Blockpass Contract Audit

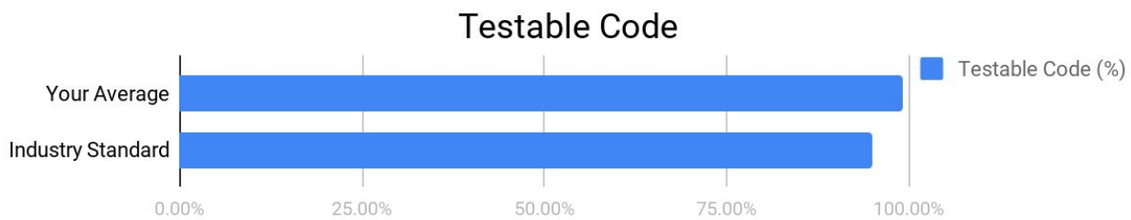
Prepared by Hosho
May 29, 2018

V 1.0

Executive Summary

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

There are two low level issues and a code suggestion that need to be addressed. (See Complete Analysis)



Testable code is 99.24% which is above 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 Blockpass 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

The Hosho Team has performed a thorough review of the smart contract code, the latest version as written and updated on May 3, 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; and
- Uses methods safe from reentrance attacks.
- Is not affected by the latest vulnerabilities.

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of Blockpass's token contract. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they were discovered. Part of this work included 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.

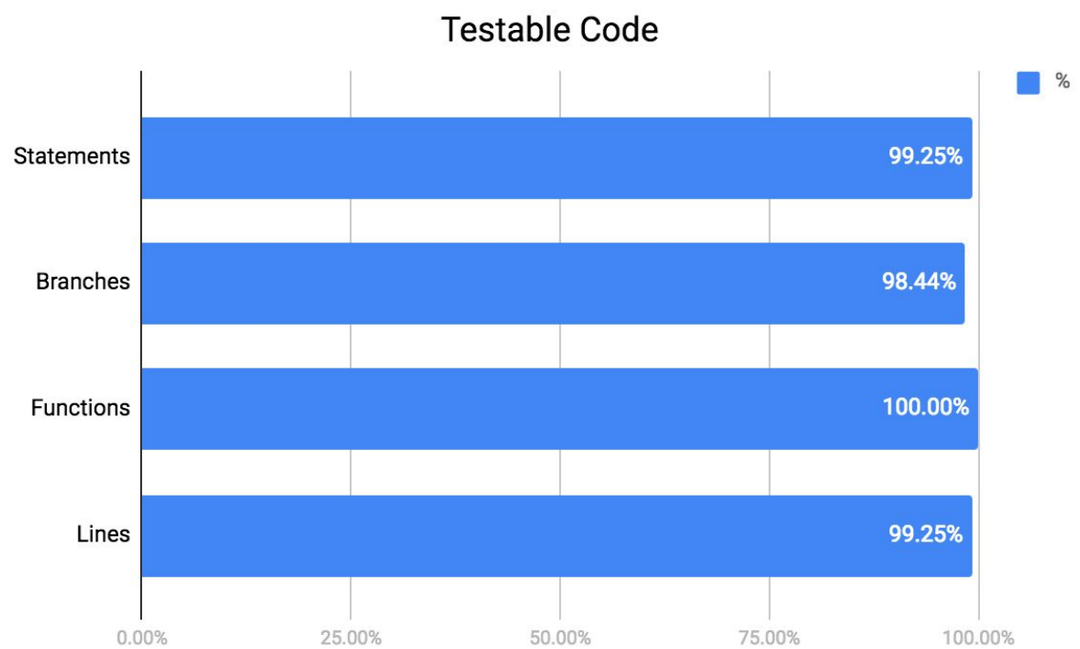
2. Structure Analysis and Test Results

2.1 Summary

The Blockpass contracts comprise an ERC-20 token based on Open Zeppelin contracts. The token is ERC-20 compliant with additional functions to upgrade the token. These upgrade functions override the Open Zeppelin contracts only in the case that a new token is available and the contract is paused.

2.2 Coverage Report

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



For individual files see Additional Coverage Report

2.3 Failing Tests

No failing tests.

See Test Suite Results for all tests.

NOTE: Due to issues with Solidity Coverage on 0.4.22 and the new constructor() format, the constructors were changed to function `Pass1` and `Operable` for testing purposes.

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 addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

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

3.1 Resolved, Medium: Missing Event

Contract: Pass1.sol

Explanation

All issuance, including initial setup, should emit an event for tracking purposes. The `Pass1` function within this contract initializes the available tokens, but there is no event emitted for this issuance. This breaks 3rd party integrations that rely on this event for reporting the initialization of the contract.

Resolution

Resolved by the Blockpass team; a Transfer event was added on line 45 for the original issuance of the tokens.

3.2 Resolved, Low: Unused Modifier

Contract: Operable.sol

Explanation

This contract contains the modifier `isOperator,` which is a check to validate that the `msg.sender` is included in the list of `operators`. No functions utilize this check and it should be removed or added to the appropriate functions as designed.

Resolved

The `isOperator` modifier has been removed by the Blockpass Team, resolving this issue.

3.3 Unresolved, Low: Solidity Version

Contract: Pass1.sol

Explanation

Due to the update of this contract to use `constructor()` instead of `function`, the solidity compiler version in this file should be updated to 0.4.22 in order to properly work with 3rd party tools, such as Solidity Coverage.

3.4 Unresolved, Low: Solidity Version

Contract: Operable.sol

Explanation

Due to the update of this contract to use `constructor()` instead of `function`, the solidity compiler version in this file should be updated to 0.4.22 in order to properly work with 3rd party tools, such as Solidity Coverage.

3.5 Unresolved, Informational: Unhittable Code

Contract: Pass1.sol

Explanation

The `mint` function within this contract is supposed to return a `bool` value. Since there is a `revert()` on line 86 the return is never called.

---

4. Closing Statement

We are grateful to have been given the opportunity to work with the BlockpassTeam.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Blockpass contract is free of any critical issues. However, there are a few low level issues and a suggestion that require mention.

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 Blockpass 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

NOTE: Due to issues with Solidity Coverage on 0.4.22 and the new constructor() format, the constructors were changed to function `Pass1` and `Operable` for testing purposes.

Contract: ERC-20 Tests for Pass1

✓ Should deploy a token with the proper configuration (99ms)

✓ Should allocate tokens per the minting function, and validate balances (253ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (127ms)

✓ Should not transfer negative token amounts (45ms)

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

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (56ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (54ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (227ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0

✓ Should not transfer tokens to 0x0 (41ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

✓ Should allow an approval to be set, then increased, and decreased (224ms)

Ensure 'Pass1' 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: Additional Tests Written for Pass1

✓ Should Upgrade Existing Tokens and Burn Previous Supply (286ms)

✓ Should Require New Token to be Minted (160ms)

✓ Should Whitelist Address (103ms)

✓ Should Whitelist Multiple Addresses from Original Whitelist (140ms)

✓ Should Whitelist Multiple Addresses in Client Code Whitelist (141ms)

✓ Should Remove Address from Whitelist (94ms)

✓ Should Remove Multiple Addresses from Original Whitelist (107ms)

✓ Should Remove Multiple Addresses from Client Code Whitelist (95ms)

✓ Should Mint New Tokens During Upgrade (63ms)

✓ Should Burn Tokens During Mint Upgrade (114ms)

✓ Should Expect Revert if Amount to Burn is Greater than Supply (102ms)

✓ Should Expect Revert in Clients Burn Function

✓ Should Add Operators (66ms)

✓ Should Remove Operators (115ms)

✓ Should Test Operators

✓ Should Check if Paused

✓ Should Check Whitelist

✓ Should Not Mint Tokens to 0x0

✓ Should Require Whitelisted Accounts

✓ Should Expert Revert in Clients Mint Function (295ms

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

Contract: ERC-20 Tests for Pass1 Minting

✓ Should deploy a token with the proper configuration (75ms)

