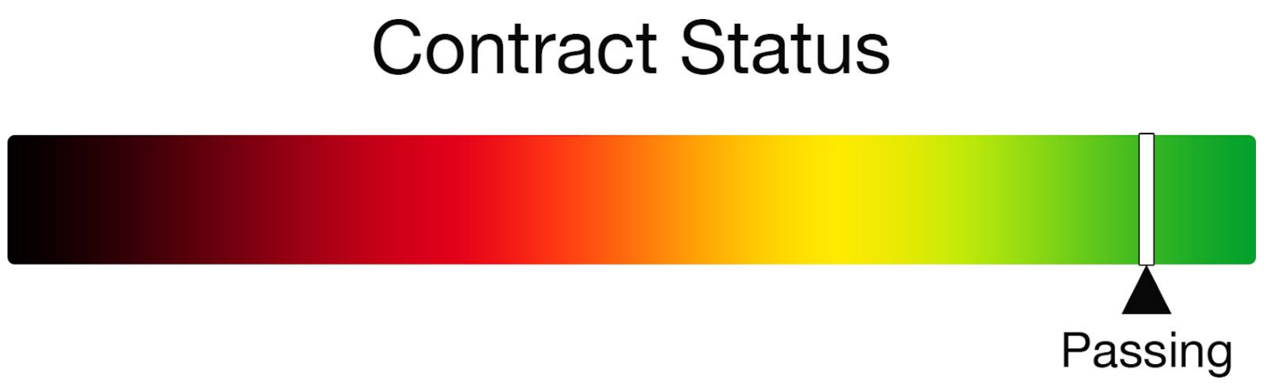
Invictus Capital Contract Audit

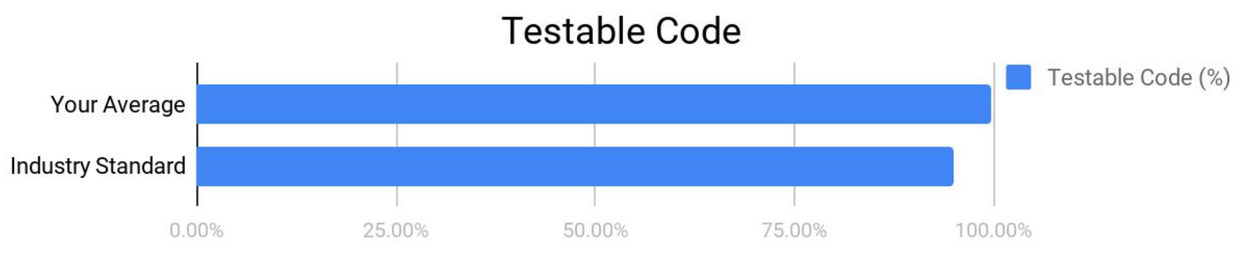
by Hosho, April 2018

Executive Summary

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



The low level issue contained in this contract has been remediated. (See Complete Analysis)



Testable code is 99.72% which is higher 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 Invictus Capital 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 April 24, 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 Invictus Capital'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.

