HOSHO

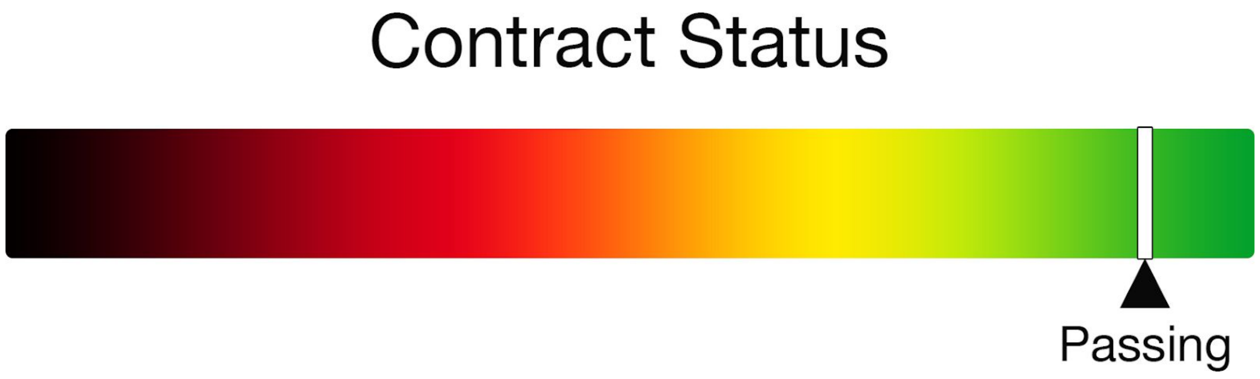
Menlo Contract Audit

Prepared by Hosho
July 9th, 2018

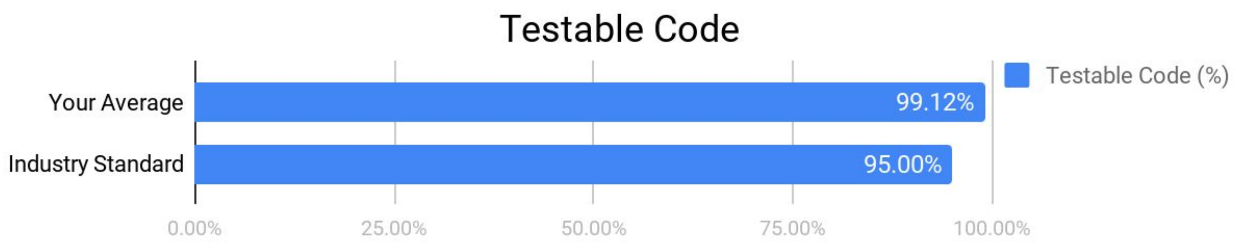
Report Version: 1.0

Executive Summary

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



These contracts have passed the rigorous auditing process performed by the Hosho team. (See Complete Analysis)



Testable code is 99.12% which is above the 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 Menlo 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 30th, 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 Menlo's token contract. To do so, the code is reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they are discovered. Part of this work includes 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.

