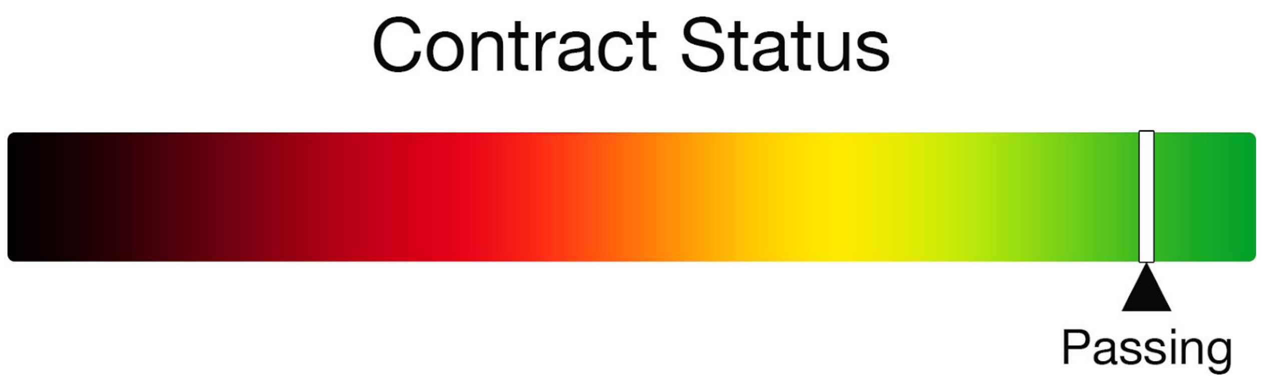
HealthNexus Contract Audit

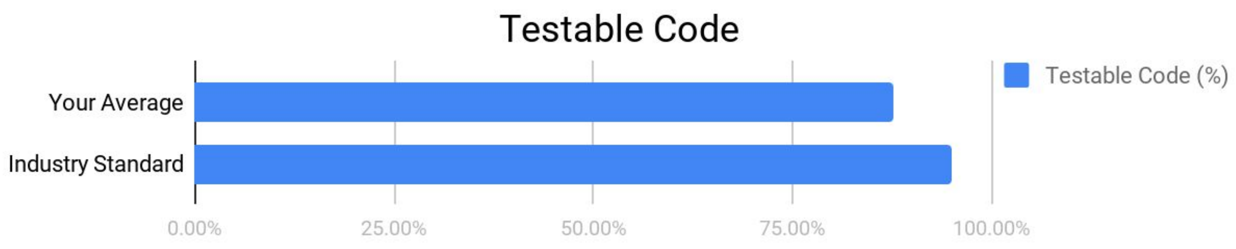
by Hosho, February 2018

Executive Summary

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



All issues have been corrected and the contract is in a passing state. (See Complete Analysis)



Testable code is lower than is standard but this is due to system design and does not indicate any issues. (See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract; it is merely an assessment of its 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, the Hosho Team recommends that the HealthNexus staff 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 an initial review of the code on January 31, 2018 and a thorough secondary review of the code as written and last updated on February 23, 2018. All of the 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 standard 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
- Is not affected by the latest vulnerabilities

The Hosho Team has followed best practices and industry-standard techniques to verify the proper implementation of HealthNexus' token contract. Our staff of expert pentesters and smart contract developers reviewed the contract line by line, documenting any issues as they were discovered. Part of this work included writing a code-specific unit test suite using the Truffle testing framework. As demonstrated, our strategies consist largely of manual collaboration between multiple team members at each stage of the review, including:

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.

