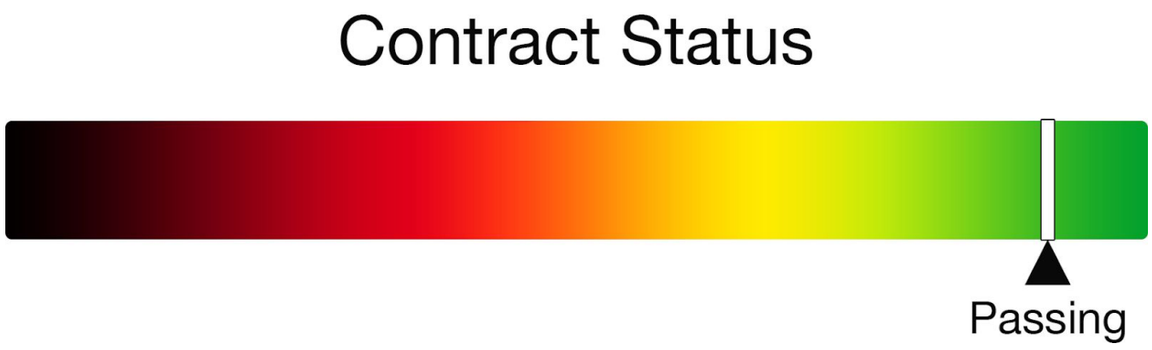
Tech Coins Contract Audit

Prepared by Hosho
June 20th, 2018

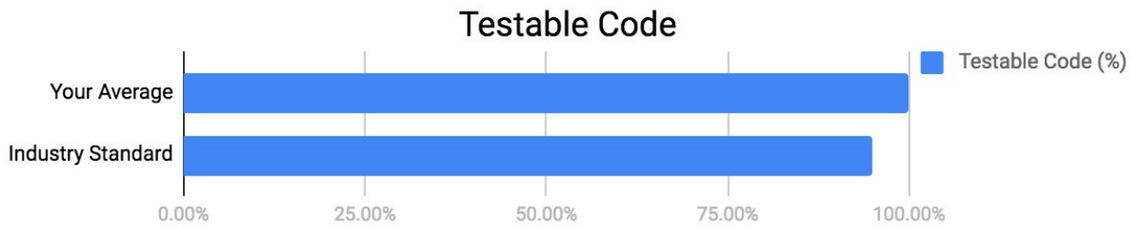
Report Version: 1.0

Executive Summary

This document outlines the overall security of Tech Coins' smart contract as evaluated by Hosho's Smart Contract auditing team. The scope of this audit was to analyze and document Tech Coins' token contract codebase for quality, security, and correctness.



These contracts have passed the rigorous auditing process performed by the Hosho team. (See Complete Analysis)



Testable code is 100% which is greater than industry standard. (See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Hosho recommend that the Tech Coins team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table Of Contents

1. Auditing Strategy and Techniques Applied 3

2. Structure Analysis and Test Results 4

2.1 Summary

2.2 Coverage Report

2.3 Failing Tests

3. Complete Analysis 5

4. Closing Statement 6

5. Appendix A 7

Test Suite Results

6. Appendix B 10

All Contract Files Tested

7. Appendix C 11

Individual File Coverage Report

1. Auditing Strategy and Techniques Applied

The Hosho team has performed a thorough review of the smart contract code, the latest version as written and updated on June 12th, 2018. All main contract files were reviewed using the following tools and processes. (See All Files Covered)

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing ERC-20 Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks; and
- Is not affected by the latest vulnerabilities.

The Hosho team has followed best practices and industry-standard techniques to verify the implementation of Tech Coins' token contract. To do so, the code is reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.
5. Deploying the smart contract to testnet and production networks using multiple client implementations to run live tests.

