DLT Labs

# OriginTrail Code Audit

## By

## DLT Labs

# Table of Contents

# 1 INTRODUCTION

## 1.1. DOCUMENT PURPOSE

**DLT Labs** conducted the review of the smart contracts that makes up the **OriginTrail Token Sale** service. The findings of the review are presented in this document.

## 1.2. REVIEW GOALS & FOCUS

### Sound Architecture

This review includes both objective findings from the contract code as well as subjective assessments of the overall architecture and design choices. Given the subjective nature of certain findings commentary from the OriginTrail development team has been included in the report as appropriate.

### Smart Contract Best Practices

This review will evaluate whether the codebase follows the current established best practices for smart contract development.

### Code Correctness

This review will evaluate whether the code does what it is intended to do.

### Code Quality

This review will evaluate whether the code has been written in a way that ensures readability and maintainability.

### Security

This review will look for exploitable security vulnerabilities.

## *1.3.* TERMINOLOGY

This review uses the following terminology:

### Severity Terms

Measure the magnitude of an issue.

### Minor

Minor issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Minor issues in general will not indicate an actual problem or bug in code.

The maintainers should use their own judgement as to whether addressing these issues improves the codebase.

### Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable such as requiring a specific condition to arise to be exploited.

Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

### Critical

Critical issues are directly exploitable bugs or security vulnerabilities. Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.

# 2. SOURCE CODE

## 2.1. REPOSITORY

This audit has been done for contracts in the following repository:
https://github.com/prospeh/token-sale-contracts/tree/master/contracts

## 2.2. CONTRACTS

The code audit carried out was limited to the smart contract(s) shared over the repository as below: -

- **TraceToken.sol**
- **TraceTokenSale.sol**
- **Base contracts available in math, ownership and token folder**

# 3. DISCLAIMER

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The Audit also holds no responsibility on the analysis given by open source tools used in this report. The output of the tools should only be used as help in the analysis.

# 4. FINDINGS

This section contains detailed issues and analysis.

## 4.1. CRITICAL ISSUES

***None***

## 4.2. MAJOR ISSUES

**Issue**# 001
**Codes commit#**: cd7a51ca4d4ce99684869ffb8c197e385e588ca4

**Description**:
While buying tokens, tokens are not allocated in decimals.
Example: During the second week of presale, when tokens are bought for 1 ether, 9444.000000000000000000 tokens are allocated, instead 9444.444444444444444444 tokens should be received.
**Problem:** In **calcAmount()** function, discounting on the variable token_per_wei discards the decimals that should be available after multiplying by msg.value
**Solution:** Multiply first, and afterwards calculate discount
*msg.value.mul(token_per_wei.mul(100)).div(80)*

**Comment**:
**Action**:

**Issue**# 002
**Codes commit#**: cd7a51ca4d4ce99684869ffb8c197e385e588ca4

**Description**:
In **refund()** function, tokens are not burned of an account(who asked for refund) after refunding ether.

**Comment**:
**Action**:

## 4.3. MINOR ISSUES

**Issue**# 003
**Codes commit**#: cd7a51ca4d4ce99684869ffb8c197e385e588ca4
**Description**:

In **TraceTokenSale()** function of TraceTokenSale.sol, there is no zero address check for input parameter **_foundersWallet** and **_advisorsAndTeamWallet**

**Comment**:
**Action**:

**Issue**# 004
**Codes commit**#: cd7a51ca4d4ce99684869ffb8c197e385e588ca4

**Description**:
In **addWhitelist()** function of TraceTokenSale.sol, if owner whitelist a contributor that is already whitelisted then contributor's whitelist amount will be reset to new value irrespective of his current whitelist amount.
If addWhitelist() function is intended to add whitelist amount to contributor's whitelist amount then implementation should consider contributor's existing whitelist amount as

```
whitelist[contributor] = whitelist[contributor]+eth_amount;
Whitelist(contributor, whitelist[contributor]);
```

**Comment**:
**Action**:

# 5. General Comments

Following are some general comments which should be fixed/modified for better code quality: -

- The contract is coded to be complied with solidity version – 0.4.18. Few improvements and new features have been added in this solidity version.
  Below is one of them-
  Constant functions are split into following two types of function –

  View – Does not write to the state.
  Pure – Neither reads from nor writes to the state.

  It is recommended to replace '**constant'** keyword with '**pure'** and '**view'** keyword as per their use case. Solidity version 0.4.17 enforces to use the above-mentioned keywords. Although, it won't give error, but it will give warnings.