BlockchainLabs.NZ

Blockchain Solution Design & Development

Audit report for TravelBlock Contracts

July 2018

Contents

Preamble

Scope

Analysis

Issues

- Severity Description
- Minor
- Moderate
- Major
- Critical

Conclusion

- Disclaimer

Preamble

This audit report was undertaken for TravelBlock, by their request, and has subsequently been shared publicly without any express or implied warranty.

Solidity contracts were sourced directly from the TravelBlock team and the most recent commit we have audited is this commit

We would encourage all community members and token holders to make their own assessment of the contracts.

Scope

The following contracts were subject for analysis:

- `contracts/`
  - `Admin.sol`
  - `RewardToken.sol`
  - `TRVLToken.sol`
  - `Whitelist.sol`

Analysis

- Functional test
- Dynamic Analysis
- Test Coverage
- Gas Consumption

Issues

Severity Description

Minor	A defect that does not have a material impact on the contract execution and is likely to be subjective.
Moderate	A defect that could impact the desired outcome of the contract execution in a specific scenario.
Major	A defect that impacts the desired outcome of the contract execution or introduces a weakness that may be exploited.
Critical	A defect that presents a significant security vulnerability or failure of the contract across a range of scenarios.

Minor

- Naming problem reduces code readability `best practice`
- The name of this function and return statement could be improved. The name implies that it gets/returns a percentage, but it actually returns an amount of calculated reward tokens.

  RewardToken.sol, Line 87:

  ```
  _rewardPercentage =
  _amount.mul(rewardPercentage[_rewardPercentageIndex]).div(rewardPercentageDivisor);
  ```
  View on GitHub

- Enforce the use of specific Solidity compiler version `best practice`
- The code is declaring the following `pragma solidity ^0.4.24` which is latest version currently. The use of `^` states that any solc compiler greater or equal to `0.4.24` and lower than `0.5.0` can be used to run the code. However, there could be some discrepancies in the compiled code which could cause unexpected behaviour when using different solidity version. Therefore we recommend using `pragma solidity 0.4.24` without the `^` and adding `"solc": "0.4.24"` to the dependencies to enforce testing, compiling and deploying with that compiler. View on GitHub

Moderate

- None found

Major

- None found

Critical

- None found

Conclusion

The contracts show care taken by the development team to follow the latest features and a strong knowledge of Solidity. All the contracts are well written and organized. They are also using a fully-tested and audited framework, openzeppelin-solidity, which improves confidence in the contracts security. Additionally there is a very high test coverage which should increase confidence in the security of these contracts, and their maintainability in the future.

Overall we are satisfied that these Smart Contracts do not exhibit any known security vulnerabilities.

Disclaimer

Our team uses our current understanding of the best practises for Solidity and Smart Contracts. Development in Solidity and for Blockchain is an emerging area of software engineering which still has a lot of room to grow, hence our current understanding of best practices may not find all of the issues in this code and design.

We have not analysed any of the assembly code generated by the Solidity compiler. We have not verified the deployment process and configurations of the contracts. We have only analysed the code outlined in the scope. We have not verified any of the claims made by any of the organisations behind this code.

Security audits do not warrant bug-free code. We encourage all users interacting with smart contract code to continue to analyse and inform themselves of any risks before interacting with any smart contracts.