Cloudbric Crowdsale Smart Contract Audit

1. Introduction

iosiro was commissioned by Cloudbric to conduct an audit on their token and crowdsale smart contracts for their Initial Coin Offering (ICO). The audit was performed between 14 August 2018 and 16 August 2018.

This report is organized into the following sections.

- **Section 2 - Executive Summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit Details:** A description of the scope and methodology of the audit.
- **Section 4 - Design Specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed Findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to understand the risk exposure of the smart contracts, and as a guide to improve the security posture of the smart contracts by remediating the issues that were identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

2. Executive Summary

This report presents the findings of an audit performed by iosiro on the client's token and crowdsale smart contracts. The purpose of the audit was to achieve the following.

- Ensure that the smart contracts functioned as intended.
- Identify potential security flaws.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regards to cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. There are a number of techniques that can help to achieve this, some of which are described below.

- Security should be integrated into the development lifecycle.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed wherever possible.

At the conclusion of the audit, one low risk issue was open that could result in accidentally allocating excess tokens to an address during the early investment round. Additionally, an informational issue was open that presented design recommendations that would result in more strictly following expected behaviour and best practice.

The code was of a high standard, as it was well designed and clearly written. It separated token and crowdsale logic and made use of commonly used libraries, where reasonable. A comprehensive test suite was also provided by the development team.

The risk posed by the smart contracts can be further mitigated by using the following controls prior to releasing the contracts to a production environment.

- Use a public bug bounty program to identify security vulnerabilities.
- Perform additional audits using different teams.

3. Audit Details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from any other files are considered to be out-of-scope.

3.1.1 Cloudbric-Project

Project Name: Cloudbric-Contracts
Commit: b0fbefa
Files: Cloudbric.sol, CloudbricSale.sol

3.2 Methodology

A variety of techniques were used to perform the audit, these are outlined below.

3.2.1 Dynamic Analysis

The contracts were compiled, deployed, and tested using both Truffle tests and manually on a local test network. A number of pre-existing tests were included in the project.

3.2.2 Automated Analysis

Tools were used to automatically detect the presence of potential vulnerabilities, such as reentrancy, timestamp dependency bugs, transaction-ordering dependency bugs, and so on. Static analysis was conducted using Mythril and Oyente. Additional tools, such as the Remix IDE, compilation output and linters were used to identify potential security flaws.

3.2.3 Code Review

Source code was manually reviewed to identify potential security flaws. This type of analysis is useful for detecting business logic flaws and edge-cases that may not be detected through dynamic or static analysis.

3.3 Risk Ratings

Each Issue identified during the audit is assigned a risk rating. The rating is dependent on the criteria outlined below..

- **High Risk** - The issue could result in a loss of funds for the contract owner or users.
- **Medium Risk** - The issue results in the code specification operating incorrectly.
- **Low Risk** - A best practice or design issue that could affect the security standard of the contract.
- **Informational** - The issue addresses a lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.
- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

4. Design Specification

The following section outlines the intended functionality of the smart contracts.

4.1 Cloudbric Token

The token functionality is described below.

ERC20 Token

The token implements the ERC20 standard.

Field	**Value**
Symbol	CLB
Name	Cloudbric
Decimals	18
Total Supply	1,000,000,000

Token Allocations

- Crowdsale - 540,000,000 tokens
- Admin (Team) - 460,000,000 tokens

Lock

It is possible to place a lock on an address. This prohibits a token holder from withdrawing more than a specified number of tokens.

Pausable

It is possible to stop the transfer of tokens entirely, except for the admin account and token sale addresses. The default state is paused.

Burnable

It is possible to remove tokens from the total supply.

4.2 Crowdsale

The crowdsale functionality is described below.

Rounds

There are four distinct rounds:

- Early Investment - Tokens can be purchased through the crowdsale or sent by the admin account directly to participants without requiring ether in exchange.

This can be used for off-chain contributions.

- Presale1 - Tokens can be purchased in exchange for ether at the specified exchange rate.
- Presale2 - Tokens can be purchased in exchange for ether at the specified exchange rate.
- Crowdsale - Tokens can be purchased in exchange for ether at the specified exchange rate.

Stages

Each round has three stages these are described below.

- Setup - The round details (i.e. minimum contribution amount, maximum contribution amount, hard cap, and exchange rate per round) can be set at this stage.
- Started - It is possible to purchase tokens during this stage.
- Ended - At this stage, it is possible to proceed to the next round.

Whitelist

Participants need to be whitelisted before being able to contribute funds to the crowdsale.

Allocations

During the early investment round, it is possible for the crowdsale owner to send tokens to participants without requiring ether in exchange.

Defaults

- Hard cap - 20,000 ETH
- Cloudbric to ETH rate - 10,000 CLB/ETH
- Minimum contribution amount - 0.1 ETH

5. Detailed Findings

The following section includes in depth descriptions of the findings of the audit.

5.1 High Risk

No high risk issues were present at the conclusion of the audit.

5.2 Medium Risk

No medium risk issues were present at the conclusion of the audit.

5.3 Low Risk

5.3.1 Possible to Allocate Excess Tokens

CloudbricSale.sol: Lines 463-475

Description

It was possible for `allocateTokens(...)` to allocate more tokens than intended if called multiple times with the same address. This vulnerability stemmed from the fact that `tokenAmount` was not deducted from the balance of the address after allocating the tokens. For example, if `allowedAmount` is set to 100, then `allocateTokens(...)` could in theory be called several times, allocating up to 100 tokens for a single address each time that the function is called.

Remedial Action

It is recommended that the following line is added to the end of the `allocateTokens(...)` function before returning. This will correctly deduct the allocation from the allocation balance of the address.

```
allocationList[to].allowedAmount =
allocationList[to].allowedAmount.sub(tokenAmount);
```

5.4 Informational

5.4.1 Design Comments

General

The following describes possible actions to improve the functionality and readability of the codebase.

Round times unenforced

It was possible to call the endSale function, ending the crowdsale, at any stage during the `Started` stage. It is recommended that assertions are used to ensure that now is larger than `endTime` or that the `weiRaised` round variable is used to determine whether the round hard cap has been reached. In this way, participants can be assured that the crowdsale will last the expected duration.

Possible to restart rounds

It was possible to call the `setUpSale` function with the same round multiple times. While this does not impact on the functionality of the code, it can lead to unintuitive outcomes. For example, it may be possible to have multiple presale rounds. Unless this is desired functionality, it is recommended that the round is automatically incremented when `endSale` is called, ensuring that only the expected rounds are used.

Unnecessary use of private visibility

A number of state variable were found to have a `private` visibility set. As all information on the Ethereum blockchain is visible, it is still possible to access variables marked as private. The `private` visibility simply prohibits other contracts from accessing the variable. It is recommended that the following changes are made:

- CloudbricSale.sol: lines 15, 70 should be made public.
- Cloudbric.sol: line 29 should be made public

From a developer's perspective, marking a variable as private complicates the process of ensuring that variables have been correctly assigned, and offers little to no security benefit.

Emit keyword not used

The emit keyword was found to be missing before events. This style has been deprecated, so it is recommended that the `emit` keyword is added before each event, e.g. `emit Transfer(...)`.

5.5 Closed

No closed issues were present at the conclusion of the audit.