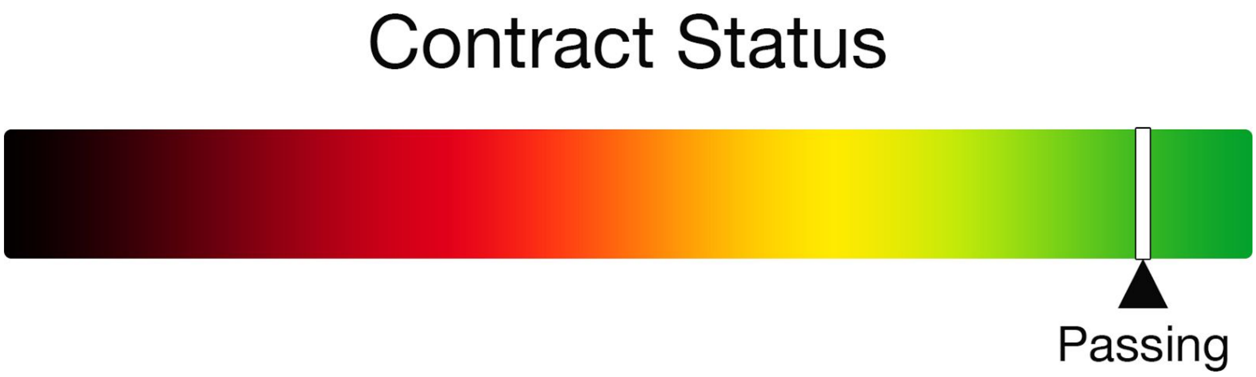
OriginTrail Follow-up Contract Audit

by Hosho, January 2018

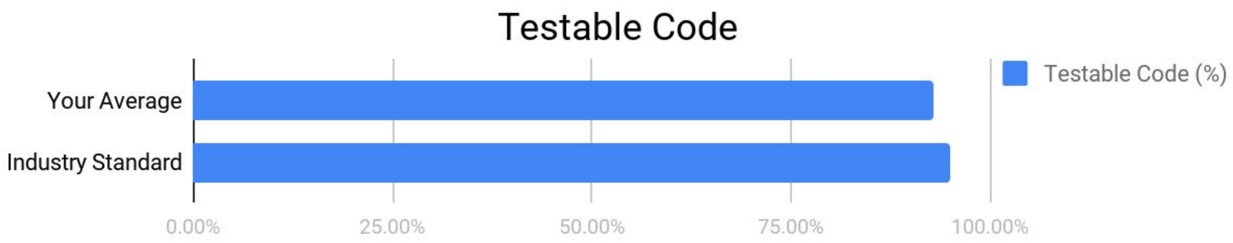
NOTICE: *This document is a draft and is not representative of a completed audit.*

Executive Summary

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



All issues have been successfully addressed and suggestions implemented. See Complete Analysis.



The testable code is on par with industry standard. 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 OriginTrail 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 smart contracts written on January 5, 2018 and then completed a thorough follow-up review of the code as written and last updated on January 13, 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 OriginTrail's 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