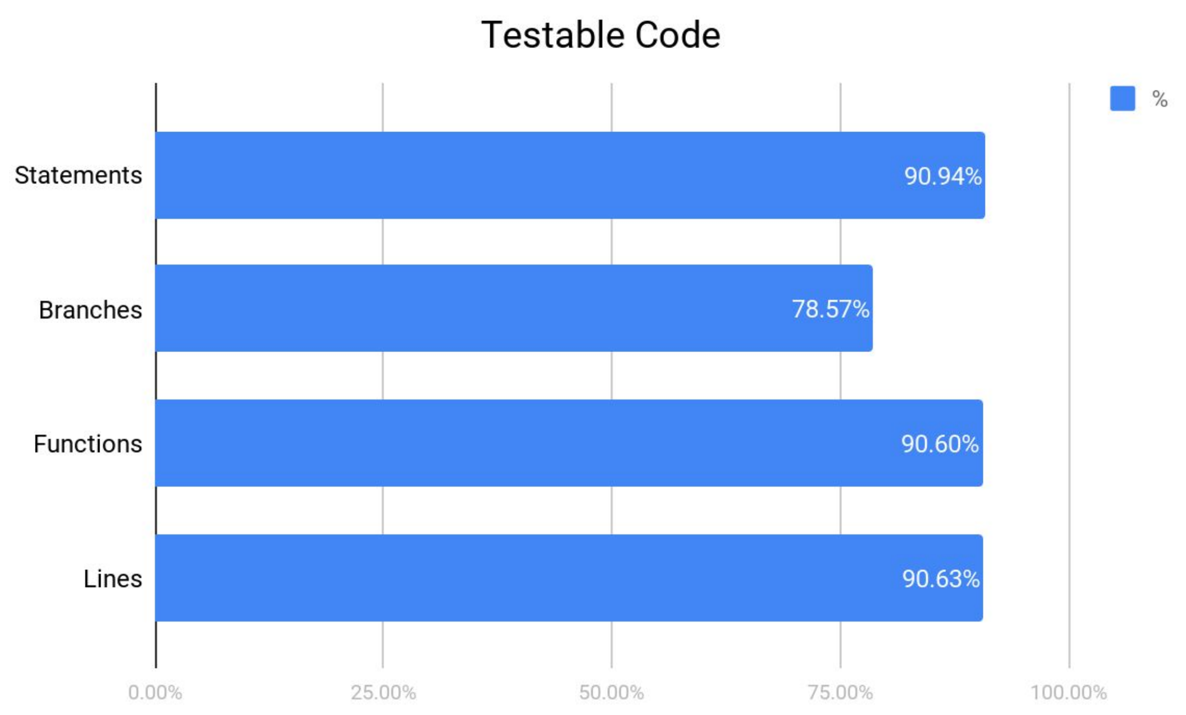
2. Structure Analysis and Test Results

2.1. Summary

The HealthNexus contracts constitute a solid ICO contract, based off of the OpenZeppelin contracts and utilizing TokenMarket ICO contracts. The contracts are well written and highly functional. The code coverage within our testing suite is lower than typical due to the heavy use of `ecrevocer` which requires unique circumstances to validate functionality as well as a few non-reachable statements, a result of overloading functions in parent contracts. All code not covered by test suite received manual verification.

2.2 Coverage Report

As part of our work assisting HealthNexus 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

All failing tests have been corrected.

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:

- **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.

3.1. Resolved, Critical: Token Claim Broken

WhitelistProxyBuyer.sol

Explanation

In the claim function, there is a check that is supposed to ensure that the amount requested is less than the remaining claim amount. The actual code reads as follows:

```
getClaimLeft(investor) <= amount
```

This check is inverted in that it validates that the remaining claim is less than the amount requested as opposed to verifying that the amount requested is less than the remaining claim. Partial withdrawals end up being blocked as well as allowing a user to request more than their allotted claim amount.

Resolution

The check has been updated to properly reflect the intended comparison by the HealthNexus Team.

3.2. Resolved, Low: Crowdsale Address Verification

WhitelistProxyBuyer.sol

Explanation

In the whitelist contract, there is no verification that the set crowdsale address is in fact an address that is part of the crowdsale.

Resolution

The HealthNexus Team added `require(_crowdsale.isCrowdsale());` to ensure the address passed in is a valid crowdsale contract.

