Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.06.06, the SlowMist security team received the TON team's security audit application for TON, 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.

Level	Description
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	Others	Passed
2	State Consistency Audit	Some Risks
3	Failure Rollback Audit	Passed

NO.	Audit Items	Result
4	Unit Test Audit	Some Risks
5	Integer Overflow Audit	Some Risks
6	Parameter Verification Audit	Passed
7	Error Unhandle Audit	Passed
8	Boundary Check Audit	Some Risks
9	SAST	Passed

3 Project Overview

3.1 Project Introduction

TON is the next gen network to unite all blockchains and the existing Internet.

3.2 Coverage

Target Code and Revision:

https://github.com/newton-blockchain/ton/tree/master/catchain

Commit: ae5c0720143e231c32c3d2034cfe4e533a16d969

https://github.com/newton-blockchain/ton/tree/master/validator-session

Commit: ae5c0720143e231c32c3d2034cfe4e533a16d969

https://github.com/newton-blockchain/ton/tree/master/validator-engine

Commit: ae5c0720143e231c32c3d2034cfe4e533a16d969

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Lack of unit testing	Unit Test Audit	Low	Confirming
N2	Overflow risks	Integer Overflow Audit	Suggestion	Confirming
N3	Divisor not checked	Boundary Check Audit	Suggestion	Confirming
N4	Using system time	State Consistency Audit	Low	Confirming

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] [Low] Lack of unit testing

Category: Unit Test Audit

Content

- catchain/catchain-block.cpp
- catchain/catchain-block.hpp
- catchain/catchain-received-block.cpp
- catchain/catchain-received-block.h
- catchain/catchain-received-block.hpp
- catchain/catchain-receiver-interface.h

- catchain/catchain-receiver-source.cpp
- catchain/catchain-receiver-source.h
- catchain/catchain-receiver-source.hpp
- catchain/catchain-receiver.cpp
- catchain/catchain-receiver.h
- catchain/catchain-receiver.hpp
- catchain/catchain-types.h
- catchain/catchain.cpp
- catchain/catchain.h
- catchain/catchain.hpp
- validator-engine/validator-engine.hpp
- validator-engine/validator-engine.cpp
- validator-session/persistent-vector.cpp
- validator-session/persistent-vector.h
- validator-session/validator-session-description.cpp
- validator-session/validator-session-description.h
- validator-session/validator-session-description.hpp
- validator-session/validator-session-state.cpp
- validator-session/validator-session-state.h
- validator-session/validator-session-types.h
- validator-session/validator-session.cpp
- validator-session/validator-session.h
- validator-session/validator-session.hpp

Unit test coverage is too low (~0%).

Unit testing is critical to ensure that a product or functionality will perform even after a change.

Solution

Increase the coverage of unit testing

Status

Confirming

[N2] [Suggestion] Overflow risks

Category: Integer Overflow Audit

Content

Use `+-*/` for arithmetic operations, risk of numerical overflow.

Solution

Using safemath function for arithmetic operations.

Status

Confirming

[N3] [Suggestion] Divisor not checked

Category: Boundary Check Audit

Content

- validator-session/persistent-vector.h

```
td::uint32 size() const {
  return static_cast<td::uint32>(data_size_ / sizeof(T));
}
td::uint32 size() const {
  return static_cast<td::int32>(data_size_ / sizeof(T));
}

* validator-session/validator-session-description.cpp

```c++
//#L176
```
```

```
return static_cast<void *>(pdata_perm_[s / pdata_perm_size_] + (s %
pdata_perm_size_));
```

- validator-session/validator-session-description.hpp

```
  td::uint32 get_attempt_seqno(td::uint64 ts) const override {
    return get_unixtime(ts) / opts_.round_attempt_duration;
  }
```

- validator-session/validator-session-description.hpp

```
double lambda = 10.0 / description().get_total_nodes();
```

Failure to check if the divisor is zero.

Solution

check if the divisor is zero.

Status

Confirming

[N4] [Low] Using system time

Category: State Consistency Audit

Content

- validator-engine/validator-engine.cpp

```
//#L1675
td::uint32 ts = static_cast<td::uint32>(td::Clocks::system());

//#L2318
auto obj = ton::create_tl_object<ton::ton_api::engine_validator_time>
(static_cast<td::int32>(td::Clocks::system()));
```

Using system time in the blockchain may cause inconsistent state among nodes.

Solution

Detecting the difference between system time and block time.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206270001	SlowMist Security Team	2022.06.06 - 2022.06.27	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 2 low risk, 2 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