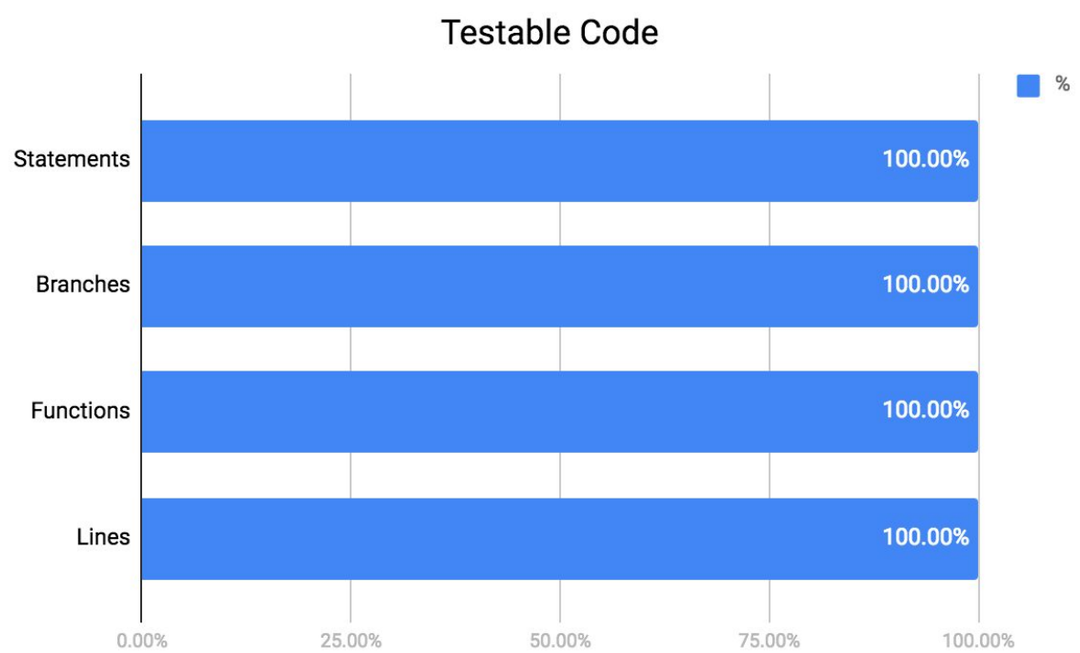
2. Structure Analysis and Test Results

2.1 Summary

Tech Coins has provided a well-implemented ERC-20 token, with additional functionality, including the minting, pausing, and burning of tokens, allowing the token to be both created and destroyed as needed. The coin is implemented utilizing OpenZeppelin very heavily to ensure proper structures and well-tested features.

2.2 Coverage Report

As part of our work assisting Tech Coins in verifying the correctness of their contract code, our team was responsible for writing a unit test suite using the Truffle testing framework.



For each file see Individual File Coverage Report

2.3 Failing Tests

No failing tests.

See Test Suite Results for all tests.

3. Complete Analysis

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged "Resolved" or "Unresolved" depending on whether they have been fixed or still need addressing. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.
- **Low** - The issue has minimal impact on the contract's ability to operate.
- **Informational** - The issue has no impact on the contract's ability to operate, and is meant only as additional information.

No issues were discovered in the Tech Coins' contracts during the audit process.

---

4. Closing Statement

The Hosho team is grateful to have been given the opportunity to work with the Tech Coins team. It was a pleasure to work with the Tech Coins team and we look forward to working with them on any future projects.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Tech Coins contract is free of any critical issues, having passed the rigorous Hosho auditing process.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

We at Hosho recommend that the Tech Coins team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Appendix A

Test Suite Results

Contract: Ownership Tests for UCoin

Deployment

√ Should deploy with the owner being the deployer of the contract

Transfer

√ Should not allow a non-owner to transfer ownership (56ms)

√ Should not allow the owner to transfer to 0x0 (59ms)

√ Should allow the owner to transfer ownership (97ms)

Contract: BurnableToken Tests for UCoin

√ Should allow owners to burn own tokens (109ms)

√ Should require token burn amount to be less than or equal balance

√ Should require logging burn event (96ms)