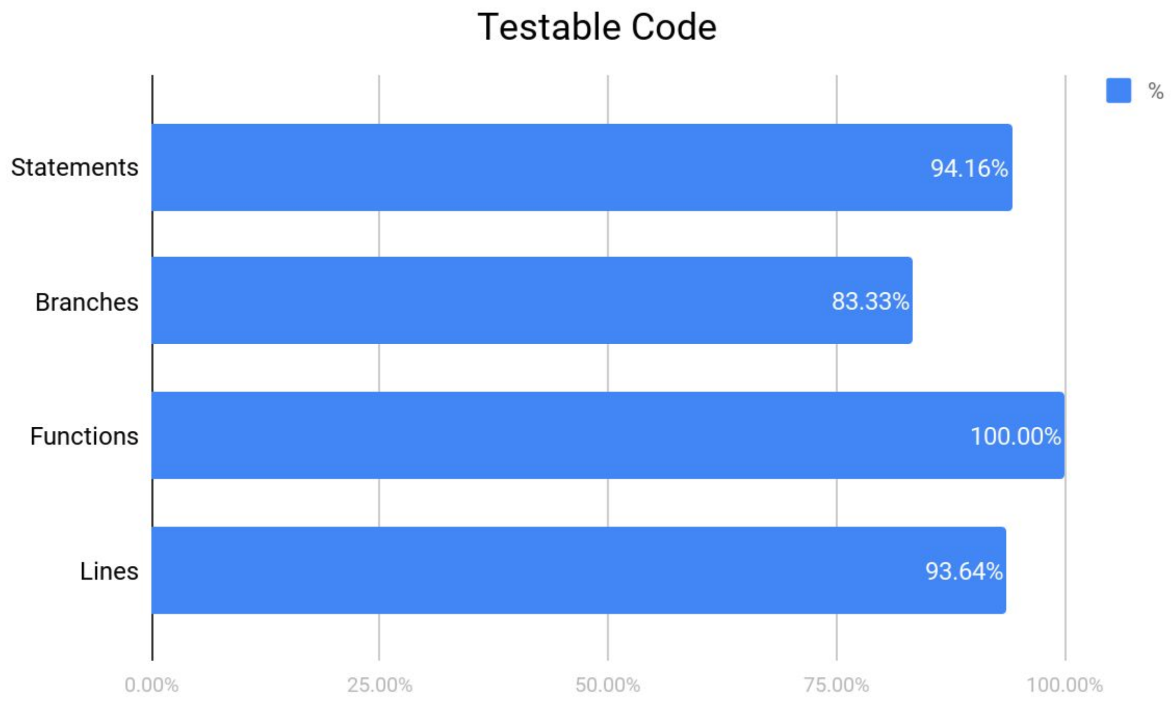
The TraceToken and the associated TraceTokenSale are a standard pairing of an ERC-20 token with token sale functionality. The token sale portion contains a simple tranching mechanism designed to issue discounted tokens during the initial two weeks of the crowdsale. The ERC-20 token is written precisely to ERC-20 guidelines and standards. Both components are well written and the paths through code execution are easy to navigate. Each of the issues discovered during the auditing process has been successfully corrected by the OriginTrail team.

Testable code is slightly lower than industry standard, however, this is simply due to the need to manually test the refund process within the TraceTokenSale. There is a high amount of time shifting that is required to validate these contracts, which is why manual testing is necessary.

Lastly, we completed a successful re-deployment against the Ropsten test-network, with permission from the OriginTrail Team. The deployment utilized approximately 3.5M gas, of which about 920k was burned to deploy the individual token component, which is why the total gas usage is fairly high. Optimizations were enabled in order to ensure the contract would fit on the network properly within required gas limits.

2.2 Coverage Report

As part of our work assisting OriginTrail 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:

- **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.

7.1. Resolved, High: Invalid Inheritance

TraceToken.sol

Explanation

The `finishMinting` function contained in TraceToken overrides the `finishMinting` function in MintableToken and alters the function signature. When this is called via `super.mintableToken`, an invalid argument count error is thrown due to the signature alteration. This appears to only occur when the function is called directly during testing. However, if it is called through a secondary contract, TraceTokenSale for example, the function operates as intended.

Resolution

The OriginTrail Team updated the function `finishMinting` in TraceToken.sol to `endMinting` eliminating inheritance as well as having `endMinting` call `super.finishMinting` (in MintableToken.sol)

7.2. Resolved, Medium: Double Event Issuance

TraceToken.sol

Explanation

The Minting function issues two Transfer events, rather than the recommended single event issuance. Having two events fire for a single function leads to incorrect logging and runs contrary to ERC-20 guidelines.

Resolution

The Transfer event in TraceToken.sol has been removed by the OriginTrail Team and the Transfer event in MintableToken.sol remains as the sole issuance event.

