

Codatta Ton Audit Report

Mon Nov 25 2024



contact@bitslab.xyz



https://twitter.com/tonbit_



Codatta Ton Audit Report

1 Executive Summary

1.1 Project Information

Description	A staking project deployed on Ton, allows users to stake Ton to earn the bonus.
Type	DeFi
Auditors	TonBit
Timeline	Tue Nov 19 2024 - Wed Nov 20 2024
Languages	FunC
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/codatta/vault-ton-contract
Commits	e19322552c9c37dd3f97b0881b76a4520ffb160dca46027faa6d1fe856c5e28e77c2bc026ab7773c

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
STA	contracts/staking.fc	7979e117a6975791fdb34377e9bff 276214eb523
UTI	contracts/utls.fc	22551c59c5aa954e879176c803f27 368f684b254

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	1	4
Informational	0	0	0
Minor	1	1	0
Medium	3	0	3
Major	1	0	1
Critical	0	0	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by [Codatta](#) to identify any potential issues and vulnerabilities in the source code of the [Codatta Ton](#) smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 5 issues of varying severity, listed below.

ID	Title	Severity	Status
STA-1	Centralization Risk	Major	Acknowledged
STA-2	The Signature Does Not Include slashed	Medium	Acknowledged
STA-3	Division Before Multiplication Result in Precision Loss	Medium	Acknowledged
STA-4	Transfer Boundary Value	Medium	Acknowledged
UTI-1	Unused Definitions	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the [Codatta Ton](#) Smart Contract :

Owner

- The owner can call the `admin_recycle` function to withdraw the ton of the contract.

User

- Users can call the `staking` function to stake their ton.
- Users can call the `withdraw` function to get back their ton and bonus.

4 Findings

STA-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/staking.fc#142

Descriptions:

Centralization risk was identified in the smart contract. The admin can:

- The `admin_address` address can call `admin_recycle` to withdraw any amount of balance of the protocol.

Suggestion:

It is recommended to take measures to reduce the risk of centralization.

STA-2 The Signature Does Not Include slashed

Severity: Medium

Status: Acknowledged

Code Location:

contracts/staking.fc#115

Descriptions:

In the `withdraw` process, the `slashed` field is not involved in the calculation of the signature data, resulting in the user being able to control the value of the slashed field and thus obtaining an additional `bonus_amount`.

```
var signature_cell = in_msg_body~load_ref();
slice signature_slice = signature_cell.begin_parse();
slice signature = first_bits(signature_slice, sig_length);
```

```
throw_unless(error::sig_valid_error, check_data_signature(sig_data, signature,
sig_pk));
```

Suggestion:

It is recommended to ensure that this is in accordance with the protocol design.

Resolution:

The client will use centralized services off-chain to impose a 14-day limit on staking, as well as blacklist management capabilities.

STA-3 Division Before Multiplication Result in Precision Loss

Severity: Medium

Status: Acknowledged

Code Location:

contracts/staking.fc#120

Descriptions:

The line of code may potentially cause precision loss due to the sequence of multiplication and division operations performed. Specifically, the calculation:

```
if(slashed == 0){  
    bonus_amount = withdraw_amount / 100 * 5;  
}
```

performs the multiplication and division in a manner that can lead to rounding errors and precision loss.

Suggestion:

Avoid division before multiplication and only perform division at last.

STA-4 Transfer Boundary Value

Severity: Medium

Status: Acknowledged

Code Location:

contracts/staking.fc#127

Descriptions:

When the contract funds are exactly equal to the amount to be withdrawn, it cannot be executed. This may be an unreasonable expected boundary value. For example, when there is 104 in the contract and 100 + 5 (bonus_amount) is about to be withdrawn, the execution will fail. At this time, the user can only withdraw 99 (103.95)

```
var [balance, _] = get_balance();  
throw_unless(error::not_enough, balance > total_amount);
```

Suggestion:

It is recommended to confirm whether the current boundary value needs to be changed (due to the ton chain payment mechanism, the execution can be successful when it is greater than, but there will be unfairness because gas will not be returned when the execution fails).

Resolution:

The client will use off-chain monitoring to ensure the funds of the staking contract are running normally.

UTI-1 Unused Definitions

Severity: Minor

Status: Fixed

Code Location:

contracts/utls.fc#1

Descriptions:

Unils.fc defines unused fee-related constants, uses const, performs mathematical calculations, and has unclear type definitions.

```
const const::tx_fee = 2 * 10000000;  
const const::tx_fee_send_jetton = 40000000;
```

Suggestion:

It is recommended to check whether constants should be used, or should be `const int` .

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