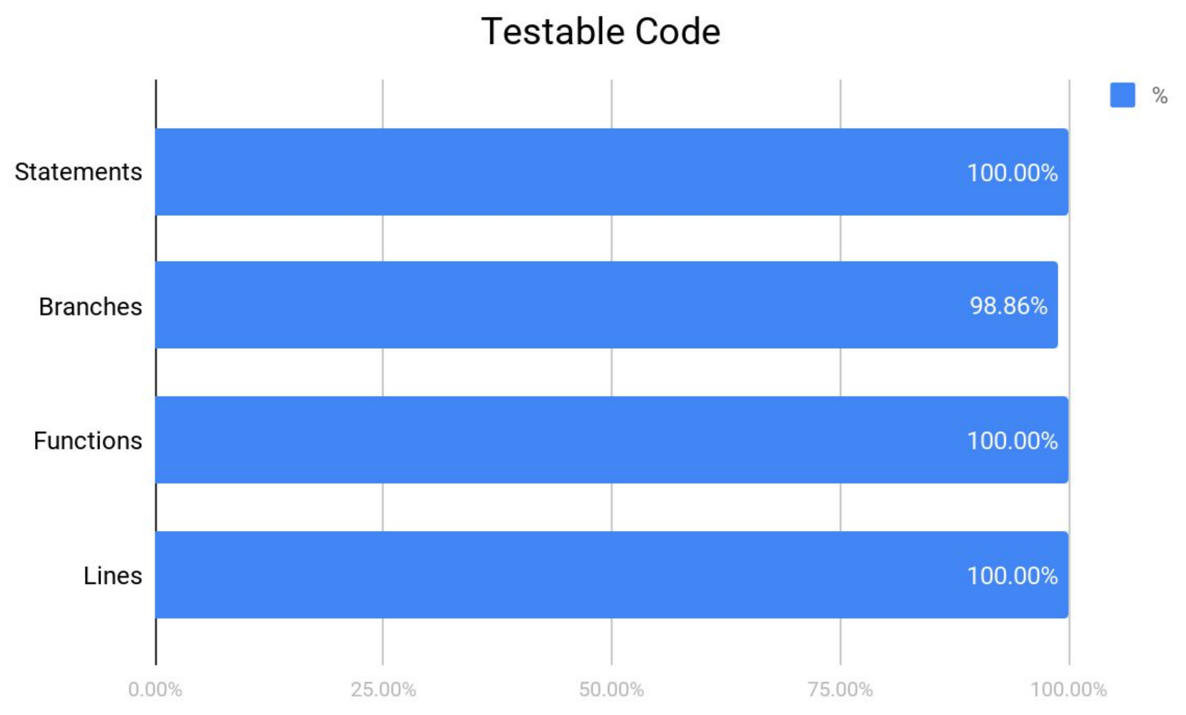
2. Structure Analysis and Test Results

2.1. Summary

The Invictus Capital contracts comprise an ERC-20 token with an issuance system to distribute the tokens. This system is controlled by multiple wallets as well as a vesting system to deliver the tokens over time. As the hard cap is being implemented outside of this contract, the tokens for the Invictus Capital team are distributed via the vesting schedule as opposed to directly to the team during normal token issuance.

2.2 Coverage Report

As part of our work assisting Invictus Capital 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.

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, Low: Incorrect Variable Type

IHF

Explanation

The ERC-20 standards, state that the declaration for `decimals` should be a `uint8` as opposed to `uint256` as it currently is in this contract.

Resolution

The declaration of the `decimals` variable has been updated by the Invictus Capital team to a `uint8`, returning the contract to compliance with ERC-20 standards.

---

4. Closing Statement

We are grateful to have been given the opportunity to work with the Invictus Capital Team.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Invictus Capital contract is free of any critical issues.

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 Invictus Capital Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Coverage Report:

Contract: ERC-20 Tests for IHF

✓ Should deploy a token with the proper configuration (82ms)

✓ Should allocate tokens per the minting function, and validate balances (229ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (158ms)

✓ Should not transfer negative token amounts (152ms)

✓ Should not transfer more tokens than you have

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (38ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (42ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (140ms)

✓ 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

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

✓ Should allow an approval to be set, then increased, and decreased (180ms)

Contract: More tests for IHF

✓ Should fail to deploy a token with invalid backupAccount (43ms)

✓ Should fail to deploy a token with invalid futureBlockNumber

✓ Should deploy a token with the proper configuration (187ms)

✓ Should fail to set the vesting contract to an invalid address

✓ Should fail to set the vesting contract from an address that isn't a fundWallet

✓ Should fail to batch allocate tokens before vesting contract is set

✓ Should fail to adjustBalance before vesting contract is set

✓ Should set the vesting contract

✓ Should fail to set the vesting contract because it has already been set

✓ Should allocate tokens per the minting function, and validate balances (291ms)

✓ Should fail to update the funding end block when new end block isn't greater than the current block number