7.3. Resolved, Low: Event Best Practices

TraceToken.sol

Explanation

Best practices indicate that an event should always have a variable name attached to it in the form of `EventName(bool variableName)`. The TransferAllowed event, `TransferAllowed(bool)`, does not have a variable name. An example of a proper format would be `TransferAllowed(bool canTransfer).`

Resolution

The event has been updated by the OriginTrail Team to `TransferAllowed(bool transferIsAllowed)` which contains the variable name as suggested.

7.3. Resolved, Low: Non-Needed Check

TraceTokenSale.sol

Explanation

CalcAmount performs a `require(now<=endTime)` check to ensure that the current time is before the set `endTime` for the contract, which is performed previously during the execution of `validPurchase()`. Additional unnecessary checks cause code clutter, making it less readable and harder to follow code paths.

Resolution

The unnecessary check has been removed by the OriginTrail Team.

7.3. Resolved, Low: Non-Needed Check

TraceTokenSale.sol

Explanation

CalcAmount performs an unnecessary check on line 123 verifying that the current time is within week two. There is an outer check, on line 108, that performs this same within 2 week check, followed by an if statement that performs the 1 week check. Additional unnecessary checks cause code clutter, making it less readable and harder to follow code paths.

Resolution

The unnecessary check has been removed by the OriginTrail Team.

7.4. Resolved, Informational: Ambiguous Initial Value

TraceToken.sol

Explanation

The `transferAllowed` `public bool`, initializes on line 11 without a value. For contract readability and clarity, this should always be initialized with a default false value.

Resolution

The `transferAllowed` variable is now initialized with a default false value, updated by the OriginTrail Team.

7.5. Resolved, Informational: Grammar

TraceToken.sol

Explanation

The name of the token is "Trace token". Standard naming conventions indicate that this would normally be "Trace" or "Trace Token".

Resolution

The token name has been updated from “Trace token” to “Trace Token” bringing it in line with standard naming conventions.

---

4. Closing Statement

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

Overall, the ERC-20 token and token sale contracts are well written and function as intended. The OriginTrail team has been responsive and quick to correct any issues that were discovered during the auditing process. Any suggestions made were promptly and correctly implemented. As a small team of experts, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, we can say with confidence that the OriginTrail 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 OriginTrail Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Code Edits

- Softcap - Set to 110 ETH
- Hardcap - Set to 500 ETH
- presaleLimit - Set to 200 ETH
- max_investment_eth - Set to 50 ETH

Contract: ERC-20 Tests for TraceToken

✓ Should deploy a token with the proper configuration (57ms)

✓ Should allocate tokens per the minting function, and validate balances (216ms)

✓ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (65ms)

✓ Should not transfer negative token amounts (48ms)

✓ Should not transfer more tokens than you have (44ms)

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (39ms)

✓ Should not allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer an additional 500 tokens once authorized, and authorization balance is > 0

✓ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (61ms)

✓ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (145ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (53ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0 (48ms)

✓ Should not transfer tokens to 0x0 (53ms)

✓ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (63ms)

✓ Should allow an approval to be set, then increased, and decreased (167ms)

Contract: TraceToken customizations

✓ Should allocate tokens per the minting function (164ms)

✓ Should not permit trading before the minting is complete

✓ Should not permit minting once minting is complete (78ms)

✓ Should allow the owner, and only the owner to change ownership of the token to a non 0x0 address (82ms)

```
Contract: TraceTokenSale
  ✓ Should allow whitelisting by the owner, with the correct number of parameters (737ms)
  ✓ Should not allow founder, advisory, or refund payouts before the contract ends
  ✓ Should not allow purchases before the start time hits (44ms)
  ✓ Should after, the start time has hit, allow the purchase of tokens - Presale Week 1 (841ms)
  ✓ Should not permit a withdraw until the softcap is hit (129ms)
  ✓ Should issue the correct tokens for week 2 crowdsale (508ms)
  ✓ Should permit withdraws once the softcap is hit (109ms)
  ✓ Should not permit a 0 wei withdrawl (86ms)
  ✓ Should not be finalizable if the hardcap isn't hit, or the end time isn't reached.
  ✓ Should let the contract be maxed out, then ended (955ms)
  ✓ Should only be able to be finalized once
  ✓ Should once the contract hasEnded due to time, block whitelisting and purchases (386ms)
  ✓ Should, every 90 days release funds as appropriate to the correct wallets (2168ms)
  ✓ Should allow transfers once the token sales are complete (42ms)
```