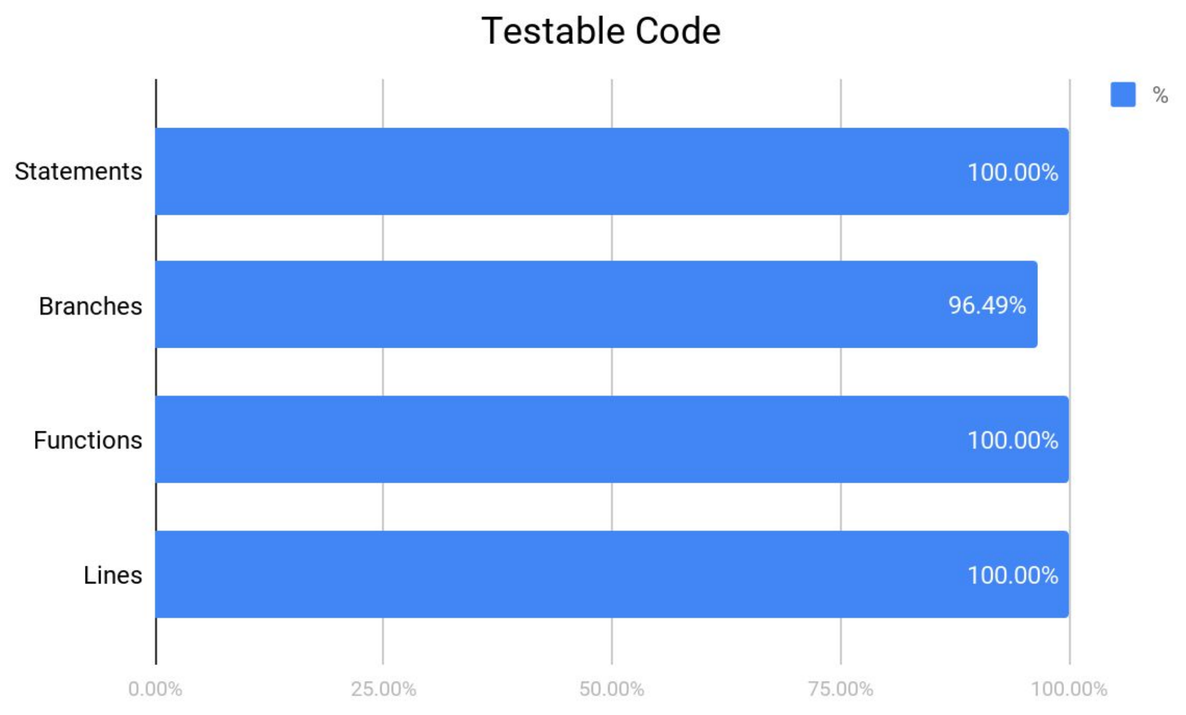
2. Structure Analysis and Test Results

2.1 Summary

The Menlo contracts consist of a standard ERC-20 token with crowdsale and presale features along with a token vault and timelocks. The crowdsale and presale are based on a sale library developed internally with support for a time-based tranche system that changes the bonuses weekly. The Menlo team has also included additional token functionality for pausing and burning tokens with a system to reclaim the tokens, if necessary.

2.2 Coverage Report

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



For each file see Individual File 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 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:

- **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, and is meant only as additional information.

No issues were discovered in the Menlo contracts during the audit process.

---

4. Closing Statement

The Hosho team is grateful to have been given the opportunity to work with the Menlo team.

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

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 Menlo 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

<u>Contract: Function Tests for MenloToken</u>

initializePresale() Tests

√ Should initialize a presale (227ms)

√ Should fail to initialize a presale after it's already been initialized (272ms)

initializeCrowdsale() Tests

√ Should initialize a crowdsale (245ms)

√ Should fail to initialize a crowdsale after it has already been initialized (296ms)

withdraw() Tests

√ Should run the `withdraw` function (59ms)

<u>Contract: Token Reclaim Tests for MenloToken</u>

Token reclaim

√ Should allow the reclaim of tokens from the tested contract (143ms)

√ Should safely handle a failed token transfer (154ms)

<u>Contract: ERC-20 Tests for MenloToken</u>

√ Should deploy a token with the proper configuration (52ms)

√ Should allocate tokens per the minting function, and validate balances (52ms)

√ Should `transfer` tokens from *0xd86543882b609b1791d39e77f0efc748dfff7dff* to *0x42adbad92ed3e86db13e4f6380223f36df9980ef* (170ms)

√ Should not `transfer` negative token amounts (65ms)

√ Should not `transfer` more tokens than you have (68ms)

√ Should allow *0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3* to authorize *0x341106cb00828c87cd3ac0de55eda7255e04933f* to `transfer` 1000 tokens (92ms)

√ Should allow *0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3* to zero out the *0x341106cb00828c87cd3ac0de55eda7255e04933f* authorization (171ms)

√ Should allow *0x667632a620d245b062c0c83c9749c9bfadf84e3b* to authorize *0x53353ef6da4bbb18d242b53a17f7a976265878d5* for 1000 token spend, and *0x53353ef6da4bbb18d242b53a17f7a976265878d5* should be able to send these tokens to *0x341106cb00828c87cd3ac0de55eda7255e04933f* (413ms)

√ Should not allow *0x53353ef6da4bbb18d242b53a17f7a976265878d5* to `transfer` negative tokens from *0x667632a620d245b062c0c83c9749c9bfadf84e3b* (140ms)

√ Should not allow *0x53353ef6da4bbb18d242b53a17f7a976265878d5* to `transfer` tokens from *0x667632a620d245b062c0c83c9749c9bfadf84e3b* to 0x0 (134ms)

