Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.02.19, the SlowMist security team received the bitlayer team's security audit application for bitlayer-l2, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

In black box testing and gray box testing, we use methods such as fuzz testing and script testing to test the robustness of the interface or the stability of the components by feeding random data or constructing data with a specific structure, and to mine some boundaries Abnormal performance of the system under conditions such as bugs or abnormal performance. In white box testing, we use methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to analyze the object definition and logic implementation of the code to ensure that the code has the key components of the key logic. Realize no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

NO.	Audit Items	Result
1	Design Logic Audit	Some Risks
2	Others	Some Risks
3	State Consistency Audit	Passed
4	Failure Rollback Audit	Passed
5	Unit Test Audit	Passed
6	Integer Overflow Audit	Passed
7	Parameter Verification Audit	Passed
8	Error Unhandle Audit	Passed

NO.	Audit Items	Result
9	Boundary Check Audit	Passed
10	SAST	Passed

3 Project Overview

3.1 Project Introduction

BitLayer layer 2 EVM client.

3.2 Coverage

Target Code and Revision:

https://github.com/bitlayer-org/bitlayer-l2/tree/master/consensus/merlion

Commit: 44ae502bb8c1032d06befd4166ae773a12a35c37

Module to be audit：consensus/merlion

3.3 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Redundant code	Others	Suggestion	Confirmed
N2	The address of the EngineCaller may have a private key.	Others	Information	Ignored
N3	Validator should allow time deviation	Design Logic Audit	Suggestion	Acknowledged
N4	Errors unhandled	Design Logic Audit	Low	Acknowledged
N5	Ethereum hardfork codes should not be	Design Logic Audit	Suggestion	Acknowledged

NO	Title	Category	Level	Status
	retained			

4 Findings

4.1 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

4.2 Vulnerability Summary

[N1] [Suggestion] Redundant code

Category: Others

Content

- bitlayer-l2/consensus/merlion/api.go

```
func (api *API) Status() (*status, error) {
    //...
    diff += h.Difficulty.Uint64() //SlowMist//
    //...
}
```

Solution

Remove useless codes

Status

Confirmed

[N2] [Information] The address of the EngineCaller may have a private key.

Category: Others

Content

- bitlayer-l2/consensus/merlion/systemcontract/contract_caller.go

```
var (
        EngineCaller =
common.HexToAddress("0x00000000000004269746c61796572456e67696e65")
)
```

EngineCaller is a system account which can invoke system functions without private key, because it is coded in low level blockchain layer. However, `0x00000000000004269746c61796572456e67696e65` looks no special enough to be an system account, it may be produce by vanity tools, which may have a private key. If this address have a private key, it means the system functions can be invoke by EOA, which is out of control by validators.

Solution

Change this address to an more special address, such as `0x0000000000000000000000000000000000006e65`

Status

Ignored; This hex address represents ASCII code of "BitlayerEngine"

[N3] [Suggestion] Validator should allow time deviation

Category: Design Logic Audit

Content

- bitlayer-l2/consensus/merlion/merlion.go

```
func (c *Merlion) verifyHeader(chain consensus.ChainHeaderReader, header
*types.Header, parents []*types.Header) error {
    //...
        if header.Time > uint64(time.Now().Unix()) {
                return consensus.ErrFutureBlock
        }
    //...
```

The system time may not be accurate in different servers, in this situation, the blocks produced by such validators will be rejected by other validators easily.

Solution

Allow one second deviation.

Status

Acknowledged

[N4] [Low] Errors unhandled

Category: Design Logic Audit

Content

- bitlayer-l2/consensus/merlion/merlion.go

```
func New(chainConfig *params.ChainConfig, db ethdb.Database) (*Merlion, error) {
        //...
        recents, _ := lru.NewARC(inmemorySnapshots)
        signatures, _ := lru.NewARC(inmemorySignatures)
        eventCheckRules, _ := lru.New(inmemoryAccesslist)
    //...
}
```

Unhandled errors often lead to unforeseen logic errors.

Solution

Handle errors instead of using `_` to ignore them.

Status

Acknowledged

[N5] [Suggestion] Ethereum hardfork codes should not be retained

Category: Design Logic Audit

Content

- bitlayer-l2/consensus/merlion/merlion.go

```
func (c *Merlion) verifyCascadingFields(chain consensus.ChainHeaderReader, header
*types.Header, parents []*types.Header) error {
    //...
        if !chain.Config().IsLondon(header.Number) { //SlowMist//
                // Verify BaseFee not present before EIP-1559 fork.
                if header.BaseFee != nil {
                        return fmt.Errorf("invalid baseFee before fork: have %d, want
<nil>", header.BaseFee)
                }
                if err := misc.VerifyGaslimit(parent.GasLimit, header.GasLimit); err
!= nil {
                        return err
                }
```

```
        } else if err := eip1559.VerifyEIP1559Header(chain.Config(), parent, header);
err != nil {
                // Verify the header's EIP-1559 attributes.
                return err
        }

        // All basic checks passed, verify the seal and return
        return c.verifySeal(chain, header, parents)
}
```

Ethereum hardfork codes should not be retained.

Solution

Rewrite it.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002402270001	SlowMist Security Team	2024.02.19 - 2024.02.27	Low Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 low risk, 3 suggestion vulnerabilities. And 1 suggestion vulnerabilities were confirmed and being fixed; 1 vulnerabilities were ignored;

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist