Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.03.04, the SlowMist security team received the morph-l2 team's security audit application for morph-l2/tendermint, 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	Some Risks

3 Project Overview

3.1 Project Introduction

Morph is an innovative force reshaping the consumer blockchain landscape for practical, everyday use. At the core of Morph is a revolutionary approach to Ethereum Layer 2 scalability, harnessing the power of advanced rollup technology.

3.2 Coverage

Target Code and Revision:

https://github.com/morph-l2/tendermint/tree/layer2

Initial audit commit:f7cf181d017a9049cfc13cfb372724a86c8d1807

Final audit commit:a0e8684be97042e7b9d543b6d67208e29845fb46

```
blocksync/reactor.go
blssignatures/bls_signatures.go
blssignatures/file.go
blssignatures/util/colors.go
blssignatures/util/pseudorandom.go
blssignatures/util/testhelpers.go
cmd/tendermint/commands/init.go
cmd/tendermint/commands/rewind.go
cmd/tendermint/commands/run_node.go
config/config.go
config/toml.go
consensus/batch.go
consensus/reactor.go
consensus/replay.go
consensus/replay_file.go
consensus/state.go
consensus/types/height_vote_set.go
```

```
consensus/wal_generator.go
l2node/l2node.go
l2node/mock.go
l2node/notifier.go
node/node.go
proto/tendermint/abci/types.proto
proto/tendermint/mempool/types.proto
proto/tendermint/rpc/grpc/types.proto
proto/tendermint/types/canonical.proto
proto/tendermint/types/params.proto
proto/tendermint/types/types.proto
proxy/app_conn.go
proxy/multi_app_conn.go
state/execution.go
state/rewind.go
state/state.go
state/tx_filter.go
statesync/reactor.go
statesync/stateprovider.go
store/store.go
types/block.go
types/block_meta.go
types/canonical.go
types/params.go
types/test_util.go
types/validator_set.go
types/vote.go
types/vote_set.go
```

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Use of codebases with security risks	SAST	Medium	Fixed
N2	Error Handling Recommendations for BLS Signature Verification	Others	Suggestion	Fixed
N3	Redundant codes	Others	Suggestion	Acknowledged
N4	Direct comparison of errors	Others	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N5	Random number safety recommendations.	Design Logic Audit	Suggestion	Acknowledged

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] [Medium] Use of codebases with security risks

Category: SAST

Content

- https://github.com/morph-l2/tendermint

```
Dependency go:golang.org/x/crypto:v0.6.0 Vulnerable, security version v0.21.0
https://devhub.checkmarx.com/cve-details/CVE-2023-48795
https://devhub.checkmarx.com/cve-details/CVE-2023-42818

Dependency go:golang.org/x/net:v0.6.0 Vulnerable, security version v0.22.0
https://devhub.checkmarx.com/cve-details/CVE-2022-41723
https://devhub.checkmarx.com/cve-details/CVE-2023-44487
https://devhub.checkmarx.com/cve-details/CVE-2023-39325

Dependencies go:google.golang.org/protobuf:v1.28.1 Vulnerable, security version
v1.33.0
https://devhub.checkmarx.com/cve-details/CVE-2024-24786

Dependencies go:github.com/btcsuite/btcd:v0.23.0 Vulnerable, security version v0.22.3
https://devhub.checkmarx.com/cve-details/CVE-2022-39389/
https://devhub.checkmarx.com/cve-details/CVE-2022-44797/

Dependencies go:github.com/containerd/containerd:v1.6.6 Vulnerable, secure version
v1.7.14
https://devhub.checkmarx.com/cve-details/CVE-2022-23471/
https://devhub.checkmarx.com/cve-details/CVE-2023-25153/
```

```
https://devhub.checkmarx.com/cve-details/CVE-2023-25173/
https://devhub.checkmarx.com/cve-details/Cx68162182-3f7b

Dependencies go:github.com/docker/distribution:v2.8.1+incompatible Vulnerable, secure
version v2.8.2+incompatible
https://devhub.checkmarx.com/cve-details/CVE-2023-2253

Dependencies go:github.com/moby/buildkit:v0.10.3 Vulnerable, safe version v0.13.0
https://devhub.checkmarx.com/cve-details/CVE-2024-23651
https://devhub.checkmarx.com/cve-details/CVE-2024-23650

Dependencies go:github.com/opencontainers/runc:v1.1.3 Vulnerable, secure version
v1.1.12
https://devhub.checkmarx.com/cve-details/CVE-2023-27561
https://devhub.checkmarx.com/cve-details/CVE-2023-28642
https://devhub.checkmarx.com/cve-details/CVE-2023-25809
https://devhub.checkmarx.com/cve-details/CVE-2024-21626
```

Solution

Update to secure version.

