Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.08.01, the SlowMist security team received the community's security audit application for go-ethereum, 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 "black, grey box lead, white box assists" 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.

2 Audit Methodology

The security audit process of SlowMist security team for the chain includes two steps:

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

Manual audit of the codes for security issues. The codes 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 chain:

NO.	Audit Items	Result
1	SAST	Some Risks
2	Cryptographic Security Audit	Passed
3	Account and Transaction Security Audit	Some Risks
4	RPC Security Audit	Passed
5	P2P Security Audit	Passed
6	Consensus Security Audit	Passed

3 Project Overview

3.1 Project Introduction

Go implementation of the Ethereum protocol.

3.2 Coverage

Target Code and Revision:

https://github.com/ethereum/go-ethereum

v1.13.15

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	False top-up risk of exchanges	Account and Transaction Security Audit	Information	Acknowledged
N2	Errors unhandled	SAST	Low	Acknowledged
N3	Implicit memory aliasing in for loop.	SAST	Low	Acknowledged
N4	Potential slowloris attack	SAST	Suggestion	Acknowledged

4 Findings

4.1 Vulnerability Summary

[N1] [Information] False top-up risk of exchanges

Category: Account and Transaction Security Audit

Content

In a complex transaction, even if an internal cross-contract call fails, the entire transaction may still be marked as successful. This feature may lead to potential security risks like 'false top-up'

Solution

Exchanges need to pay attention to the risk of fake top-up and strictly check deposit transactions.

Status

Acknowledged

[N2] [Low] Errors unhandled

Category: SAST

Content

The following code does not process the returned error information during the calling process, which may cause the program to not terminate in time when an error occurs, resulting in a logical error.

Provide a separate document for the discovered part for reference.

Solution

Check the return value of a function call.

Status

Acknowledged

[N3] [Low] Implicit memory aliasing in for loop.

Category: SAST

Content

- p2p/simulations/http.go:437

```
   436:                 for _, node := range snap.Nodes {
 > 437:                         event := NewEvent(&node.Node)
   438:                         if err := writeEvent(event); err != nil {
```

- internal/ethapi/transaction_args.go:353

```
   352:         for i, c := range args.Commitments {
 > 353:                 hashes[i] = kzg4844.CalcBlobHashV1(hasher, &c)
   354:         }
```

- cmd/geth/accountcmd.go:214

```
   213:                 for _, account := range wallet.Accounts() {
 > 214:                         fmt.Printf("Account #%d: {%x} %s\n", index,
account.Address, &account.URL)
   215:                         index++
```

Solution

By creating a copy of the variable, you ensure that each iteration receives an independent copy of the current variable, thus avoiding implicit memory aliasing problems.

Status

Acknowledged

[N4] [Suggestion] Potential slowloris attack

Category: SAST

Content

Potential Slowloris Attack because ReadHeaderTimeout is not configured in the http.Server

- node/rpcstack.go:141

```
  140:         // Initialize the server.
> 141:         h.server = &http.Server{Handler: h}
  142:         if h.timeouts != (rpc.HTTPTimeouts{}) {
```

Solution

Configure ReadHeaderTimeout.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002508010001	SlowMist Security Team	2025.08.01 - 2025.08.01	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 2 low risk, 1 suggestion vulnerabilities.

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