---

4. Closing Statement

We are grateful to have been given the opportunity to work with the HealthNexus Team once again.

Overall, the HealthNexus contracts are well written and function as intended. All issues have been corrected and the contracts are now functioning as intended. Code coverage is lower than usual but anything that is not covered by tests was manually verified. As a small team of experts, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that the HealthNexus 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 HealthNexus Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Contract: ERC-20 Tests for HealthCashToken

✓ Should allocate tokens per the minting function, and validate balances (170ms)

✓ Should not transfer until token is released

✓ Should not release token until release agent is set

✓ Should set a release agent for the token (174ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (103ms)

✓ Should not transfer negative token amounts

✓ Should not transfer more tokens than you have

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (49ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (70ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (188ms)

✓ 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 (38ms)

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

✓ Should allow an approval to be set, then increased, and decreased (214ms)

✓ Should burn tokens on the account

✓ Should not let you burn more tokens than you have on the the account

Contract: Crowdsale

Deployment

✓ Should require a token to be set (74ms)

✓ Should require a pricing strategy to be set (67ms)

✓ Should require a receiving wallet to be set (66ms)

✓ Should require a start time > 0 (66ms)

✓ Should require an end time > 0 (75ms)

✓ Should require an end time > start time (66ms)

✓ Should not accept money via its fallback function

```
      ✓ Should be a crowdsale

      ✓ Should return the number of referrers

    Purchasing

      ✓ Should allow the owner to give out tokens (117ms)

      ✓ Should block direct access to the invest function if the signed address is required

      ✓ Should block direct access to the invest function if the signedAddress is required (42ms)

      ✓ Should not let the buyer invest if the contract is pre-funding (140ms)

      ✓ Should not let the buyer invest if the purchase would break the cap

    Management

      ✓ Should let the owner set the end time, if the end time is > now and it's over the starting
time (45ms)

      ✓ Should not let the owner set an end time before now

      ✓ Should not let the owner set an end time before the start time (45ms)

      ✓ Should let the owner set a new start time

      ✓ Should require that a new start time is after the current time

      ✓ Should not allow an start time after the end time

      ✓ Should not let the owner set an end time before the start of the contract (171ms)

      ✓ Should let the owner change the multisig address before 5 investments happen, then stop
them (1571ms)

      ✓ Should report if the minimum goal has been reached (742ms)

      ✓ Should report if the finalizer is sane (50ms)

      ✓ Should report if the pricing is sane

      ✓ Should not allow an invalid finalize agent to be set

      ✓ Should not allow an invalid whitelist to be set

      ✓ Should not permit finalizing until the contract is able to be finalized (44ms)

      ✓ Should return failed if the crowdsale doesn't receive any eth (300ms)

      ✓ Should return refunding if the crowdsale doesn't hit minimums, then is reloaded with eth
(1067ms)

      ✓ Should return success if the crowdsale hits minimums and is over (839ms)

    Refunds

      ✓ Should safely handle refunds on failure, once loaded (1065ms)


  Contract: Airdrop Tests

    Token Issuance

      ✓ Should allow the owner to issue tokens as desired (127ms)

      ✓ Should not allow double issuance to the same address (68ms)


  Contract: Ownership Tests for Crowdsale
```

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 (66ms)

Contract: Crowdsale

Deployment

✓ Should require a token to be set (66ms)

✓ Should require a pricing strategy to be set (68ms)

✓ Should require a receiving wallet to be set (68ms)

✓ Should require a start time > 0 (64ms)

✓ Should require an end time > 0 (69ms)

✓ Should require an end time > start time (66ms)

✓ Should not accept money via its fallback function

Purchasing

✓ Should allow the owner to give out tokens (117ms)

✓ Should block direct access to the invest function if the signed address is required

✓ Should block direct access to the invest function if the signedAddress is required (46ms)

✓ Should not let the buyer invest if the contract is pre-funding (142ms)

