LeadCoin Smart Contracts Audit by ZK Labs

MATTHEW DI FERRANTE

2018-02-13

Introduction

On 2018-02-13, Matthew Di Ferrante performed an audit of the LeadCoin smart contracts. My findings are detailed below.

I, Matthew Di Ferrante have no stake or vested interest in LeadCoin. This audit was performed under a contracted rate with no other compensation.

Authenticity

This document should have an attached cryptographic signature to ensure it has not been tampered with. The signature can be verified using the public key from http://keybase.io/mattdf

Audit Goals and Focus

Smart Contract Best Practices

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

Code Correctness

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

Code Quality

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

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to either the operators of ChainLink or its users.

Testing and testability

This audit will examine how easily tested the code is, and review how thoroughly tested the code is.

About LeadCoin

LeadCoin is a platform that enables businesses to sell their unused leads and buy hot leads from other businesses. Sellers automatically share their unused leads in real-time and matching algorithms pair them with buyers.

Terminology

This audit uses the following terminology.

Likelihood

How likely a bug is to be encountered or exploited in the wild, as specified by the OWASP risk rating methodology.

Impact

The impact a bug would have if exploited, as specified by the OWASP risk rating methodology.

Severity

How serious the issue is, derived from Likelihood and Impact as specified by the OWASP risk rating methodology.

Overview

Source Code

The following files were audited:

```
543b3f456717d6445b134c4de5f9be38170f3a0284bbe862dcb0ad7fef156406
   LeadcoinSmartToken.sol
a3527460ff8142416964a6d46ac8e504a9b42deaa0afaa1b069bd2db14cfe47e
   LeadcoinCrowdsale.sol
```

The code makes extensive use of Bancor and OpenZeppelin library code, which was *not* audited as part of this audit.

General Notes

The code is generally well structured, without overly complex functions. It makes extensive use of the OpenZeppelin smart contracts, which reduces the count of lines that need to be independently audited and the risk of bugs.

Contracts

LeadcoinCrowdsale

`LeadcoinCrowdsale` implements the main ICO logic, and inherits from `OpenZeppelin`'s `FinalizableCrowdsale`. Beyond function overrides, the main functionality added is a set of functions to manage `grantees`, or allocations for presale or non-ether buyers.

The sale runs as a fixed price, time limited sale, with unrestricted participation, and frozen transfers until the sale completes. The `grantee` modifications can only take place while the sale runs. There is no limit to the *amount* that can be issued to grantees, even though there is a limit to the number of grantees (mostly to not run into gas issues).

The sale can be closed or finalized by the owner once the `hasEnded` condition returns true, which in this case would be when the cap is reached or the time has run out. Upon finalization, the new supply is calculated, and tokens are issued to team, company, and reserve wallets. Transfers are enabled, `destroying` tokens is re-enabled, and the token contract's ownership is transferred to the owner of the crowdsale contract.

Besides this, the majority of the logic implemented from OpenZeppelin's `Crowdsale` is untouched. There is only an extension to `validPurchase` which adds fiat raised (converted to wei) when calculating the amount raised, to account for non-ether contributions.

LeadcoinSmartToken

The `LeadcoinSmartToken` contract is an instance of `LimitedTransferBancorSmartToken` and `TokenHolder`. It contains variable initializations for token name, symbol and decimals. There are no issues in the contract.

Testing

Test coverage is comprehensive, covering cases for each individual file.

No automated build is set up for the repository.

We recommend setting up an automated build, so new commits can be vetted against the existing test suite.

Findings

We found 1 note issue in the crowdsale's construction.

Note Issues

Token ownership should be restricted explicitly to a vetted smart contract

- Likelihood: low
- Impact: medium

Once the crowdsale finalizes, the token ownership is set to the deployer address of the crowdsale. That address will have the ability to issue tokens at will. Even though issuance cannot be disabled due to the bancor functionality relying on it, token issuance should be restricted to just the bancor smart contract. The severity is low due to the fact that once the token ownership is transferred to the bancor contract this will be mitigated, and trusting the crowdsale operators not to issue tokens in the interim is not too strict an assumption.

Low Issues

None.

Medium Issues

None.

High Issues

None found.

Critical Issues

None found.