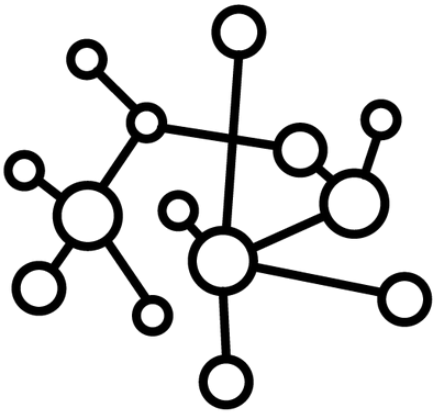

BLOCKCHAIN TESTAMENT

Blockchain Testament Security Analysis

by Pessimistic

This report is public

February 6, 2023

Abstract 2
Disclaimer 2
Summary 2
General recommendations 2
Project overview 3
Project description 3
Codebase update 3
Audit process 4
Manual analysis 5
Critical issues 5
C01. Wrong argument order (fixed) 5
Medium severity issues 6
M01. Insufficient documentation 6
M02. Possible price manipulation attacks (commented) 6
M03. Possible way to avoid payment of the full price of a subscription 6
Low severity issues 7
L01. Missing slippage check (commented) 7
L02. Code quality (fixed) 7
L03. Code quality (fixed) 7
L04. Code quality (fixed) 7
L05. Indexed events (commented) 7
L06. Redundant code (fixed) 8
L07. Misleading error message (fixed) 8
L08. Not handled scenario with a user balance change 8
Notes 9
N01. Owner role (commented) 9
N02. Edge case in the price calculation formula (commented) 9
N03. Automatical death confirmation when a quorum is zero (fixed) 9

Abstract

In this report, we consider the security of smart contracts of Blockchain Testament 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 Blockchain Testament smart contracts. We described the audit process in the section below.

The initial audit showed one critical issue: Wrong argument order. The audit also revealed several issues of medium severity: Insufficient Documentation, Possible price manipulation attacks and Possible way to avoid payment of the full price of a subscription. Moreover, several low-severity issues were found.

The overall quality of the initial code was good. However, it contained several issues that we recommended fixing prior to using contracts in a production environment.

After the audit, the developers provided a new version of the code. Several issues were not addressed in that update. However, developers fixed critical issue and improved NatSpec coverage.

General recommendations

We recommend fixing the mentioned issues.

Project overview

Project description

For the initial audit, we were provided with Blockchain Testament project on a private GitHub repository, commit ea26eb4e8ba5e0c99989effc65769b7ef465fab4.

The scope of the initial audit included the whole repository.

The documentation for the project includes the following link.

All 8 tests pass successfully. The code coverage is 89.76%.

The total LOC of audited sources is 576.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit 990d323cf38b4e4204b09c6f0ff308eeb019b8c3.

This update included a fix for the Wrong argument order issue, as well as improvements to the code structure. However, several issues were not fixed.

All 9 tests pass successfully. The code coverage is 89.29%.

Audit process

We started the audit on January 18 and finished on January 23, 2023.

We inspected the materials provided for the audit. Then, we prepared a list of questions to the developers. During the work, we stayed in touch with them 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 calculations are correct;
- Whether reentrancy attacks are possible due to NFT usage;
- The general user flow.

We manually verified all the occurrences found by the tools.

We ran tests and calculated the code coverage.

Also, we analyzed the project with Slither and manually verified all found occurrences.

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

After the initial audit, we discussed the results with the developers. On January 24 the developers provided us with an updated version of the code. In this update, they fixed several of the issues from our report, optimized the code structure, added one test, and added additional comments.

We reviewed the updated codebase and did not find any issues in the newly implemented functionality.

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.

C01. Wrong argument order (fixed)

In **BT.sol** line 337 creates `DeathConfirmation` structure. However, provided arguments have the wrong order. According to the struct definition, the `quorum` field should be a second one.

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

Medium severity issues

Medium 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. Insufficient documentation

We recommend adding documentation for the project, as well as covering code with NatSpec comments as it helps to avoid errors.