√ Should not `transfer` tokens to *0x0* (55ms)

√ Should not allow *0x53353ef6da4bbb18d242b53a17f7a976265878d5* to `transfer` more tokens than authorized from *0x667632a620d245b062c0c83c9749c9bfadf84e3b* (68ms)

√ Should allow an approval to be set, then increased, and decreased (413ms)

<u>Contract: Pause Tests for MenloToken</u>

Deployment

√ Should deploy in an un-paused state

Pause configuration

√ Should be able to be paused (94ms)

√ Should be able to be unpaused (174ms)

√ Should not be able to be unpaused while unpaused (52ms)

√ Should not be able to be paused while paused (123ms)

<u>Contract: BurnableToken Tests for MenloToken</u>

√ Should allow owners to `burn` own tokens (142ms)

√ Should require token burn amount to be less than or equal to balance (48ms)

√ Should require logging burn event (121ms)

<u>Contract: Ownership Tests for MenloToken</u>

Deployment

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

Transfer Ownership

√ Should not allow a non-owner to transfer ownership (53ms)

√ Should not allow the owner to transfer to *0x0* (68ms)

√ Should allow the owner to transfer ownership (99ms)

√ Should allow the owner to renounce ownership (65ms)

<u>Contract: MenloTokenTimelock Tests</u>

Constructor() Tests

√ Should fail to initialize with a release date before now (54ms)

Deposit() Tests

√ Should fail to `deposit` from a non-presale address

Release() Tests

√ Should release tokens (1162ms)

√ Should fail to release tokens before unlock time (1011ms)

√ Should fail to release tokens to an account that has none (1130ms)

√ Should fail to release tokens from empty Timelock (75ms)

<u>Contract: SafeMath</u>

√ 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

<u>Contract: SafeERC20 Tests</u>

Failure Mode Tests

√ Should revert on `transfer` failure (43ms)

√ Should revert on `transferFrom` failure (46ms)

√ Should revert on `approve` failure (50ms)

Success Mode Tests

√ Should pass on `transfer` success (51ms)

√ Should pass on `transferFrom` success (51ms)

√ Should pass on `approve` success (57ms)

<u>Contract: MenloTokenTimelock Tests</u>

Function Tests

√ Should `setToken` (56ms)

√ Should `release` (350ms)

√ Should fail to `release` before `releaseTime` (204ms)

√ Should fail to `release` 0 tokens (144ms)

Constructor() Tests

√ Should fail to initialize with a release date before now (57ms)

<u>Contract: MenloTokenPresale Tests</u>

setTokenTimeLock() Tests

√ Should set the token timelock address (54ms)

calculateBonusRate() Tests

√ Should calculate bonus rate

fallback purchase Tests

√ Should purchase some tokens (932ms)

<u>Contract: MenloTokenSale Tests</u>

calculateBonusRate() Tests

√ Should calculate bonus rate for hour 1 (50ms)

√ Should calculate bonus rate for week 1 (60ms)

√ Should calculate bonus rate for week 2 (69ms)

√ Should calculate bonus rate for week 3 (61ms)

√ Should calculate bonus rate for week 4 (70ms)

√ Should calculate bonus rate for week 5 (71ms)

√ Should calculate bonus rate after week 5 (68ms)

fallback purchase Tests

√ Should purchase some tokens (923ms)

<u>Contract: MenloSaleBase Tests</u>

Constructor Tests

√ Should fail to construct without valid start time (99ms)

√ Should fail to construct without valid end time (107ms)

√ Should fail to construct without valid wallet (111ms)

√ Should fail to construct without valid cap (128ms)

√ Should fail to construct without valid token (125ms)

purchase failure Tests

√ Should purchase some tokens (816ms)

√ Should purchase the remaining tokens when given more eth than cap (880ms)

√ Should fail to purchase from unwhitelisted account (319ms)

√ Should fail to purchase outside sale time (412ms)

√ Should fail to purchase with no eth (441ms)

√ Should fail to purchase from an empty contract (267ms)

√ Should fail to purchase after finalized (603ms)

Other function Tests

√ Should `withdraw` (52ms)

√ Should fail to finalize when already finalized (393ms)

√ Should `refund` tokens (592ms)

√ Should fail to `refund` tokens before finalized (306ms)