✓ Should update the funding end block

✓ Should fail to transfer tokens before trading is enabled

✓ Should fail to enable trading when required block hasn't been reached

✓ Should remove the balance from 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 (52ms)

✓ Should reject eth payment of 10

✓ Should accept eth payment without value

✓ Should enable trading after failing before block number has been reached (187ms)

✓ Should fail to update the funding end block after it has already been passed

✓ Should fail to batch allocate tokens when funding end block has passed

✓ Should fail to adjustBalance after funding end block has been reached

✓ Should remove eth from the account

✓ Should fail to claim tokens from an invalid (0) address

✓ Should fail to claim tokens from an invalid address (41ms)

✓ Should fail to change backup wallet to an invalid wallet

✓ Should change backup wallet to a new wallet

✓ Should confirm new backup wallet works (by resetting back to the original backup)

✓ Should fail to change fund wallet to an invalid wallet

✓ Should change fund wallet to a new wallet

✓ Should fail to burn more tokens than the sender has

✓ Should burn 6969 tokens in the senders account

Contract: Tests for IHFVesting

✓ Should fail to deploy the IHFVesting token with an invalid IHF token address

✓ Should succeed to deploy the IHFVesting token instance (44ms)

✓ Should set IHF's vesting contract address

✓ Should fail to update the future funding block number from an invalid account

✓ Should fail to update the future funding block number to one less than the current block number

✓ Should fail to update the future funding block number from an invalid account

✓ Should update the future funding block number

✓ Should fail to claim the tokens and transfer it to the beneficiary when a different account attempts to claim

✓ Should fail to claim the tokens before the funding block has ended

✓ Should claim the tokens and transfer it to the beneficiary (431ms)

✓ Should fail to claim the 1st stage of funds before the proper date (47ms)

✓ Should claim the 1st stage of funds (87ms)

✓ Should fail to claim the 2nd stage of funds before the proper date

✓ Should claim the 2nd stage of funds (89ms)

✓ Should fail to claim the 3rd stage of funds before the proper date

✓ Should claim the 3rd stage of funds (204ms)

✓ Should fail to claim the 4th stage of funds before the proper date

✓ Should claim the 4th stage of funds (81ms)

✓ Should fail to update the future funding block number after the funding block has passed

✓ Should check the balance of the contract

✓ Should fail to claim other tokens from an unauthorized account

✓ Should fail to claim other tokens from a blank contract address

✓ Should fail to claim IHF tokens using claimOtherTokens

✓ Should claim other tokens (53ms)

✓ Should fail to change the beneficiary from an unauthorized account

✓ Should fail to change the beneficiary to a blank address

✓ Should change the beneficiary

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. All Contract Files Tested

File	Fingerprint (SHA256)
contracts/BasicToken.sol	c361a573a837bb7cc16a7b821e1622c7027fc0d94e366b05a5cf1e0b0e39fe35
contracts/ERC20.sol	6b75acd05c29968b057ec1facf659c064dbe0a79ac01444530629f01ef3a3abf
contracts/ERC20Basic.sol	86c0a5fc6cb564ae77140da57a8ff9a22f46404240e69a6782ff741e286d373a
contracts/IHF.sol	7ded0c6067cf2133c47b54bfd76351095ee6ff97d5934886439b7b0ac50a627d
contracts/IHFVesting.sol	91805da90869aceacc09851febf4ee02d408d368354eb60a4bb6d160526c94a1
contracts/SafeMath.sol	79c36e0aec10ebe744135afa9e0edb27d0a2e2f15cc3a2685169c679154fc05e
contracts/StandardToken.sol	1fb0056e57f58413566e9f10132536b07fb4f4bf01a3563729568d398714fda7

7. Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
contracts/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/IHF.sol	100.00%	97.22%	100.00%	100.00%
contracts/IHFVesting.sol	100.00%	100.00%	100.00%	100.00%
contracts/SafeMath.sol	100.00%	100.00%	100.00%	100.00%
contracts/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
All files	**100.00%**	**98.86%**	**100.00%**	**100.00%**