wibson

Wibson Contracts Audit

—

Presented by

Manuel Araoz
CTO, Zeppelin

September 05, 2018

zeppelin
audits

01. Introduction

This document includes the results of the audit performed by the Zeppelin team on the Wibson project, at the request of the Wibson team (https://wibson.org/).The audited code can be found in the public wibson-core repo on Github repository, and the version used for this report is commit

```
3304352e76373b09d8df65e07ed9cfcefeee0fdb
```

Some fixed and partially fixed issues from a previous audit can also be found in Appendix A.

The goal of this audit is to review their data exchange contracts, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

Note: In addition, the Wibson team asked us to review the minor renaming changes introduced in the commit below. This changes introduce no security issues.

```
b69827c8724484830efe286fe2056d9398059607
```

— Disclaimer

Note that as of the date of publishing, the contents of this document reflect the current understanding of known security patterns and state of the art regarding [topic, eg: smart contract security, compilers]. Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

— Structure of the document

This report contains a list of issues and comments on different aspects of the project. Each issue is assigned a severity level based on the potential impact of the issue, as well as a small example to reproduce it and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

02. About Zeppelin

Zeppelin Solutions is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Zeppelin Solutions has developed industry security standards for designing and deploying smart contract systems.

Zeppelin Solutions is the creator, maintainer, and major contributor of OpenZeppelin, the standard framework for secure smart contract development, maintained by a community of 3000+ developers distributed around the globe.

Over $600 million have been raised with Zeppelin's audited smart contracts. Clients include Golem, Brave, Augur, Blockchain Capital, Status, Cosmos, and Storj, among others.

More info at: https://zeppelin.solutions

03. Severity level reference

Every issue in this report was assigned a severity level from the following:

CRITICAL

Critical severity issues need to be fixed as soon as possible.

HIGH

High severity issues will probably bring problems and should be fixed.

MEDIUM

Medium severity issues could potentially bring problems and should eventually be fixed.

LOW

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

04. List of issues by severity

CRITICAL

None.

HIGH

None.

MEDIUM

None.

LOW

None.

05. Notes & Additional Information

- Consider renaming the notary argument to `notaryAddress`, to make it clear that it is an address and not an structure, name, or something else.

  Update: See #57 (comment) and #57 (comment)

- On many functions, it is documented that the return value is "Whether [something] was successful or not." This implies that false is returned when something goes wrong, but in most of the cases the function reverts instead. Consider making the docs clearer, saying something like: "returns true when [something] was successful. Reverts when it was not". Ideally, specify all the cases that revert.
  Fixed in: 82313a9

- On `MultiMap#insert`, there is a check that a push to the array increases the length by 1. That is a correct safeguard, but if that fails it means that there is a very big bug on Solidity and we have bigger problems. Consider moving the check to the test suite.

  Fixed in: 4812c79 **and** e591b24

- Along the same lines, in the `MultiMap#exist` function, there is a superfluous check of the targetIndex being smaller than the length, which should be already covered by the condition on the right of the and operator (&&). Consider moving the first check to the test suite.
  See: #57 (comment)

- `DataOrder` has an `initialBudgetForAudits`, which is not used. Consider removing it.
  Fixed in: 097ac5c

- Consider renaming `publicKey` to buyer `buyerPublicKey` in `DataOrder`.
  Fixed in: a4dd297

- In `DataOrder`, consider using `hasSellerBeenAccepted` inside `getSellerInfo` and `getNotaryForSeller`, to avoid code duplication.

  Fixed in: 8c39497

- There are duplicate checks in lines 122 and 123 of `DataOrder` as the `orderStatus` completed should be equivalent with `transactionCompletedAt` being nonzero. Consider removing one of these checks.
  Fixed in: fd56003

- In `DataOrder`, as the `getDataResponseStatusAsString` default condition should never be reached, consider throwing instead of reverting.
  Fixed in: 5e48aeb

- In `DataExchange`, `validNotaries` is never used. Consider removing it.
  Fixed in: ab5175c

- In `DataExchange`, `setMinimumInitialBudgedForAudits` can be called while paused. Consider adding the `whenNotPaused` modifier.
  Fixed in: 11fcfa7

- In `DataExchange`, consider renaming `NotaryAdded` to `NotaryAddedToOrder` To Order, because the current name is too similar to `NotaryRegistered`.
  Fixed in: f0ac909

- In `DataExchange`, it is not necessary to use `SafeMath` on `allDistinct`, because the maximum value is 5.

  Fixed in: b65b096

- In `DataExchange`, `chargeBuyer` checks the allowance. This is duplicating the check in `StandardToken`.
  Fixed in: 093abd7

- In Data Exchange, a comment in payPlayers is wrong. It should say: “if notarization was done and data is invalid, order price tokens go back to buyer. **Fixed**

- In `DataExchange`, the `TransactionCompleted` event is emitted when `DataResponse.statu` is `TransactionCompleted` or `RefundedToBuyer`. This could be misleading. Consider adding an event for when the order is not completed but refunded. **Fixed in:** dboocf8

- In `DataExchange`, the new payment logic is unclear, and there are inconsistencies between the code and the documentation. In particular, there seems to be a with the arguments of the min function. This min is not mentioned in the documentation, and it's likely that it is meant to avoid subtracting more than the available budget. Consider replacing `notarizationFee` with `totalCharges` in line 525, or fixing the documentation.

  Fixed in: 632e19e