Intentable's Smart Transaction technology powered by the Kirobo FCT Platform

FCT Extensions Security Analysis

by Pessimistic

This report is public

October 2, 2024

Abstract

In this report, we consider the security of smart contracts of Intentable 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 the FCT Extensions submodule of Intentable smart contracts. We described the audit process in the section below.

The report has designations like C01, M05, L10, N08. The letter represents severity, and the number represents the issue number.

This is a condensed version of the report; a more detailed one can be found at the following links:

- Report #1. The report includes the M04 issue, which was addressed and was relevant to the whole project.
- Report #2. The fixed issues:
    - L09, L10, L11, L14, which were relevant to the current submodule;
    - L04, which was relevant for the whole project.
- Report #3. The report includes:
    - M01, which was addressed and was relevant to the whole project;
    - L03, which was relevant to the current submodule.

The report includes the N01 (N09 in the Report #2) issue with the `commented` status. All the tests passed. The code coverage is sufficient.

After the last recheck, which was fully described in Report #3 (see the Codebase update #8) and copied to the Project description, the codebase was updated again. The number of tests and the code coverage increased. The developers fixed note.

According to our recommendations, the developers split the long function in the **FCT_BatchMultiSig** contract, improving the code readability. However, it is still important to note that the project and its architecture are complex, though we realize that it is difficult or impossible to implement simply.

It is crucial to study the three reports above to get a full picture of the security of smart contracts.

General recommendations

We recommend implementing CI to run tests, calculate code coverage, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with Intentable project on a private GitHub repository, commit cf31b27dd9e44e016ced8e3060a487e7e1e2d5cc.

It is a condensed version of the report. More detailed information about codebase updates can be found at the following links:

- Report #1;
- Report #2;
- Report #3.

The scope included:

- Decimals.sol;
- FCT_Ext_SecureStorage.sol;
- FCT_Ext_SignersValidator.sol;
- FCT_Ext_TokensSafeMath.sol;
- FCT_Ext_TokensValidator.sol.

The documentation for the project included https://kirobo.gitbook.io/fct-developers-guide/.

The total LOC of the audited scope is 685.

All 592 tests passed. The code coverage of the project was 84.64%.

Codebase update #1

For the recheck, we were provided with Intentable project on a private GitHub repository, commit 476e33239267aa680f46d7aec3312669fea7f9eb.

The developers provided the test results and the code coverage. All 634 tests passed, the scope's code coverage was 95.76%, and the FCT platforms' code coverage was 95.68%. The developers fixed note.

Audit process

We started to check the FCT platform around two years ago (2022) and created four extensive reports. The developers asked us to split these reports into seven submodules. Each submodule has:

- The last common commit for all reports;
- The same results of the tests and the code coverage for the whole project;
- The individual scope;
- Links to the corresponding previous reports;
- The list of fixed issues, e.g., M01, M04, L03 (see the description in the Summary), etc.;
- The descriptions of unfixed or commented issues that are still actual;
- The findings relevant to the whole project.

Issue descriptions copied from previous reports can have different contract names and lines, as they were checked again and updated due to the last commit in the Project description.

We started auditing the FCT platforms in 2022. During this time, we made three FCT reports and one report for the Smart Wallet part.

Each report is a continuation of the previous one. We made the split in the process to avoid one infinitely extensive report. The chronology of the reports:

- Smart Wallet report - finished on August 15, 2022;
- Report #1 - finished on November 17, 2022;
- Report #2 - finished on July 26, 2023, and last updated on January 16, 2024;
- Report #3 - finished on June 26, 2024.

See the previous reports for a more detailed description of the issues and process. The following rechecks related to a specific module will only appear in the corresponding report.

We made the recheck #1 on July 3, 2024. We checked the fix for N01, asked the developer for test results as we could not run them and recalculated 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.

The audit showed no issues of medium severity.

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.

Notes

N01. Collisions (fixed)

If the `keccak256` hash is calculated from the result of `abi.encodePacked(a, b)`, the collisions are possible since both the following cases will give the same hash: a=AAA, b=BB and a=AA, b=ABB.

The `write_uint256`, `write_bytes32` and `write_bytes` functions store data to the `private_uint256`, `private_bytes32` and `private_bytes` mappings in the **FCT_Ext_SecureStorage** contract. The data mappings can be changed in the `fctCall` function via the FCT running.

And if `abi.encodePaked(fctMsgHash, key)` == `abi.encodePacked(msg.sender, key)`, then the data can be replaced via the FCT. Theoretically, it is possible since `fctMsgHash` is a hash of FCT data and `key` is a free parameter.

> *Comment from the developers: In order to reduce the chance of collision to the level of the network itself we use the following:*
>
> 1. *There is a new function in FCT_Controller that verifies a given messageHash equals the current running FCT messageHash.*
> 2. *This function is used in SecureStorage to make sure that the messageHash is genuine.*
> 3. *Because the FCT holds also a random number (part of the message meta), theoretically the FCT messagehash can be manipulated to match an existing address. In order to eliminate this possibility there is a a constant hash, at the beginning of the slot data to be hashed, that indicates this slot belongs to a specific FCT (rather than private slots that belongs to Runners).*
> 4. *FCT slots: keccack256(<fct_hash(bytes32)> <fct_messageHash(bytes32)> <key(bytes32)>).*
> 5. *Runner slots: keccack256(<runner_hash(bytes32)><msg.sender(address)> <key(bytes32)>).*
> 6. *FCT_Controller also makes sure that only one FCT messageHash can be registered in the system (assuming purge flag is false).*

> *Pessimistic's comment:*
> *During the rechecks, the **SecureStorage** contract was renamed **FCT_Ext_SecureStorage**.*

The issues have been fixed and are not present in the latest version of the code.

This analysis was performed by Pessimistic:

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

October 2, 2024