Ensure 'UCoin' defines the ERC20 Token Standard Interface

√ Should have the correct 'name' definition

√ Should have the correct 'approve' definition

√ Should have the correct 'totalSupply' definition

√ Should have the correct 'transferFrom' definition

√ Should have the correct 'decimals' definition

√ Should have the correct 'balanceOf' definition

√ Should have the correct 'symbol' definition

√ Should have the correct 'transfer' definition

√ Should have the correct 'allowance' definition

√ Should have the correct 'Transfer' definition

√ Should have the correct 'Approval' definition

Contract: Ownership Tests for UCoin

Deployment

√ Should deploy with the owner being the deployer of the contract

Transfer

√ Should not allow a non-owner to transfer ownership (43ms)

√ Should not allow the owner to transfer to 0x0 (48ms)

√ Should allow the owner to transfer ownership (87ms)

Contract: ERC-20 Tests for UCoin

√ Should deploy a token with the proper configuration (59ms)

√ Should allocate tokens per the minting function, and validate balances (323ms)

√ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (112ms)

√ Should not transfer negative token amounts (46ms)

√ Should not transfer more tokens than you have (45ms)

√ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (67ms)

√ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (63ms)

√ Should require transfer to valid address (43ms)

√ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (219ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (43ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0 (38ms)

√ Should not transfer tokens to 0x0 (38ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (48ms)

√ Should allow an approval to be set, then increased, and decreased (288ms)

√ Should allow minting only when the minting finished flag has not been set (87ms)

Contract: Pause Tests for UCoin

Deployment

√ Should deploy in an un-paused state

Pause configuration

√ Should be able to be paused (73ms)

√ Should be able to be unpaused (122ms)

√ Should not be able to be unpaused while unpaused

√ Should not be able to be paused while paused (91ms)

Contract: SafeMath

√ Should skip operation on multiply by zero

√ Should revert on multiply overflow

√ Should allow regular multiple

√ Should revert on divide by zero

√ Should allow regular division

√ Should revert on subtraction overflow

√ Should allow regular subtraction

√ Should revert on addition overflow

√ Should allow regular addition

6. Appendix B

All Contract Files Tested

File	Fingerprint (SHA256)
contracts/UCoin.sol	857680b2c5b67f23d017827aa5f5cd73111bb3c2d80956344f8bf0406207290f
contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol	c6e760dc28a45872e67ad98066c973c9ef52d92041608ca71225defe1379f191
contracts/openzeppelin-solidity/contracts/math/SafeMath.sol	d4d0a9aafa36cd3c3b06a6d2959f252ed07977d51e7ff60dcdf904b05d2e5361
contracts/openzeppelin-solidity/contracts/ownership/ownable.sol	babaa6f0611e340344d684696f668294d8835467da172f82bf28e0b22bc1b26f
contracts/openzeppelin-solidity/contracts/token/ERC20/BasicToken.sol	5c930c1b0b5a1e689c95c2bc0701afcc68473918c5a7727bd14478ddef057df2
contracts/openzeppelin-solidity/contracts/token/ERC20/BurnableToken.sol	b3aa6387d986270f2c232d8cffe965a253dc9f6d5e3d9c28cd88f897b496d450
contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol	53c6af71322f1e0d7cb8b52d2f46005ef105e39a6e2151718dd7c690517bd12b
contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol	a9cf1d9073a8a58ca6044a1720a93a69020ea80fab3f5169192630590707d593
contracts/openzeppelin-solidity/contracts/token/ERC20/MintableToken.sol	eaf2f9e8940e6124da4c9f11803bf3be12578bb454818bf6a6f160002fdf685d
contracts/openzeppelin-solidity/contracts/token/ERC20/PausableToken.sol	d3697e8720db62e48026c1d2d42d8c8325d35950dbd489afac73bd86ab74b114
contracts/openzeppelin-solidity/contracts/token/ERC20/StandardToken.sol	66b71fd90d53eb78d951cee6d3259cde72ece31c0bede645e7bda74068f3d288

7. Appendix C

Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
contracts/UCoin.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/math/SafeMath.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/ownership/ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/MintableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/PausableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/openzeppelin-solidity/contracts/token/ERC20/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
All files	**100.00%**	**100.00%**	**100.00%**	**100.00%**