KATX/KATM Token Contract Audit

by Hosho, November 2017

Table of Contents

Technical Summary

This document outlines the overall security of KATM's smart contract as evaluated by Hosho's Smart Contract auditing team.

The scope of this audit was to analyze and document KATM's codebase for quality, security, and correctness.

The KATX/KATM Crowdsale/Token systems are made up of 6 individual contracts, laid out as follows:

1. KATMCrowdsale.sol - Crowdsale for KATM, non-utility token
2. KATXCrowdsale.sol - Crowdsale for KATX, utility token
3. KATMToken.sol - Token for KATM, non-utility token
4. KATXToken.sol - Token for KATX, utility token
5. KATMTokenChanger.sol - Conversion system between KATX/KATM
6. Whitelist.sol - Whitelist manager for purchase management and control

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, 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, we at Hosho recommend that the KATM Team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Auditing Strategy and Techniques Applied

The Hosho Team has performed a thorough review of the smart contract code as written and last updated on November 18, 2017. The following contract files and their respective SHA256 fingerprints were evaluated:

File	Fingerprint (SHA256)
infrastructure/authentication/whitelist/Whitelist.sol	`1d8031d3a590994630d80014de60ac512e3a1b391fce18a80d450a73454bbaa0`
infrastructure/behaviour/Observable.sol	`73e35f93d0fd67dd045fe1bdd28d74e352e818d986e906df1898e20e2b2de768`
infrastructure/modifier/InputValidator.sol	`d73361152f92d94cf0b30363ab892cf8ae533551ac287b93cfd6ea790bf6079d`
infrastructure/ownership/MultiOwned.sol	`199c2086344b0f1ef69a4028167ac1ec0cf13f1ff1075384e44f557a18049c22`
infrastructure/ownership/Ownership.sol	`eda033ed2df78b99b6b4bb169f973801cf0ade35b781a45382e98244034f191d`

infrastructure/ownership/TransferableOwnership.sol	19845764245e86ed64182ea636d62150489558c8f2e2442623c2821d72c92526
source/KATMCrowdsale.sol	aa8853f3ed7469c63b7f75cdd13ab16bbf3c706e151da351ffa1bf8d87949cc9
source/KATMToken.sol	cfb02c07a9264796ce85866e0a8260b964d01000d93edead146e9e5b26c67772
source/KATMTokenChanger.sol	6e02e9969f0d22b5e518aa56f30e6f72d2fbbe749e6aebdceaef2eb450858c11
source/KATXCrowdsale.sol	ad7ad2068b1e85c60911dcf98c64a318a3b6903d4bfb725e61ba5bf761585d6d
source/KATXToken.sol	5f1cffc7a331b4ec64cd2fb04535029ef81ed9f154d5e8577570e03a8e892f70
source/crowdsale/Crowdsale.sol	4e0a170c76855a5c76ee742798df126fed18dad7053f67f4196d87daf34dc25a
source/token/ManagedToken.sol	118c97ff4565a6990a54cdf2a33b4e2eb436542996c18ea062103a8c5564d940
source/token/Token.sol	6659ac4f032948c317ebed6103168072d6518b28d7d6fd2adef104bc9fe30ccc
source/token/changer/TokenChanger.sol	4cc6d45d1b6d6d25c3550b723aab5a6f1ef7586af51333596d4a8712c5ac9b97
source/token/observer/TokenObserver.sol	6199172cf54913f23f4898fc49cfdfb80312782ce6a91501b55e90024e4dac60
source/token/retriever/TokenRetriever.sol	43c3dc8b9a586ca1230089df6107dfad5900a4067dad9438c0d7912212617cb6

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

- Implements and adheres to existing ERC20 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; and
- Uses methods safe from reentrance attacks.

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of KATM'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.

Contract Analysis and Test Results

Summary

The Hosho Team is pleased to report that all contracts are very well written and, as of the creation of this report, are also bug free.The tokens meet ERC-20 standards for security and functionality, including all calculations performed by the contracts are entirely accurate when compared with Hosho's independent testing. Additional functionality above the standard requirements have been added, such as the self-destruct feature to correct accidental token transmission, making these exceptionally well-written contracts.

