Sooloox ICO Audit

by MonteLabs

March 4, 2018

1 Introduction

The Sooloox ICO consists of two parts, (i) the CAK Crowdsale and (ii) the CAK token.

This audit checks for security and correctness issues in both parts, as well as in their interfaces.

The code was written by Validity Labs. This document is a report containing our findings while analysing and testing the code.

We have analysed the two smart contracts:

- `CakCrowdsale.sol`;
- `CakToken.sol`;

These contracts use OpenZeppelin smart contract libraries which are considered standard and have been heavily audited and used by the community in general.

Figure 1 shows the relation between the used smart contracts, where the red nodes are the OpenZeppelin libraries and the green nodes are the newly implemented smart contracts, object of this audit.

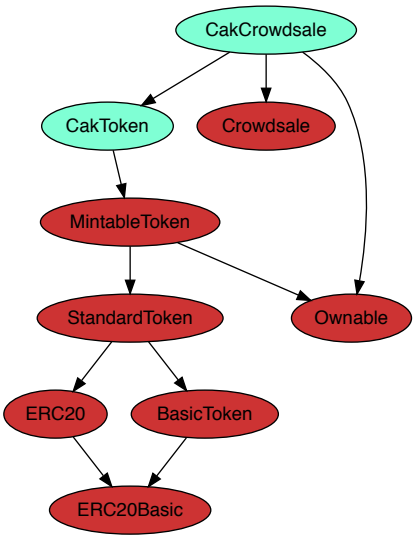

Figure 1: Relationship between the smart contracts used in the Sooloox ICO.

The code has a very high quality, employs good practices and contains many important tests.

The `CakToken` implementation is fairly simple, it implements OpenZeppelin's `MintableToken` contract, just changing three parameters for token identification

and to define the token's decimals. Although the choice of having 0 decimals in a token is unusual, it is not a problem, and no issues were found in the `CakToken` implementation.

Section 2 lists the issues found during this audit for the Crowdsale and the Sooloox token, classified according to its severity. For each issue we suggest a solution.

Section 3 presents our concluding thoughts on the audit.

2 Issues

Only *Low severity* issues were found, which can be easily fixed.

2.1 Low severity

Function `buyTokens` can be more expensive than it should.
Due to the unusual nature of the `CAK` token of having no decimals, special attention should be made to inform the user about certain calls. Users should call `fallback` or `buyTokens`, with an amount of Ether that is divisible by `rate`. Failing to do so will result in a more expensive transaction. In our tests, the difference is about 14427 gas, out of 120157 gas for the transaction. That is an increase of 12% in the transaction cost.

Timestamp dependency.
The crowdsale contract relies on timestamp (`now`) for its internal logic. The timestamp of a block can be manipulated by a malicious miner by a few hours. We do not think this can be a major problem, since the smallest time unit used in the contracts is `days`.

Function `forwardFunds(uint256 refund)` does not override OpenZeppelin's Crowdsale function.
Since the function at `CakCrowdsale.sol:210` has a signature that is different from the one in OpenZeppelin's `Crowdsale` contract, this implementation does not override the function in OpenZeppelin's `Crowdsale` contract. We suggest renaming this function or changing the function's documentation (comments).

2.2 Minor suggestions

- There is no need for the fallback function at `CakCrowdsale.sol:71`. The same function is already implemented in OpenZeppelin's `Crowdsale` contract.
- Function `refundLeftOverWei` at `Crowdsale.sol:194` can be made internal, since it is only used in function `buyTokens`. This function is expected to be called often, so this action should save gas.
- In function `refundLeftOverWei` at `Crowdsale.sol:194`, there is no need to check that `weiInvested > 0` assuming that `rate != 0`, because of the line `require(msg.value >= rate);` at `Crowdsale.sol:99`.
- In order to save `gas`, some functions can be made `external`. For instance, if `managers` from `CakCrowdsale.sol` are never smart contracts, functions `whiteListInvestor`, `batchWhiteListInvestors`, `unWhiteListInvestor` can be made `external`.

3 Conclusion

The Sooloox ICO smart contracts provided by Validity Labs have very good quality, and their test suite leaves almost no space for problems. We have not found any major issues in the smart contracts.

As a general guideline, the use of already deployed libraries could make the contracts much cheaper to deploy. Heavily tested libraries, such as the OpenZeppelin smart contracts, can also help the overall Ethereum ecosystem. Following those principles, we store audits on-chain backed by IPFS (InterPlanetary File System). This audit can be accessed at montelabs.com/audits as soon as the Sooloox contracts are deployed, where all the security evidences can be seen and checked.