TBCC Wallet Security Analysis

by Pessimistic

This report is public.

Published: February 25, 2021

Abstract 2
Disclaimer 2
Summary 2
General recommendations 2
Procedure 3
Project overview 4
  Project description 4
Manual analysis 5
  Critical issues 5
  Medium severity issues 5
    No documentation 5
  Low severity issues 6
    Code quality 6
    Project design 6
    Dependency management 6
    Gas consumption 6
  Notes 6
    Overpowered owner 6

Abstract

In this report, we consider the security of the token smart contract of TBCC Wallet 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. One 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, security audit is not an investment advice.

Summary

In this report, we considered the security of TBCC Wallet token smart contract. We performed our audit according to the procedure described below.

The audit showed several issues of low severity in the code. They do not endanger the security of the project.

However, a few project-level issues were found, including no documentation and dependency management.

The contract is already deployed to Ethereum mainnet.

General recommendations

We recommend adding public documentation to the project.

If the developers decide to redeploy the token contract, we also recommend fixing the mentioned issues.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
    - We scan project’s code base with automated tools: Slither and SmartCheck.
    - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
    - We manually analyze code base for security vulnerabilities.
    - We assess overall project structure and quality.
- Report
    - We reflect all the gathered information in the report.

Project overview

Project description

For the audit, we were provided with an address of the token contract of TBCC Wallet project, that is deployed to Ethereum mainnet already.

The project has no documentation.

The total LOC of audited sources is 141.

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 smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

No documentation

The project has no documentation. As a result, it is sometimes unclear for the auditor what is the intention of the code, is its behavior correct, and whether the architecture of the project is appropriate.

The token contract should also have publicly available documentation.

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code quality

- Consider declaring `abstract contract`s **ApproveAndCallFallBack** and **ERC20Interface** at lines 37 and 25 as `interface`s.
- Visibility of state variables should be declared explicitly.
- Consider declaring functions as `external` instead of `public` where possible.

Project design

- `approveAndCall()` function of **TBCC** contract partially implements eip-1438 pattern. Consider using eip-2612 instead.
- The logic of `multisend()` function in **TBCC** contract can also be useful for users. However, now it is only accessible by the owner.

Dependency management

No dependencies specified in the project. Standard SafeMath library is copied into the code. Consider managing it as a dependency.

Gas consumption

Consider declaring variables `name`, `symbol`, and `decimals` at lines 76–78 as `constant`s to optimize gas consumption.

Notes

Overpowered owner

The owner of the contract can pause all the transactions of the token.

This might become an issue in case if the private keys of the owner become compromised. Thus, we recommend designing contracts in a trustless manner or implementing a proper key management system, e.g. multisig.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Founder

February 25, 2021