Coverage Report

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

The small amount left uncovered in branches were tested manually if possible using time-shifting techniques or contract-to-contract interactions. Some lines were inaccessible because they were failsafes for highly unlikely scenarios.

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

File	% Statements	% Branches	% Functions	% Lines
infrastructure/authentication/whitelist/Whitelist.sol	100.00%	100.00%	100.00%	100.00%
infrastructure/behaviour/Observable.sol	100.00%	100.00%	100.00%	100.00%
infrastructure/modifier/InputValidator.sol	100.00%	50.00%	100.00%	100.00%
infrastructure/ownership/MultiOwned.sol	100.00%	100.00%	100.00%	100.00%
infrastructure/ownership/Ownership.sol	100.00%	100.00%	100.00%	100.00%
infrastructure/ownership/TransferableOwnership.sol	100.00%	100.00%	100.00%	100.00%

source/KATMCrowdsale.sol	100.00%	100.00%	100.00%	100.00%
source/KATMToken.sol	100.00%	100.00%	100.00%	100.00%
source/KATMTokenChanger.sol	100.00%	83.33%	100.00%	100.00%
source/KATXCrowdsale.sol	100.00%	100.00%	100.00%	100.00%
source/KATXToken.sol	100.00%	100.00%	100.00%	100.00%
source/crowdsale/Crowdsale.sol	96.13%	86.67	100.00%	96.26%
source/token/ManagedToken.sol	100.00%	75.00%	100.00%	100.00%
source/token/Token.sol	100.00%	80.00%	100.00%	100.00%
source/token/changer/TokenChanger.sol	100.00%	75.00%	100.00%	100.00%
source/token/observer/TokenObserver.sol	100.00%	100.00%	100.00%	100.00%
source/token/retriever/TokenRetriever.sol	100.00%	100.00%	100.00%	100.00%
All Files	**98.02%**	**86.59%**	**100.00%**	**98.17%**

Test Results

Contract: ERC-20 Compliant Token

✓ Should deploy with KATM Security as the name of the token

✓ Should deploy with KATM as the symbol of the token

✓ Should deploy with 8 decimals

✓ Should deploy with 0 tokens

✓ Should allocate tokens per the minting function, and validate balances (164ms)

✓ Should transfer tokens from 0x1bbb1269032bfd0b0fe0851235fc798af6bd3c9b to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (75ms)

✓ Should not transfer negative token amounts

✓ Should not transfer more tokens than you have

✓ Should allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (38ms)

✓ Should not allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer an additional 1000 tokens once authorized, and authorization balance is > 0

✓ Should allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (70ms)

✓ Should allow 0xdaef8d8c30eeb858b8c774a8d7d5e92a552bb0d9 to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (131ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0xdaef8d8c30eeb858b8c774a8d7d5e92a552bb0d9

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0xdaef8d8c30eeb858b8c774a8d7d5e92a552bb0d9

✓ Should allow the token owner to retrieve tokens (123ms)

✓ Should not accept ETH

Contract: ERC-20 Compliant Token

✓ Should deploy with KATM Utility as the name of the token

✓ Should deploy with KATX as the symbol of the token

✓ Should deploy with 8 decimals

✓ Should deploy with 0 tokens

✓ Should allocate tokens per the minting function, and validate balances (174ms)

✓ Should transfer tokens from 0x1bbb1269032bfd0b0fe0851235fc798af6bd3c9b to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (66ms)

✓ Should not transfer negative token amounts

✓ Should not transfer more tokens than you have

✓ Should allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (42ms)

✓ Should not allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer an additional 1000 tokens once authorized, and authorization balance is > 0

✓ Should allow 0x3b44fa9f7511113a8c1a1528070d45b1d7cdd101 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (44ms)

✓ Should allow 0xdaef8d8c30eeb858b8c774a8d7d5e92a552bb0d9 to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (123ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0xdaef8d8c30eeb858b8c774a8d7d5e92a552bb0d9

✓ Should allow the token owner to retrieve tokens (120ms)

