumber`, and `TraceActionByTxHash`, lack test coverage entirely.

While the coverage is commendable in some areas, these gaps pose a risk for undetected bugs and reduce the confidence in the stability and security of the codebase.

```
| Component                 | Statements Coverage |
|--------------------------|---------------------|
| merlion                   | 51.3%               |
| ├ systemcontract          |                     |
| |   ├ contract.go         | 67.6%               |
| |   └ contract_caller.go  | 63.6%               |
| ├ testdata                |                     |
| |   ├ api.go              | 0%                  |
| |   ├ interactive.go.     | 66.8%               |
| |   ├ merlion.go          | 48.6%               |
| |   └ snapshot.go         | 77.1%               |
```

Assets:

- consensus/merlion/systemcontract/contract.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/systemcontract/contract_caller.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/systemcontract/contract_test.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/api.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/interactive.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/merlion.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/merlion_test.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/snapshot.go [https://github.com/bitlayer-org/bitlayer-l2]
- consensus/merlion/snapshot_test.go [https://github.com/bitlayer-org/bitlayer-l2]

Status:	Mitigated

Recommendations

Recommendation:	Implement minimum test cases for `ActionLogger`, `TraceActionByBlockHash`, `TraceActionByBlockNumber`, and `TraceActionByTxHash` to ensure that all new or modified code paths are tested. Improve Merlion test suite: Address the issues within the `TestMerlion` suite to utilize all existing test cases effectively. This could substantially raise the overall test coverage percentage and ensure that more code paths are verified. Remediation notes: Merlion test suite was improved with additional test cases.

Appendix 1. Severity Definitions

Severity	Description
Critical	Vulnerabilities that can lead to a complete breakdown of the blockchain network's security, privacy, integrity, or availability fall under this category. They can disrupt the consensus mechanism, enabling a malicious entity to take control of the majority of nodes or facilitate 51% attacks. In addition, issues that could lead to widespread crashing of nodes, leading to a complete breakdown or significant halt of the network, are also considered critical along with issues that can lead to a massive theft of assets. Immediate attention and mitigation are required.
High	High severity vulnerabilities are those that do not immediately risk the complete security or integrity of the network but can cause substantial harm. These are issues that could cause the crashing of several nodes, leading to temporary disruption of the network, or could manipulate the consensus mechanism to a certain extent, but not enough to execute a 51% attack. Partial breaches of privacy, unauthorized but limited access to sensitive information, and affecting the reliable execution of smart contracts also fall under this category.
Medium	Medium severity vulnerabilities could negatively affect the blockchain protocol but are usually not capable of causing catastrophic damage. These could include vulnerabilities that allow minor breaches of user privacy, can slow down transaction processing, or can lead to relatively small financial losses. It may be possible to exploit these vulnerabilities under specific circumstances, or they may require a high level of access to exploit effectively.
Low	Low severity vulnerabilities are minor flaws in the blockchain protocol that might not have a direct impact on security but could cause minor inefficiencies in transaction processing or slight delays in block propagation. They might include vulnerabilities that allow attackers to cause nuisance-level disruptions or are only exploitable under extremely rare and specific conditions. These vulnerabilities should be corrected but do not represent an immediate threat to the system.

Appendix 2. Scope

The scope of the project includes the following components from the provided repository:

Scope Details	
Repository	https://github.com/bitlayer-org/bitlayer-l2
Commit	44ae502bb8c1032d06befd4166ae773a12a35c37
Whitepaper	N/A
Requirements	https://drive.google.com/file/d/12tnOjrU9ThWa2jJPqXir1lXUjQ_nmgId/view
Technical Requirements	https://drive.google.com/file/d/12tnOjrU9ThWa2jJPqXir1lXUjQ_nmgId/view

Components in Scope
https://github.com/bitlayer-org/bitlayer-l2/tree/master/consensus/merlion
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/gen_genesis.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/gen_genesis_account.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/gen_genesis_init.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/gen_genesis_validator_info.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/genesis.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/genesis_init.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/genesis_alloc.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/genesis_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/mkalloc.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state_processor.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state_transition.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state/database.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state/state_object.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state/statedb.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/state/trie_prefetcher.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/types.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/vm/logger_action.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/vm/analysis.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/vm/contract.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/vm/interpreter.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/vm/jump_table.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/blockchain.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/blockchain_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/block_validator.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/rawdb/database.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/rawdb/ancient_scheme.go

Components in Scope
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/rawdb/internal_tx.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/core/rawdb/schema.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/eth/tracers/api.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/eth/backend.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/eth/state_accessor.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/eth/sync_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/ethclient/simulated/backend_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/params/bootnodes.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/params/config.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/params/version.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/tests/block_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/trie/secure_trie.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/clef/README.md
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/clef/main.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/devp2p/README.md
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/devp2p/discv4cmd.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/devp2p/nodesetcmd.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/geth/genesis_test.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/geth/main.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/cmd/utils/flags.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/accounts/external/backend.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/accounts/accounts.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/common/gopool/gopool.go
https://github.com/bitlayer-org/bitlayer-l2/blob/master/contracts/system/abi.go