✓ Should not let the buyer invest if the purchase would break the cap

Management

✓ Should let the owner set the end time, if the end time is > now and it's over the starting time (66ms)

✓ Should not let the owner set an end time before now

✓ Should not let the owner set an end time before the start of the contract (161ms)

✓ Should let the owner change the multisig address before 5 investments happen, then stop them (1460ms)

✓ Should report if the minimum goal has been reached (752ms)

✓ Should report if the finalizer is sane (45ms)

✓ Should report if the pricing is sane

✓ Should not permit finalizing until the contract is able to be finalized (42ms)

✓ Should return failed if the crowdsale doesn't receive any eth (328ms)

✓ Should return refunding if the crowdsale doesn't hit minimums, then is reloaded with eth (1031ms)

✓ Should return success if the crowdsale hits minimums and is over (753ms)

Refunds

```
    ✓ Should safely handle refunds on failure, once loaded (1093ms)


Contract: FinalizeAgent
  ✓ Should return true for IsFinalizeAgent() call
  ✓ Should return true for IsSane() call (93ms)
  ✓ Should do nothing for finalizeCrowdsale


Contract: TokenFreezer tests for SVH
  Deployment
    ✓ Should require a non 0x0 multisig address
    ✓ Should require a non 0x0 token address
  Usage
    ✓ Should not allow ETH payments to be made to the contract
    ✓ Should not allow a call to the callback function
    ✓ Should not allow unlock until the expected unlock time has passed
    ✓ Should transfer tokens on unlock (358ms)


Contract: Pricing
  ✓ Should verify contract is instantiated correctly (52ms)
  ✓ Should not allow non owner to set preicoAddress
  ✓ Should setPreicoAddress for accounts[0] (74ms)
  ✓ Should return the amount of the first tranche
  ✓ Should return the amount of the last tranche
  ✓ Should return the hardcoded boolean true for isSane
  ✓ Should return the current price based on passed wei param (135ms)
  ✓ Should determine if address is in the presale list
  ✓ Should calculate price for buy in amount (73ms)
  ✓ Should not allow eth to be sent to contract
  ✓ Should report back the correct values for a tranche on getTranche
  Deployment
    ✓ Should not allow deployment with an odd number of tranches
    ✓ Should not allow deployment with more than 2x MAX_TRANCHES tranches (62ms)
    ✓ Should not allow tranche starting amount to decrease in value (38ms)
    ✓ Should require that the first trance amount is 0 (40ms)
    ✓ Should require that the last trance price is 0 (40ms)


Contract: ProxyPurchaser
```

```
    ✓ Should check proxy instantiation (243ms)
    ✓ Should check proxy instantiation (43ms)
    ✓ Should check cs token through proxy (47ms)
    Pausable
      ✓ Should allow the owner to halt the contract (74ms)
      ✓ Should, when paused, deny purchase rights through buyForEverybody if not called by the
owner
      ✓ Should, when paused, deny purchase rights through buy() to everyone
      ✓ Should allow the owner to unpause a contract
      ✓ Should not allow the unpausing of an unpaused contract
    Deployment
      ✓ Should not allow a 0 value for freezeEndsAt (46ms)
      ✓ Should not allow a 0 value for weiMinimumLimit (50ms)
      ✓ Should not allow a 0 value for weiMaximumLimit (51ms)
      ✓ Should not accept funds to the default/fallback function
    Setup
      ✓ Should not return a token if the token is not yet set (54ms)
      ✓ Should not allow a non-crowdsale address to be set as the crowdsale
      ✓ Should allow a new minimum limit to be set
      ✓ Should allow a new maximum limit to be set
      ✓ Should allow a new wei cap to be set
    Usage
      ✓ Should buy tokens (184ms)
      ✓ Should buy tokens with referral data (633ms)
      ✓ Should allow users to be removed from the whitelist
      ✓ Should return the correct token as being used by the crowdsale (43ms)
      ✓ Should allow the contract to take orders via the buy function (138ms)
      ✓ Should allow the contract to take orders via the buyWithCustomerId function (178ms)
      ✓ Should not allow 0 wei orders
      ✓ Should not allow orders in a refunding state (52ms)
      ✓ Should not allow orders in a refunding state (42ms)
      ✓ Should safely handle the same investor investing multiple times (315ms)
      ✓ Should revert if more WEI is sent than permitted by maxWei (132ms)
      ✓ Should revert if more WEI is sent than permitted by weiCap (135ms)
      ✓ Should not allow the contract to determine the claim amount before the distribution has
begun
      ✓ Should not allow the contract to issue refunds in a non-refund state
      ✓ Should not allow the contract to issue refunds if the investors balance is 0 (39ms)
```