✓ Should not accept ETH

Contract: ERC-20 Compliant Token

✓ Should allocate tokens per the minting function, and validate balances (166ms)

✓ Should burn tokens from an owned account as expected (64ms)

✓ Should not burn more tokens than you have

✓ Should allow the token to be locked by an owner (41ms)

✓ Should not allow the token to be unlocked by a non-owner

✓ Should disallow transfers while locked

✓ Should allow the token to be unlocked by an owner (67ms)

✓ Should not allow the token to be locked by a non-owner

Contract: Observable

✓ Should not allow a non-owner to add an observer

✓ Should not allow a non-owner to remove an observer

✓ Should allow an owner to add an observer, but only on the contract it belongs to (83ms)

✓ Should allow an owner to remove an observer, but only on the contract it belongs to-KATM (110ms)

✓ Should allow an owner to remove an observer, but only on the contract it belongs to (KATX) (126ms)

✓ Should allow you to retrieve an observer at the numerical index

✓ Should not allow double adding/removing an observer (128ms)

Contract: Ownership

✓ Should return if someone is an owner or not - KATM (48ms)

✓ Should return if someone is an owner or not - KATX

✓ Should let an owner add a new owner - KATX (81ms)

✓ Should allow an owner to be removed - KATX (65ms)

✓ Should allow not an owner to be removed twice - KATX

Contract: KATM Whitelist

✓ Should allow someone to be added to the whitelist (54ms)

✓ Should allow someone to be verified against the whitelist

✓ Should not allow a non-owner to whitelist someone

✓ Should allow someone to be removed from the whitelist

✓ Should allow someone to be removed from the whitelist even if they aren't on it

✓ Should not allow a non-owner to remove a whitelist

✓ Should allow someone to be re-added to the whitelist (66ms)

✓ Should allow ownership to be transferred

✓ Should confirm if someone is an owner

Contract: KATX<->KATM Token Changer

✓ Should initialize with KATM as the left, and KATX as the right

✓ Should initialize with the correct settings for KATM -> KATX (46ms)

✓ Should only let the owner pause and unpause (40ms)

✓ Should only let the owner enable and disable authentication

✓ Should not accept ETH

✓ Should not allow anyone but the owner to retrieve tokens

✓ Should allow the owner to retrieve tokens (186ms)

✓ Should not allow the owner to retrieve left side tokens

✓ Should not allow a conversion if authentication is required, and the account is not whitelisted (111ms)

✓ Should not transfer if there is a pause on the token (134ms)

✓ Should transfer tokens properly KATM -> KATX (234ms)

✓ Should transfer tokens properly KATX -> KATM (217ms)

✓ Should `transferFrom` tokens properly KATM -> KATX (249ms)

✓ Should `transferFrom` tokens properly KATX -> KATM (265ms)

✓ Should not permit token exchanges of 0 (174ms)

✓ Should be able to transfer the left token back to the owner via retrieve (40ms)

✓ Should not be able to setup whitelist once the contract state is deployed

✓ Should require that the caller of `notifyTokensReceived` be one of the two tokens

Contract: KATX/KATM Crowdsale Functionality

✓ Should instantiate contract with expected setup variables - KATM (920ms)

✓ Should instantiate contract with `getRate` of 0 - KATM

✓ Should not allow destroy before the 2 year period is up - KATM

✓ Should instantiate contract with expected `setupPhases` variables - KATM (54ms)

✓ Should instantiate contract with expected stakeholders - KATM

✓ Should only let the owner enable and disable authentication - KATM

✓ Should let the owner enable and disable authentication - KATM (73ms)

✓ Should not allow anyone but the owner to retrieve tokens - KATM

✓ Should allow the owner to retrieve tokens - KATM (230ms)

✓ Should not allow funds to be sent before the presale starts - KATM

✓ Should only let the owner enable and disable authentication - KATX

✓ Should let the owner enable and disable authentication - KATX (101ms)

✓ Should not allow anyone but the owner to retrieve tokens - KATX

✓ Should allow the owner to retrieve tokens - KATX (201ms)

