|certain| × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

November 1, 2024

Abstract 2
Disclaimer 2
Summary 2
General recommendations 2
Project overview 3
Project description 3
Token details #1 3
Codebase update #1 3
Token details #2 3
Codebase update #2 4
Audit process 5
Manual analysis 6
Critical issues 6
Medium severity issues 7
M01. Project owner role (commented) 7
M02. Inconsistency of the TIP-712 standard (fixed) 7
M03. Failed tests (fixed) 7
Low severity issues 8

ABSTRACT

In this report, we consider the security of smart contracts of Certain.fi project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Certain.fi smart contracts. We described the audit process in the section below.

The audit showed several issues of medium severity: Project owner role, Inconsistency of the TIP-712 standard, Failed tests.

After the initial audit, the codebase was updated. The developers fixed the following issues of medium-severity: Inconsistency of the TIP-712 standard and Failed tests. And they commented on the Project owner role issue of medium-severity.

After the first recheck, the codebase was updated. The developers added new functionality to smart contract. No new issues were found.

Some functionality is executed off-chain or manually, such as calculating and sending rewards. Our scope was limited, and we did not audit the off-chain part of the code. This may result in that part of the project not conforming to the description.

GENERAL RECOMMENDATIONS

We do not have any further recomendations.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Certain.fi project on a private GitHub repository, commit a03f6e10af88bf08f2a21e715468a09d7e2df26d.

The scope of the audit included everything.

The documentation for the project included the developer's comments.

The tests did not pass. The code coverage was not measured.

The total LOC of audited sources is 164.

Token details #1

Name:	Delta Neutral Coin
Symbol:	DNC
Decimals:	6
Total Supply:	Unlimited

Codebase update #1

After the initial audit, the codebase was updated, and we were provided with commit 78570f0d5e3c0f347033b8a96f10ded35594adeb.

The developers fixed or provided comments on all medium-severity issues. 21 tests out of 21 passed successfully. The code coverage was 100%.

Token details #2

Name:	Delta Neutral Token V1
Symbol:	DNT
Decimals:	6
Total Supply:	Unlimited

Codebase update #2

After the first recheck, the codebase was updated, and we were provided with a commit on a new repository 5a42215a705d90c7921b156776a2e282bf5b5a16.

The developers added new functionality on this commit, to seize and burn blacklisted user's funds. Moreover, the developers implemented CI to run tests and calculate coverage.

22 tests out of 22 passed successfully. The code coverage was 100%.

AUDIT PROCESS

We started the audit on May 13 and finished on May 14, 2024.

We inspected the materials provided for the audit and started the work.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Whether the code corresponds to the TIP-712;
- Whether the code follows the TRC-20 and ERC-20 standards;
- The owner's level of authority.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

We made the recheck on May 20-21, 2024. We checked the updated code, re-ran the tests, and calculated the code coverage. Finally, we updated the report.

We received the updated code on October 30, 2024. We checked the code, re-ran the tests, and calculated the code coverage. Finally, we updated the report.

MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Project owner role (commented)

In the current implementation, the token functionality depends heavily on the owner's role. If the owner's private keys become compromised:

- An attacker can add any account to the blacklist, and they will not be able to transfer tokens and receive them;
- An attacker can update the implementation logic at any time.

Although this behavior is implied according to the developers' comments, we recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> `Comment from the developers:`
>
> **The first revision of the contract assumes a high level of centralization, which we plan to lower further. As one of the ways to protect against compromise of private keys, we will use (including, but not limited to) a multi-signature admin wallet.**

M02. Inconsistency of the TIP-712 standard (fixed)

According to the TIP-712, the TRON 712 standard has differences from the EIP-712 standard. It leads to the following mismatch in the **DeltaNeutralCoinTron** contract.

The `chainId` at line 34 is constant and equal to 0. However, it should be calculated dynamically, as described in the standard (`block.chainid & 0xffffffff`).

> **The issue has been fixed and is not present in the latest version of the code.**

M03. Failed tests (fixed)

The tests do not pass, and the code coverage is not measured. We always note the availability of tests and code coverage as it helps to avoid additional bugs or typos.

We highly recommend making sure that all tests pass and that the code coverage is sufficient.

> **All the tests passed.**

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

The audit showed no issues of low severity.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

November 1, 2024