✓ Should not allow a non-crowdsale address to be set as the crowdsale address

✓ Should not accept funds to loadRefund, if the contract is not in a refunding state

✓ Should sanely handle the freeze ending, and permit refunds as appropriate (295ms)

✓ Should purchase tokens with funds given to it, and distribute them (360ms)

✓ Should not allow a token purchase if the crowdsale address is not set (138ms)

✓ Should not allow a 0 token purchase (105ms)

✓ Should only allow the purchase from the crowdsale to happen once (383ms)

✓ Should allow claim verification and claiming (1159ms)

✓ Should block claiming of tokens until the timelock is expired (736ms)

✓ Should not issue tokens on 0 amount claims

6. All Contract Files Tested

Original Files January 31, 2018

File	Fingerprint (SHA256)
contracts/AllocatedCrowdsaleMixin.sol	b3087058d0fb5fb35d6e346a200008acb14bf73f62160849e55bd85721c5d6e6
contracts/Crowdsale.sol	24cb47559715279d459aeb6d0680282036746817617789db48ff731ea504767a
contracts/CrowdsaleBase.sol	7bc74120189afe37eaebb3317a5b0de4665649f2960f0befa996e897e2db6497
contracts/FinalizeAgent.sol	cad99a1ca4e65207ad6558f90870edcbd4912ffd6677816c910a015dfc87bf96
contracts/HealthCashToken.sol	355f486ce420018621852d64d86a37a6166acd09891c7b6849c3fb08df7c007c
contracts/Issuer.sol	4e99aaefc788207cd3820100aa7db2a85875cf20fc1c95ba2decb8e172089c9a
contracts/ReleasableToken.sol	11701ccc08f2c1989bbbe098f9e2902f23bc0a5071303623bc191a5c7213cbbf
contracts/ReleasingFinalizeAgent.sol	1c31576ec5a59052458c951b928b4b12401d36227e293688722d16ca0c97ad9b
contracts/TokenFreezer.sol	a3848e28e33e76cf49bf6357fd2e1660c7b66585e2d5e227cbf759a668de393a
contracts/TokenTranchePricing.sol	d955dd7ea34e41d70cc3c69dac518ac67e993817fd01f0dcb2b75b6926aae402
contracts/WhiteListProxyBuyer.sol	6d28f938f4a1a8df430b230de4058b937051d361b506e23ac973c7f99d6177b3
contracts/whitelist.sol	639bd9b62a0f7a4ac60d6a5dc73f4ac823cfc72012c88efdda34782d538f439b
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	78bf21e029fc3f1c38151915db9ccce2f0553bfeae9b6685fde1c297091cdb6f
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	e434336813af116101008bcfaed8cc02fa051c9c2b612a477b6bfa0765fa17f6
contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	35dcf237365077adb1dd8d1da9e05f2b4f8e9d7b49311fc8a09b28d4ce191579
contracts/zeppelin-solidity/contracts/token/BasicToken.sol	7bc1695b8a0ab00d5795d930f9597e0553d8f6ab2e47d87386fbe6488d472bb5
contracts/zeppelin-solidity/contracts/token/BurnableToken.sol	6ae07ae6cc71af27e29af199dbfc2d97f28fcc427398d88cd57eddd456e7a351
contracts/zeppelin-solidity/contracts/token/DetailedERC20.sol	3d5721993d83b727de3066d7c45cf333a6f62eef14c62a2dd82e6a1f0067b7bc