✓ Should allocate tokens per the minting function, and validate balances (261ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (188ms)

✓ Should not transfer negative token amounts (42ms)

✓ Should not transfer more tokens than you have

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (54ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (58ms)

✓ Should require transfer to valid address

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (251ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0 (40ms)

✓ Should not transfer tokens to 0x0 (47ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b

✓ Should allow an approval to be set, then increased, and decreased (191ms)

✓ Should require minting tokens to stay under maximum supply

✓ Should allow minting only when the minting finished flag has not been set (53ms)

Contract: Ownership Tests for Pass1

Deployment

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

Transfer

✓ Should not allow a non-owner to transfer ownership

✓ Should not allow the owner to transfer to 0x0

✓ Should allow the owner to transfer ownership (64ms)

Contract: Pause Tests for Pass1

Deployment

✓ Should deploy in an un-paused state

Pause configuration

✓ Should be able to be paused (46ms)

✓ Should be able to be unpaused (69ms)

✓ Should not be able to be unpaused while unpaused

✓ Should not be able to be paused while paused (53ms)

6. Appendix B

All Contract Files Tested

Commit Hash: 2e8b8ff1364bbb35a16d5c0b543b145942e1817a

File	Fingerprint (SHA256)
contracts/Operable.sol	813404a15426f01cabea420b53f2cbf2f5aab09bb0ae0a5077acf80195aa9e0f
contracts/Pass1.sol	cd1e59787b500cc67e08d723ec0a780be94e0074ee2427d042d1d02bd2ff089e
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	78bf21e029fc3f1c38151915db9ccce2f0553bfeae9b6685fde1c297091cdb6f
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	79c36e0aec10ebe744135afa9e0edb27d0a2e2f15cc3a2685169c679154fc05e
contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	35dcf237365077adb1dd8d1da9e05f2b4f8e9d7b49311fc8a09b28d4ce191579
contracts/zeppelin-solidity/contracts/ownership/Whitelist.sol	6a40eb3c5b4e557fd24355e4bfd3992cdbe7e212fdc68d8c220747d14de47f77
contracts/zeppelin-solidity/contracts/token/ERC20/BasicToken.sol	2662ea846c0c6ca2bd32dfdb97ab20ae4108e7abade53962f146f634e9f4eba7
contracts/zeppelin-solidity/contracts/token/ERC20/BurnableToken.sol	e31c914853c72951ea1ecd3673b1d1dcc8b50c217acf8b1f6d7df5b8fe0a54e4
contracts/zeppelin-solidity/contracts/token/ERC20/DetailedERC20.sol	3d5721993d83b727de3066d7c45cf333a6f62eef14c62a2dd82e6a1f0067b7bc
contracts/zeppelin-solidity/contracts/token/ERC20/ERC20.sol	6b75acd05c29968b057ec1facf659c064dbe0a79ac01444530629f01ef3a3abf
contracts/zeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol	86c0a5fc6cb564ae77140da57a8ff9a22f46404240e69a6782ff741e286d373a
contracts/zeppelin-solidity/contracts/token/ERC20/MintableToken.sol	cba240b59fed2016bb9ed001a8c50a74150cf3e68eab41474914c7af21ff6427
contracts/zeppelin-solidity/contracts/token/ERC20/PausableToken.sol	2686975792c8dc8f507dc143aa0abb7f49eb837c8bde51017216dae3fa38dafd
contracts/zeppelin-solidity/contracts/token/ERC20/StandardToken.sol	77e45da1164753f886d7395987b46deb036eca32c2e7322ef7a2764a08f7c5da

7. Appendix C

Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
contracts/Operable.sol	100.00%	100.00%	100.00%	100.00%
contracts/Pass1.sol	97.14%	93.75%	100.00%	96.77%
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/ownership/Whitelist.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/DetailedERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/MintableToken.sol	100.00%	100.00%	100.00%	100.00%

contracts/zeppelin-solidity/contracts/token/ERC20/PausableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
All files	**99.25%**	**98.44%**	**100.00%**	**99.25%**