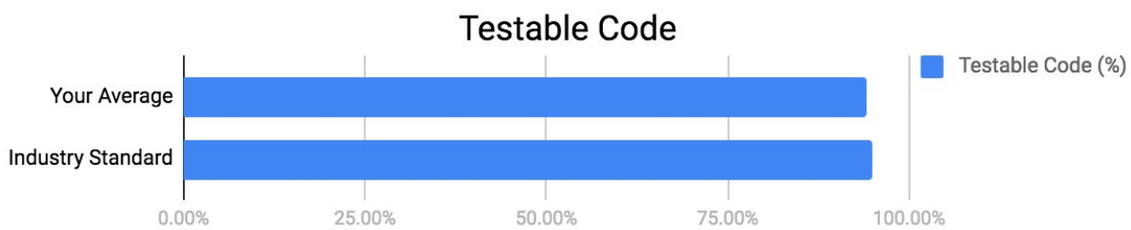
Zikto Contract Audit

Prepared by Hosho
May 28th, 2018

Executive Summary

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

All issues have been remediated by the Zikto team. (See Complete Analysis)



Testable code is 94.03%, which is slightly below industry standard. (See Coverage Report for explanation of low code coverage)

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 Zikto 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 21, 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 Zikto'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.

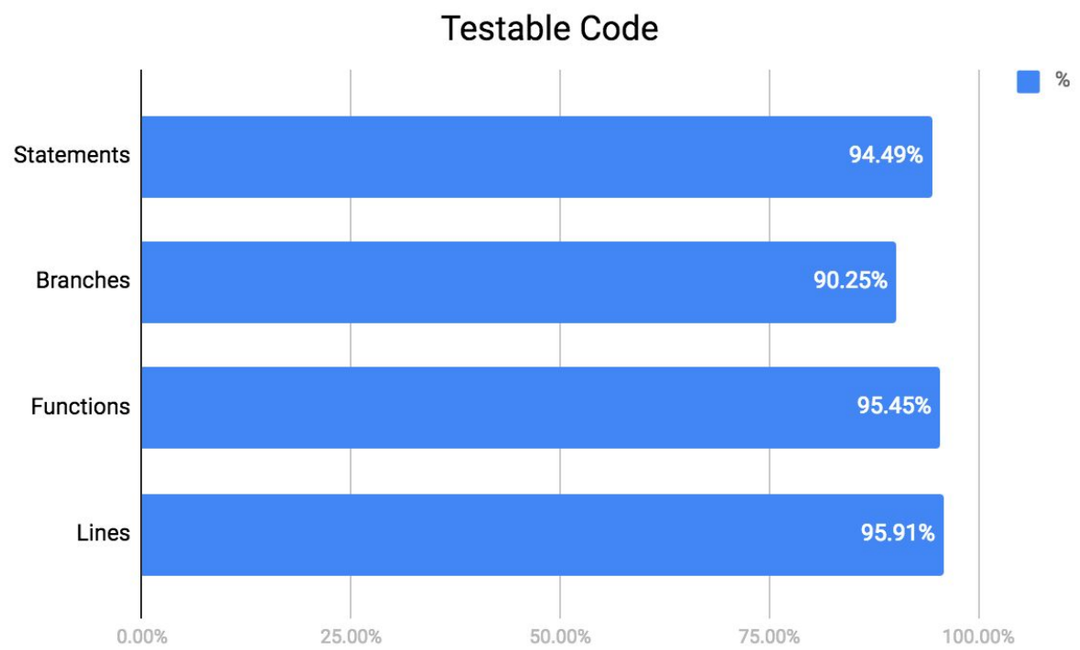
2. Structure Analysis and Test Results

2.1 Summary

Zikto is an ERC20 based contract that utilizes a know your customer (KYC) crowdsale, and has built-in multisig wallet authorization. Zikto uses contracts from the Open-Zeppelin and TokenMarket repositories.

2.2 Coverage Report

As part of our work assisting Zikto 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.

2.4 Testing Notes

In order to test the contracts in a more granular fashion, the Zikto contracts have been disassembled and tested individually to avoid retesting of commonly used code-paths, which are a result of a large amount of reused code. As part of this separation, the Hosho final report will be broken down by the contract name for individual hashes. Exclusions to this are contracts with deeply embedded self-links as a result of modifications of the underlying crowdsale code. For example, the CrowdsaleBase contract was modified for ISRKYCSaleBasic, so it maintains a copy of CrowdsaleBase internally.