contracts/zeppelin-solidity/contracts/token/ERC20.sol	6b75acd05c29968b057ec1facf659c064dbe0a79ac01444530629f01ef3a3abf
contracts/zeppelin-solidity/contracts/token/ERC20Basic.sol	86c0a5fc6cb564ae77140da57a8ff9a22f46404240e69a6782ff741e286d373a
contracts/zeppelin-solidity/contracts/token/SafeERC20.sol	bde4d7e6d38ac1c64de5327ca306a61c3aea596b1e3ffe46fa21e8ec8b52cdc6
contracts/zeppelin-solidity/contracts/token/StandardToken.sol	77e45da1164753f886d7395987b46deb036eca32c2e7322ef7a2764a08f7c5da
contracts/zeppelin-solidity/contracts/token/TokenTimelock.sol	635f556114738ffbb733cc97e985c15c91a0325a9bc5219c72395af86867033a

Updated Files February 23, 2018

File	**Fingerprint (SHA256)**
contracts/AllocatedCrowdsaleMixin.sol	b3087058d0fb5fb35d6e346a200008acb14bf73f62160849e55bd85721c5d6e6
contracts/Crowdsale.sol	24cb47559715279d459aeb6d068028203674681761 7789db48ff731ea504767a
contracts/CrowdsaleBase.sol	7bc74120189afe37eaebb3317a5b0de4665649f2960f0befa996e897e2db6497
contracts/FinalizeAgent.sol	cad99a1ca4e65207ad6558f90870edcbd4912ffd6677816c910a015dfc87bf96
contracts/HealthCashToken.sol	355f486ce420018621852d64d86a37a6166acd09891c7b6849c3fb08df7c007c
contracts/Issuer.sol	4e99aaefc788207cd3820100aa7db2a85875cf20fc1c95ba2decb8e172089c9a
contracts/ReleasableToken.sol	11701ccc08f2c1989bbbe098f9e2902f23bc0a5071303623bc191a5c7213cbbf
contracts/ReleasingFinalizeAgent.sol	1c31576ec5a59052458c951b928b4b12401d36227e293688722d16ca0c97ad9b
contracts/TokenFreezer.sol	a3848e28e33e76cf49bf6357fd2e1660c7b66585e2d5e227cbf759a668de393a
contracts/TokenTranchePricing.sol	d955dd7ea34e41d70cc3c69dac518ac67e993817fd01f0dcb2b75b6926aae402
contracts/WhiteListProxyBuyer.sol	3c8b3247be57778a25859edea0a32a4072d7c04be6570472d68f309692d51cf4
contracts/whitelist.sol	639bd9b62a0f7a4ac60d6a5dc73f4ac823cfc72012c88efdda34782d538f439b
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	78bf21e029fc3f1c38151915db9ccce2f0553bfeae9b6685fde1c297091cdb6f
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	e434336813af116101008bcfaed8cc02fa051c9c2b612a477b6bfa0765fa17f6

contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	`35dcf237365077adb1dd8d1da9e05f2b4f8e9d7b49311fc8a09b28d4ce191579`
contracts/zeppelin-solidity/contracts/token/BasicToken.sol	`7bc1695b8a0ab00d5795d930f9597e0553d8f6ab2e47d87386fbe6488d472bb5`
contracts/zeppelin-solidity/contracts/token/BurnableToken.sol	`6ae07ae6cc71af27e29af199dbfc2d97f28fcc427398d88cd57eddd456e7a351`
contracts/zeppelin-solidity/contracts/token/DetailedERC20.sol	`3d5721993d83b727de3066d7c45cf333a6f62eef14c62a2dd82e6a1f0067b7bc`
contracts/zeppelin-solidity/contracts/token/ERC20.sol	`6b75acd05c29968b057ec1facf659c064dbe0a79ac01444530629f01ef3a3abf`
contracts/zeppelin-solidity/contracts/token/ERC20Basic.sol	`86c0a5fc6cb564ae77140da57a8ff9a22f46404240e69a6782ff741e286d373a`
contracts/zeppelin-solidity/contracts/token/SafeERC20.sol	`bde4d7e6d38ac1c64de5327ca306a61c3aea596b1e3ffe46fa21e8ec8b52cdc6`
contracts/zeppelin-solidity/contracts/token/StandardToken.sol	`77e45da1164753f886d7395987b46deb036eca32c2e7322ef7a2764a08f7c5da`
contracts/zeppelin-solidity/contracts/token/TokenTimelock.sol	`635f556114738ffbb733cc97e985c15c91a0325a9bc5219c72395af86867033a`

7. Individual File Coverage Report

Original Files January 31, 2018

File	% Statements	% Branches	% Functions	% Lines
contracts/AllocatedCrowdsaleMixin.sol	100.00%	50.00%	100.00%	100.00%
contracts/Crowdsale.sol	72.00%	50.00%	66.67%	69.23%
contracts/CrowdsaleBase.sol	92.41%	77.27%	94.74%	91.89%
contracts/FinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/HealthCashToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/Issuer.sol	100.00%	100.00%	100.00%	100.00%
contracts/ReleasableToken.sol	77.78%	37.50%	87.50%	75.00%
contracts/ReleasingFinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/TokenFreezer.sol	100.00%	100.00%	100.00%	100.00%
contracts/TokenTranchePricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/WhiteListProxyBuyer.sol	100.00%	100.00%	100.00%	100.00%
contracts/whitelist.sol	65.00%	16.67%	75.00%	65.00%
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	83.33%	62.50%	75.00%	83.33%

contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/DetailedERC20.sol	0.00%	100.00%	0.00%	0.00%
contracts/zeppelin-solidity/contracts/token/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/SafeERC20.sol	33.33%	16.67%	33.33%	33.33%
contracts/zeppelin-solidity/contracts/token/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/TokenTimelock.sol	100.00%	83.33%	100.00%	100.00%
All files	**87.65%**	**79.69%**	**85.05%**	**88.14%**

Updated Files February 23, 2018

File	**% Statements**	**% Branches**	**% Functions**	**% Lines**
contracts/AllocatedCrowdsaleMixin.sol	100.00%	50.00%	100.00%	100.00%
contracts/Crowdsale.sol	72.00%	50.00%	66.67%	69.23%
contracts/CrowdsaleBase.sol	92.41%	77.27%	94.74%	91.89%
contracts/FinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/HealthCashToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/Issuer.sol	100.00%	100.00%	100.00%	100.00%
contracts/ReleasableToken.sol	77.78%	37.50%	87.50%	75.00%
contracts/ReleasingFinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/TokenFreezer.sol	100.00%	100.00%	100.00%	100.00%

contracts/TokenTranchePricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/WhiteListProxyBuyer.sol	100.00%	94.23%	100.00%	100.00%
contracts/whitelist.sol	65.00%	16.67%	75.00%	65.00%
contracts/zeppelin-solidity/contracts/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/math/SafeMath.sol	83.33%	62.50%	75.00%	83.33%
contracts/zeppelin-solidity/contracts/ownership/Ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/DetailedERC20.sol	0.00%	100.00%	0.00%	0.00%
contracts/zeppelin-solidity/contracts/token/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/SafeERC20.sol	33.33%	16.67%	33.33%	33.33%
contracts/zeppelin-solidity/contracts/token/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/contracts/token/TokenTimelock.sol	100.00%	83.33%	100.00%	100.00%
All files	**90.94%**	**78.57%**	**90.60%**	**90.63%**