Status

Fixed; Fixed in https://github.com/morph-l2/tendermint/pull/9

[N2] [Suggestion] Error Handling Recommendations for BLS Signature Verification

Category: Others

Content

- consensus/state.go

At line:2234, it is recommended that an error message log be added or an error message returned for subsequent exception finding.

```
func (cs *State) addVote(vote *types.Vote, peerID p2p.ID, replay bool) (added bool,
err error) {
    ...
        if vote.Height+1 == cs.Height && vote.Type == tmproto.PrecommitType {
                if cs.Step != cstypes.RoundStepNewHeight {
                        // Late precommit at prior height is ignored
                        cs.Logger.Debug("precommit vote came in after commit timeout
and has been ignored", "vote", vote)
                        return
                }
                // check the bls signature before adding it to LastCommit
                if len(vote.BlockID.BatchHash) > 0 {
                        if len(vote.BLSSignature) == 0 {
                                return //SLOWMIST// Exception lacks error message
logging or returns an error message
                        }
                        pubKey :=
cs.Validators.Validators[vote.ValidatorIndex].PubKey.Bytes()
                        valid, err := cs.l2Node.VerifySignature(pubKey,
vote.BlockID.BatchHash, vote.BLSSignature)
                        if err != nil {
                                cs.Logger.Error("failed to verify bls signature",
"err", err)
                                return false, nil
                        }
                        if !valid {
                                return false, fmt.Errorf("%v. sig: %x, batchHash:
%x, tmKey: %x", ErrBLSSignatureInalvid, vote.BLSSignature, vote.BlockID.BatchHash,
pubKey)
                        }
                } else if len(vote.BLSSignature) > 0 {
                        return false, errors.New("should not have bls signature while
the batchHash is empty")
                }
    ...
}
```

Solution

Returns the corresponding error message.

Status

Fixed; Fixed in https://github.com/morph-l2/tendermint/pull/11

[N3] [Suggestion] Redundant codes

Category: Others

Content

- blssignatures/bls_signatures.go

`GeneratePrivKeyString` function not used

`PublicKeyFromBytes` function is not used.

`SignatureFromBytes` function not used.

- state/execution.go

The `fireEvents` function is not used.

Solution

If it is certain that these will not be used, then they can be removed.

Status

Acknowledged

[N4] [Suggestion] Direct comparison of errors

Category: Others

Content

- consensus/state.go

It is recommended to use errors.As instead of err.(*types.ErrVoteConflictingVotes) to force type conversion.

```
func (cs *State) tryAddVote(vote *types.Vote, peerID p2p.ID, replay bool) (bool,
error) {
        added, err := cs.addVote(vote, peerID, replay)
        if err != nil {
                // If the vote height is off, we'll just ignore it,
                // But if it's a conflicting sig, add it to the cs.evpool.
                // If it's otherwise invalid, punish peer.
                //nolint: gocritic
                if voteErr, ok := err.(*types.ErrVoteConflictingVotes); ok {
//SLOWMIST//
                        if cs.privValidatorPubKey == nil {
                                return false, errPubKeyIsNotSet
                        }
        ...
}
```

Solution

Use errors.Is

Status

Acknowledged; Only one place returned types.ErrVoteConflictingVotes error.So it can be ignored.

[N5] [Suggestion] Random number safety recommendations.

Category: Design Logic Audit

Content

- blssignatures/util/pseudorandom.go

Uses an insecure library of random numbers.

```
import (
        "math/rand" //SLOWMIST//
        "testing"

        "github.com/scroll-tech/go-ethereum/common"
)
```

Solution

Use of "crypto/rand"

Status

Acknowledged; This confirms that you need to use math/rand to generate pseudorandom.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002403310001	SlowMist Security Team	2024.03.04 - 2024.03.31	Passed

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 medium risk, 4 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