Also, due to the high number of mixins and overrides, a large number of uncovered code-paths exist that are impossible to hit. Because of the disassembly mentioned previously, the Hosho Team was able to cover a higher percentage of testable code than would have possible otherwise.

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:

- **Informational** - The issue has no impact on the contract’s ability to operate, and is meant only as additional information.
- **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, High: Incorrect Variable

Contract: AllocatedCrowdsaleMixin.sol

Explanation

The `getTokensLeft` function utilizes owner as the basis of the token count. However, the associated `assignTokens` function utilizes the beneficiary address.

Resolution

Resolved by Zikto team by changing the `getTokensLeft` function to utilize the beneficiary address, rather than the owner.

---

4. Closing Statement

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

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Zikto 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 Zikto Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Appendix A

5.1 Test Suite Results

Coverage Report:

Contract: ERC-20 Tests for AMLToken

✓ Should deploy a token with the proper configuration (115ms)

✓ Should allocate tokens per the minting function, and validate balances (501ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (140ms)

✓ Should not transfer negative token amounts (1659ms)

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

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (57ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (60ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (241ms)

✓ 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 (38ms)

✓ Should allow an approval to be set, then increased, and decreased (268ms)

Contract: Custom ERC-20 tests for Zikto/ISR

✓ Should report that it is a token

✓ Should permit token burning (152ms)

✓ Should not permit burning of more tokens than owned

✓ Should allow ownership transfer (97ms)

✓ Should block transfers until the token is released, or if the sender is locked (544ms)

✓ Should only allow the release agent to release the token

✓ Should not allow a lock address or transfer agent to be set once the release occurs (114ms)

✓ Should allow the owner to reclaim tokens if the token is not released (317ms)

✓ Should allow the token information to be changed (86ms)

✓ Should handle sane deployment systems (434ms)

✓ Should only allow the minting agent to mint tokens (103ms)

Upgradeable Token

✓ Should allow the upgrade master to be set (102ms)

✓ Should allow the upgrade state to be checked (820ms)

✓ Should require that the agent not be set to 0x0

✓ Should require that only the upgradeMaster can set the agent

✓ Should require a positive true return from isUpgradeAgent (68ms)

✓ Should require that the originalSupply variable is the same as the current totalSupply (59ms)

Contract: ISRKYCCrowdsale

✓ Should allow owner to set signer address (43ms)

✓ Should allow early particpate to buy without signature check (458ms)

✓ Should allow signed purchase (819ms)

✓ Should require valid data signature for purchase (190ms)

✓ Should require signed whitelist address to match sender address (176ms)

✓ Should require purchase amount to be greater than signed min eth amount (341ms)

✓ Should require purchase amount to be less than signed max eth amount (350ms)

Contract: ISRKYCSaleBasic

✓ Should require valid deployment parameters (399ms)

✓ Should have required interface markers (65ms)

✓ Should allow owner test value usage

✓ Should report goal not reached without enough raised

✓ Should reject plain send transaction

✓ Should require whitelist for presale purchase (864ms)

✓ Should allow owner to finalize (380ms)

✓ Should require finalizing agent to finalize (180ms)

✓ Should not allow purchase after finalization (384ms)

✓ Should allow changing of the multisig wallet (43ms)

✓ Should allow setting end date (95ms)

✓ Should allow owner to set signer address (239ms)

✓ Should allow early particpate to buy without signature check (539ms)

✓ Should require whitelist for purchase during prefunding even if signed (207ms)

✓ Should allow signed purchase (782ms)

✓ Should require valid data signature for purchase (172ms)

✓ Should require signed whitelist address to match sender address (176ms)

✓ Should require purchase amount to be greater than signed min eth amount (399ms)

✓ Should require purchase amount to be less than signed max eth amount (418ms)

✓ Should require token transfer to succeed for purchase (946ms)

✓ Should require purchases to not exceed sale cap (420ms)

✓ Should allow refunding (1943ms)

Contract: Multisignature - no daily limit

✓ Should allow the addition of an owner, if done by the wallet, through a submitted transaction (929ms)

✓ Should allow the removal of an owner from the system, resetting the requirement if need be. (1205ms)

✓ Should not allow the confirmation of a non-existant transaction

✓ Should not allow double confirmations from a single user (218ms)

✓ Should not allow recovation of a confirmation if the execution is complete (704ms)

✓ Should return the number of confirmations, and whom confirmed it (727ms)

✓ Should allow owner replacement (453ms)

✓ Should not allow duplicate owners (527ms)

✓ Should allow ETH deposits (58ms)

Contract: Pricing Systems

Administration

✓ Should allow transfer of ownership (88ms)