✓ Should not allow funds to be sent before the presale starts - KATX

✓ Should return proper boolean for `hasBalance` - KATM

✓ Should return proper `uint` for ETH balance - KATM

✓ Should return proper boolean for `refundableEthBalanceOf` - KATM

✓ Should return token amount - KATM

✓ Should not allow a non beneficiary to call `confirmBeneficiary` to start crowdsale - KATM

✓ Should allow beneficiary to call `confirmBeneficiary` to start crowdsale - KATM

✓ Should instantiate contract with expected phase - KATM

✓ Should check for presale phase status - KATM

✓ Should allow someone to be added to the whitelist (48ms)

✓ Should allow funds to be sent after the presale starts - KATM (218ms)

✓ Should allow funds to be sent after the presale starts with contribute - KATM (208ms)

✓ Should allow funds to be sent after the presale starts with contribute for DCORP member - KATM (214ms)

✓ Should not allow funds under the minimum amount during presale - KATM (38ms)

✓ Should not allow funds under the minimum amount during presale - KATX

✓ Should properly return accepted members - KATM

✓ Should return the total amount of ETH raised - KATM

✓ Should check for presale phase status - KATX (42ms)

✓ Should allow funds to be sent after the presale starts with contribute for DCORP member - KATX (234ms)

✓ Should properly return accepted members - KATX (54ms)

✓ Should return the total amount of ETH raised - KATX

✓ Should not withdraw tokens if balances are 0 - KATX (132ms)

✓ Should allow funds to be sent after the presale starts with contribute - KATM (3190ms)

✓ Should allow funds over the minimum deposit to be made during presale - KATX (211ms)

✓ Should not allow the crowdsale to be ended before the end of the crowdsale or the max amount is hit

✓ Should allow whitelisting for further testing (511ms)

✓ Should not allow refund before the crowdsale is done - KATM

✓ Should allow funds to be sent after the main sale starts with contribute for DCORP member - KATX (388ms)

✓ Should allow time acceleration (266ms)

✓ Should check for presale phase status - KATM

✓ Should not accept funds less than the approved value during main round - KATM

✓ Should not accept funds from non-whitelisted addresses - KATM

✓ Should issue the correct number of tokens in the phase (737ms)

✓ Should check that sale is not ended - KATM

✓ Should instantiate contract with expected phase - KATM

✓ Should allow time acceleration (269ms)

✓ Should instantiate contract with expected phase - KATM

✓ Should issue the correct number of tokens in the phase (42ms)

✓ Should allow time acceleration (229ms)

✓ Should instantiate contract with expected phase - KATM

✓ Should issue the correct number of tokens in the phase (40ms)

✓ Should allow time acceleration (275ms)

✓ Should instantiate contract with expected phase - KATM

✓ Should issue the correct number of tokens in the phase

✓ Should allow time acceleration (259ms)

✓ Should instantiate contract with expected phase - KATM

✓ Should issue the correct number of tokens in the phase (42ms)

✓ Should allow time acceleration (264ms)

✓ Should check that stage is set to Ended - KATM

✓ Should instantiate contract with expected phase - KATM

✓ Should not allow refunds when the crowdsale is over it's `minAmount` - KATM (41ms)

✓ Should not accept funds after the end of the crowdsale

✓ Should end crowdsale with `endCrowdsale` function - KATM (264ms)

✓ Should allow time acceleration (256ms)

✓ Should withdraw tokens based on time related to `releaseDate` - KATM (205ms)

✓ Should safely end up with no sends once the withdrawal of tokens and ETH are complete - KATM (174ms)

✓ Should allow refunds for contracts that did not hit limit - KATX (62ms)

✓ Should not allow the crowdsale to be ended if the raise is not over the minimum amounts

✓ Should not refunds for contracts that did not hit limit with 0 balance on accounts - KATX (53ms)

✓ Should allow time acceleration, then destruction (311ms)

Structure and Organization of Document

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.

Complete Analysis

No issues to report.

Closing Statement

We are grateful to have been given the opportunity to work with the KATM and Frank Bonnet, their smart contract developer. Overall the Hosho team is pleased with the quality and overall design of the contract.

As a small team of experts, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that the KATM contracts are 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 KATM Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.