The developers added NatSpec comments. Still, we recommend adding technical documentation.

M02. Possible price manipulation attacks (commented)

Consider not relying on pair reserves to avoid incorrect price calculation. Pair reserves can be manipulated by usual swap. The best practice is to use TWAP (Time-Weighted Average Price).

The developers do acknowledge the existence of price manipulation attacks. However, they do not see that as an issue for their project.

M03. Possible way to avoid payment of the full price of a subscription

User can first subscribe to a cheaper subscription (also works with `FREEMIUM`) and immediately upgrade their subscription to a desired one. By doing this, the user will pay less amount than directly subscribing to it, since when the user upgrades their subscription, they receive 50% discount.

The developers rolled out a partial fix to the issue. As a result, it is not possible to use `FREEMIUM` plan for the exploit.

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.

L01. Missing slippage check (commented)

We recommend adding a parameter for the maximum amount of payment tokens that user spends since it helps to mitigate price slippage.

The developers decided not to implement a such check in order to save gas.

L02. Code quality (fixed)

`If` statements at lines 318 and 376 are redundant since payment plans with zero price are handled in `getPriceInPaymentToken` function.

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

L03. Code quality (fixed)

In `billPayment` function, line 433 transfers payment from the user in case when subscription plan is neither `FREEMIUM` nor `PREMIUM`. However, this logic will still transfer zero amount of tokens in case when another freemium plan is implemented. This might be the case when developers decide to change freemium plan benefits. Thus, we recommend adding an additional check for the actual price.

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

L04. Code quality (fixed)

The project does not contain `package-lock.json` file. We recommend adding it in order to make the development process easier when the project is maintained by multiple developers.

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

L05. Indexed events (commented)

Marking event arguments as `indexed` is useful when they are considered for further search. We recommend reviewing implemented events.

The developers commented that the arguments are not indexed in the events intentionally to save gas.

L06. Redundant code (fixed)

In **BT.sol** line 73 declares `FREEMIUM_FEE` variable that is never used in the code. We recommend removing redundant code.

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

L07. Misleading error message (fixed)

The `require` statement at lines 418-422 has an error message `"no more than one period"`, but it is possible to pay for two periods. This can be done by calling `billPayment` function in the next block after creating a testament.

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

L08. Not handled scenario with a user balance change

There is a scenario where some successors might not be able to receive tokens from a testament. This might be the case when user's balance changes after `perShare` variable has been set. Such balance changes might be the result of interacting with either rebasing tokens, tokens with included transfer fee or when other user transfer tokens via `transferFrom` function.

The developers decided to move all of the user tokens to the contract. This approach helps to avoid situation when other contract changes user balance by calling `transferFrom` function.

Notes

N01. Owner role (commented)

The owner role is crucial for the project. It is responsible for the following:

- Setting fee address;
- Adding new subscriptions;
- Enabling and disabling subscriptions.

In the current implementation, the system depends heavily on the owner. Thus, there are scenarios that can lead to undesirable consequences for the project and its users, e.g., if admin's private keys become compromised. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers: The increased power of the owner role is a part of the business logic.*

N02. Edge case in the price calculation formula (commented)

If `reserveQuote` is less than `reserveBase`, there might be issues with the formula at line 195. If the ratio is very small, then the price will equal zero and adding one will give an inaccuracy of calculation. We recommend carefully reviewing pairs that will be used in the project.

The developers do not consider that as an issue.

N03. Automatical death confirmation when a quorum is zero (fixed)

When the `quorum` is equal to 0, due to the check at line 665, the user's death will be automatically confirmed without the `CONFIRMATION_PERIOD` wait time. Note that `voting.confirmationTime` will be equal to 0 since nobody will be able to call `confirmDeath`. Moreover, the `DeathConfirmed` event will not be emitted for this user.

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

This analysis was performed by Pessimistic:

Pavel Kondratenkov, Security Engineer
Yhtyyar Sahatov, Junior Security Engineer
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

February 6, 2023