✓ Should report that it is a pricing strategy

✓ Should report that it is sane

✓ Should report presale purchase state

Flat

✓ Should not allow 0 wei per token during deployment

✓ Should return back 1 token per 1000 wei when configured for 1000 wei per token (74ms)

FlatMinMax

✓ Should require sane deployment arguments (67ms)

✓ Should return 0 for transfers less than the minimum configured amount

✓ Should return 0 for transfers over the maximum configured amount

PresalePricing

✓ Should require that a user be whitelisted through the function before they can purchase (55ms)

Milestone Pricing

✓ Should protect against invalid setup vars (143ms)

✓ Should allow preICO addresses to be set (47ms)

✓ Should not accept ether

✓ Should report correct prices and milestones at each time interval (56ms)

✓ Should handle correct pricing at each time interval (117ms)

Contract: Crowdsale

Administration

✓ Should return back the number of tokens approved from the token instance

✓ Should return if the crowdsale is full

✓ Should return if the token is breaking cap (60ms)

✓ Should require sane deployment variables (397ms)

✓ Should allow transfer of ownership (97ms)

✓ Should be able to be halted, and unhalted (99ms)

✓ Should be able to validate sanity (138ms)

✓ Should be able to handle time modifications (100ms)

✓ Should show correct state changes (346ms)

✓ Should allow changing of the multisig wallet (52ms)

Purchasing

Presale

✓ Should not allow purchase unless the address has been added to the whitelist (392ms)

Main Sale

✓ Should allow preallocation of funds (597ms)

✓ Should allow for purchase via signed addresses (870ms)

✓ Should handle customerID purchases (1076ms)

✓ Should block invest sensibly (129ms)

✓ Should handle checksum purchases (407ms)

✓ Should handle failures gracefully (193ms)

✓ Should only allow the multisig wallet changes if there are less than 5 purchases (2382ms)

Contract: KYCPayloadDeserializer

✓ Should slice a KYC payload in the correct format (121ms)

✓ Should slice a KYC presale payload in the correct format (152ms)

Contract: Issuer

✓ Should transfer tokens from the allower to the benefactor (123ms)

6. Appendix B

6.1 All Contract Files Tested

Commit Hash: 08dad3a06a578f00b62150b0f25ee47d18fafbe8

File	Fingerprint (SHA256)
contracts/AMLToken.sol	536b972fd85ca2ef20210570b4ec4b08e1c1e5ee2349478d028756984c6994be
contracts/AllocatedCrowdsale.sol	1ecd1ecadbacec422fd3253d4cdf787b934350b59633658896281fe40cca034b
contracts/FlatMinMaxPricing.sol	0ffc55ac741619f911fcee86befb02e2543f0f0715d89f9525f3fdba5b801659
contracts/FlatPricing.sol	58501e20216fbc71d9bde1b13508c7dc2f5418b570cab2c60fdc8189765009f4
contracts/IRSKYCSaleBasic.sol	5e8fe59fb1eab5537c8694e1d0f771c8862480acbb3319b91111d734e3a1b267
contracts/ISRKYCCrowdsale.sol	85979f14e8e3da5043124912985de0adaed6d5462f70828b79b9debdc67fadcf
contracts/Issuer.sol	1dc45e96d961aae448cb5d1b55e00cac6c752a91a067a75e3e994b130832b76f
contracts/MilestonePricing.sol	ad8e9925ab277cacc68853678530aa3e88fd8bf154c009dfd42bf70d039bbdf4
contracts/MultiSigWallet.sol	a46ce71adac0da8ec1c34c13f3f70719c9500691dcd599dba471dc4b65a51eab
contracts/NullFinalizeAgent.sol	4eb602e2aadab7bb5b752014b5dbec2a7f553899a57f52249669ffee7d276f74
contracts/PresalePricing.sol	6bb0f90f63ddcca14641a1c680afa4c44ccdefca9d6139ee5ac23f98685943b3
contracts/includes/AllocatedCrowdsaleMixin.sol	683e85fd689deb8d451f9062b5325b96ae4a2b259723d227f629d51c0157f44a
contracts/includes/BasicToken.sol	c73c7da67273f983a30236015f4b22c5706b004d370df8ad54b057a4c20df6ef
contracts/includes/BurnableCrowdsaleToken.sol	5f08a39fa5f7fc706c5cb1929469f830c71b1b830cd1b882797bd0b99f369e48
contracts/includes/BurnableToken.sol	624b8f5ac18a63ed74b4b46a0f2451a3831fe558153f32991a419029bcd2d093
contracts/includes/Crowdsale.sol	28d72e73380fc403585babfd541a99cc3ccbcc73a8cf47f0150fe00d3988b80d
contracts/includes/CrowdsaleBase.sol	e82e1d64c3fbce86422e4908ea628938e16e1107e12470ddfd21dac2e4c7d535
contracts/includes/CrowdsaleToken.sol	86df31860797c7a7b6e5b8ff3cca1637b5fddf0dac66befb2ace65d0965dc1b0
contracts/includes/ERC20.sol	875d9914c9807ec1f6836fb4fbcbc481bc32ec772a170c081c58f9c5d5abdfde