√ Should return false when refunding 0 tokens (659ms)

√ Should fail to `refund` tokens on invalid transfer (463ms)

√ Should fail to whitelist by someone that isn't the whitelister (44ms)

√ Should do nothing when whitelisting an account that has already been whitelisted (184ms)

√ Should fail to initialize a second presale tokens (754ms)

6. Appendix B

All Contract Files Tested

Commit Hash: 5a23c2d6fde543f729fe78d297eab9ba7895f673

File	Fingerprint (SHA256)
MenloSaleBase.sol	a69a49041c8affcf2aa2e11be96837061af78e3c55d41d868836260641995426
MenloToken.sol	621efb2c3c7c48f1851d2dca4d484fd504f465a1abdb5d14b67c5fe14da2979f
MenloTokenPresale.sol	f36f25a95356d873e099a36dc7b0d77214555631e63ff94bec23c365bfb9e3c7
MenloTokenSale.sol	c168e7e9b41603461cd3ca1f67aae1054c64bba1623671fd3bb0084ed7d5e94f
MenloTokenTimelock.sol	72b5e4b89a770801c26b60f5ac83bf2c28f4e2cca6977bc6e638c1216dbdca85
MenloWalletTimelock.sol	90f494c8140d17891e7c6333bfb9d8c491c0b9f75b6128cc5c6f99108bca9073
zeppelin-solidity/lifecycle/Pausable.sol	1651888c7d9d07418c3da274e3598dca7686875263ee61ad200f3e3db67b518a
zeppelin-solidity/math/SafeMath.sol	6966b7304174d89563770ab3b56e06aa3e20ea066b14ca84a57e13ec10a749fe
zeppelin-solidity/ownership/CanReclaimToken.sol	7ff0fa4811f7e779706e2c5f8c3f95bf49878797904d3da11d5b491e578412af
zeppelin-solidity/ownership/Ownable.sol	a4bd745530feb0c271757551370d047235e26fe79da2fa260f3b81715cd0a147
zeppelin-solidity/token/ERC20/BasicToken.sol	2ce519dcadb6455185e1478ddeb47520a7a00f6f311f69969f6ac00315f66206
zeppelin-solidity/token/ERC20/BurnableToken.sol	d69681a20a06fd3af9a6a5c317b3638930d6e688486df69c96fea4e00130f450
zeppelin-solidity/token/ERC20/ERC20.sol	ed00fe45c0deef6a6f741c1dc57c4b5d4754404e02a3ea36da2fad8157cf4e1f
zeppelin-solidity/token/ERC20/ERC20Basic.sol	7d99160795719766f3dc95d53b24c87ac6a0235429992ad63eccd48be34241cb
zeppelin-solidity/token/ERC20/PausableToken.sol	83a3059f9126e6f3ba7ccd5567b6337142011ba3863cf49c45103755e3b5658e
zeppelin-solidity/token/ERC20/SafeERC20.sol	f475424814b282b1c2f5799bfa6d66a30527d55c7090ef6a11cbeb421f6ef06c
zeppelin-solidity/token/ERC20/StandardToken.sol	66d9d19928143a013b093e56addd3641485b662b108ae10ea8f51bcc9912ca0c
zeppelin-solidity/token/ERC20/TokenTimeLock.sol	302a09eb6af84fd977b0440c36deff18261898e4ef0ccf4a32c00e166d0afa4d

7. Appendix C

Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
MenloSaleBase.sol	100.00%	100.00%	100.00%	100.00%
MenloToken.sol	100.00%	62.50%	100.00%	100.00%
MenloTokenPresale.sol	100.00%	100.00%	100.00%	100.00%
MenloTokenSale.sol	100.00%	100.00%	100.00%	100.00%
MenloTokenTimelock.sol	100.00%	91.67%	100.00%	100.00%
MenloWalletTimelock.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/math/SafeMath.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/ownership/CanReclaimToken.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/ownership/Ownable.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/BurnableToken.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/ERC20.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/PausableToken.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/SafeERC20.sol	100.00%	100.00%	100.00%	100.00%

zeppelin-solidity/token/ERC20/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
zeppelin-solidity/token/ERC20/TokenTimeLock.sol	100.00%	100.00%	100.00%	100.00%
All Files	**100.00%**	**96.49%**	**100.00%**	**100.00%**