6. All Contract Files Tested

Original Files

File	Fingerprint (SHA256)
contracts/TraceToken.sol	d189514d8c746e58464d721024d5b9d9814c4b8d73bd8f89b361610d027b9852
contracts/TraceTokenSale.sol	7e16a924e6c9c85967b570735dd462904bf36f61ebbb587e6929749619620f2f
contracts/math/SafeMath.sol	76aaf63d1bacf12497338a3be3ffcd2bf7b212826b6ddff6e1c008352a3fbf3b
contracts/ownership/Ownable.sol	306c614dcec1cbdc5415919c13784b721084aba160685b1427892f55dad8aa68
contracts/token/BasicToken.sol	ec2fbeb88936a555470405a603c89ce9db44b345fd998095bd682f3d1b6204cd
contracts/token/ERC20.sol	5145438d41545f1cccc95d55254f57b3bc81d68da3f9ef4d116bfae55d332104
contracts/token/ERC20Basic.sol	5c1392929d1a8c2caeb33a746e83294d5a55d7340c8870b2c829f4d7f6ed9434
contracts/token/MintableToken.sol	7cb2747b21e2d7f21b4fde47765e35b466a8244e933585a455163c5aa21d983a
contrcts/token/StandardToken.sol	6de05d4b4672fe6e15fe5a2e76b0b3721b40a637643da5c4e43b50b99224b9ca

Files last updated on January 9, 2018

File	Fingerprint (SHA256)
contracts/TraceToken.sol	932c1882a9758d518a595409501a93c6280acd3552290d39abfef61207276979
contracts/TraceTokenSale.sol	654be15cb7dfd2d5741401698b3f42c1557b72db0d2e84675ada5a9d5f334397
contracts/math/SafeMath.sol	76aaf63d1bacf12497338a3be3ffcd2bf7b212826b6ddff6e1c008352a3fbf3b
contracts/ownership/Ownable.sol	306c614dcec1cbdc5415919c13784b721084aba160685b1427892f55dad8aa68
contracts/token/BasicToken.sol	ec2fbeb88936a555470405a603c89ce9db44b345fd998095bd682f3d1b6204cd
contracts/token/ERC20.sol	5145438d41545f1cccc95d55254f57b3bc81d68da3f9ef4d116bfae55d332104
contracts/token/ERC20Basic.sol	5c1392929d1a8c2caeb33a746e83294d5a55d7340c8870b2c829f4d7f6ed9434
contracts/token/MintableToken.sol	7cb2747b21e2d7f21b4fde47765e35b466a8244e933585a455163c5aa21d983a

contracts/token/StandardToken.sol	`6de05d4b4672fe6e15fe5a2e76b0b3721b40a637643da5c4e43b50b99224b9ca`

7. Individual File Coverage Report

Coverage has not changed for updated files.

File	% Statements	% Branches	% Functions	% Lines
contracts/TraceToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/TraceTokenSale.sol	90.43%	81.52%	100.00%	90.27%
contracts/math/SafeMath.sol	100.00%	50.00%	100.00%	100.00%
contracts/ownership/Ownable.sol	100.00%	50.00%	100.00%	100.00%
contracts/token/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/token/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/token/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/token/MintableToken.sol	100.00%	100.00%	100.00%	100.00%
contrcts/token/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
All files	**94.19%**	**83.33%**	**100.00%**	**93.68%**