contracts/includes/ERC20Basic.sol	ec9e596cf1b29b4639aba482cac8e65991e626cca4cdf574500f15b60ab8f5f0
contracts/includes/FinalizeAgent.sol	10fd55b762d303fe80363c5c7c58161941a560616d3add8ad920b7fcc926bb41
contracts/includes/FractionalERC20.sol	d03fa3b5b8f835be613b8885a50e9673ee07af831a3eb3c7409993d7f8fd2b79
contracts/includes/Haltable.sol	f1d62164473cdaca7ea699ef16f49632b172b6e6e086154fe8dc5b0138be9b5
contracts/includes/KYCPayloadDeserializer.sol	0c2fd50c39360f1b80cf7b590e1290e29d03ec81315f2e965e02394577a03b2d
contracts/includes/MintableToken.sol	096ef754d436f9a335d614b07627b3927407e29eb79bf700f6bed5a0e3ddc9ea
contracts/includes/Ownable.sol	5d2f1683718330bf82f0ac9b68ca85740c34bcc29f39cc8819bd44e290f3b951
contracts/includes/PricingStrategy.sol	c073ea4a55eae067f06c92d6b81c7780f6d5968f05b4b2513b0889f396c33386
contracts/includes/ReleasableToken.sol	f24cf0a580c54a8b5fea904d7a31e6a862d4494c51b193b1ce51d4f6458e6321
contracts/includes/SafeMath.sol	58488973a1ffc5d5503b1b0c422b4dec61dfc404b8e8452c297d342ff037f884
contracts/includes/SafeMathLib.sol	33ce68e0ce9f080d6a5d0bc1c9eacefc4eba2aa05628563e11e8dde834d0efd0
contracts/includes/StandardToken.sol	38a8e315573f724075c8eff73720f4c92e02b41016a245177fee802afacf3c78
contracts/includes/StandardTokenExt.sol	515fcec6bfb50c18083de634bffc9df19adb90ea1f4afa5907461a2835621e32
contracts/includes/UpgradeableToken.sol	e848b460d64012eef7f6933bcdcaf5bb96e79cbd496551878b70cb77c4314ae7

7. Appendix C

7.1 Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
contracts/AMLToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/AllocatedCrowdsale.sol	100.00%	100.00%	100.00%	100.00%
contracts/FlatMinMaxPricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/FlatPricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/IRSKYCSaleBasic.sol	93.04%	87.18%	100.00%	96.00%
contracts/ISRKYCCrowdsale.sol	100.00%	100.00%	100.00%	100.00%
contracts/Issuer.sol	100.00%	100.00%	100.00%	100.00%
contracts/MilestonePricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/MultiSigWallet.sol	100.00%	97.73%	100.00%	100.00%
contracts/NullFinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/PresalePricing.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/AllocatedCrowdsaleMixin.sol	87.50%	75.00%	100.00%	100.00%
contracts/includes/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/BurnableCrowdsaleToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/Crowdsale.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/CrowdsaleBase.sol	84.81%	81.48%	81.25%	87.88%
contracts/includes/CrowdsaleToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/ERC20.sol	100.00%	100.00%	100.00%	100.00%

contracts/includes/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/FinalizeAgent.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/FractionalERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/Haltable.sol	62.5%	50.00%	80.00%	75.00%
contracts/includes/KYCPayloadDeserializer.sol	88.46%	100.00%	85.71%	87.10%
contracts/includes/MintableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/Ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/PricingStrategy.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/ReleasableToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/SafeMath.sol	83.33%	75.00%	75.00%	83.33%
contracts/includes/SafeMathLib.sol	75.00%	33.33%	66.67%	75.00%
contracts/includes/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/StandardTokenExt.sol	100.00%	100.00%	100.00%	100.00%
contracts/includes/UpgradeableToken.sol	97.30%	100.00%	83.33%	95.83%
All files	**94.49%**	**90.25%**	**95.45%**	**95.91%**