SOLIDIFIED

Audit Report for ICOStart. May 14, 2018.

Summary

Audit Report prepared by Solidified for ICOStart covering their crowdsale smart contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below contracts. The debrief took place on May 14, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- ICOStartPromo.sol
- ICOStartSale.sol
- ICOStartReservation.sol
- ICOStartToken.sol
- LockableWhitelisted.sol

Notes:
The audit was performed on commit `eb2e72f1d48e9d511acae4680fd07b90678626b1`
The audit was based on the solidity compiler `0.4.23+commit.124ca40d`

Intended Behavior

The purpose of the contracts is to implement the ICOStart token crowdsale. Behavior was checked purely against specification declared in https://github.com/ICOStart/Crowdsale/blob/eb2e72f1d48e9d511acae4680fd07b90678626b1/README.md

For clarity we would like to state that the contracts do not contain any trustless refund mechanism, don't enforce any minimum or maximum total contribution (softcap or hardcap) and their sole purpose is to gather Ether from users and tokenize their investment in form of an ERC20 token.

Issues Found

Critical

No critical vulnerabilities were identified.

Major

No major vulnerabilities were identified.

Minor

1. Sale periods values are not trustless

The owner can arbitrary modify sale periods. An attack would be to open the crowdsale at some period with an attractive price, then delete the periods and add another period with higher rates. This is essentially the same as being able to change the token price arbitrarily during the sale, which should be avoided at all costs.

Recommendation

Consider setting the sale periods in the constructor and removing the function `clearPeriods()`.

2. Reservation payout is not trustless

In order for tokens to be made claimable by investors in `ICOStartReservation.sol`, firstly the `pay()` function needs to be called. At the moment, this function is only callable by the owner, which means that investors can claim their coins only if the owner allows for it.

Recommendation

Consider removing the `onlyOwner` modifier from `pay()` and the `whenPaid()` modifier from claim tokens.

3. Owner can drain ether from reservations without issuing tokens

The owner can kill any `ICOStartReservation` contract and drain the ether that is being reserved in the contract without submitting the reservation.

Recommendation

Remove the `destroy` function from `ICOStartReservation`.

AMENDED [17 May 2018]

ICOStart response, regarding issues 1 to 3:

By design we trust the owner, as he has no economic incentive to attack the contracts to steal funds or distribute useless tokens to contributors. As such, the `onlyOwner` functions were put in exclusively as last-resort emergency measures, not aiming for full trustlessness.

Solidified response:
We agree that the contract owners do not have a direct incentive to attack the contracts, but these issues represent potential breaks in the intended behaviour or in the user expectations.

4. ICOStartToken transfers can be paused by owner

Due to the import of `LockableWhiteListed` contract in `ICOStartToken.sol`, an owner can prevent users from transacting with their coins by pausing them.

Recommendation

Consider disallowing the call to `lock` in `LockableWhiteListed` after the coins have been unlocked once.

AMENDED [17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645.`

5. Fallback function in ICOStartReservation can result in a user depositing more funds than they want

Users are not accustomed to sending 0-ether transactions to a contract. As a result, it may be unclear to a non-technical user how they are supposed to claim their tokens through the fallback function.

Recommendation

Consider removing the `if msg.value == 0` block of the code and encourage users to call `claimTokens`.

6. Users can send Ether to ICOStartPromo

As per the supplied README file, the `ICOStartPromo` contract should not accept ether, however it defines a payable fallback function.

Recommendation

Consider removing the fallback function.

AMENDED [17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645.`

7. ICO Periods are not enforced to be sequential

The function addPeriod accepts the time duration for sale in any order and even a time in the past. This can allow the owner to add invalid sale durations.

Recommendation

It is recommended to verify the sale duration for correct order for each addition process.

Notes

8. Information disclosure in repository configuration files

Internal IP Addresses as well as the live address of the developer account are disclosed in truffle configuration files. While this is not necessarily a security vulnerability, consider removing these details from the configuration files for maintaining operations security.

AMENDED [17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645.`

9. Install zeppelin-solidity via npm

The implementations of dependencies taken from OpenZeppelin are manually included in the repository. Receiving updates and security fixes needs to be done manually which can result in not applying a critical security patch if a bug is found in OpenZeppelin's contracts.

Recommendation
Consider installing zeppelin-solidity through npm via
`npm install --save zeppelin-solidity`
Refer to Zeppelin's Documentation.

AMENDED [17 May 2018]

ICOStart response:

Due to a bug in truffle-flattener, files included from the /node_modules directory are not found. Plus, with the default directory layout we cannot run zeppelin-solidity's own unit tests, which we might want to. Plus, by copying only the necessary files from the /contracts and /test directories of zeppelin-solidity we make it more clear what we are using and what we are not. Each development cycle uses a fixed version of everything and as such any updates and security fixes in zeppelin-solidity would trigger a new development cycle anyway, with all attached related reviews and scrutiny. Finally, we prefer to wait a bit before using a newly released version in case any critical vulnerability pop up (unless of course the previous version has itself critical vulnerabilities fixed by the new one).

10. ICOStartPromo can transfer more tokens than total supply

A `totalSupply` variable is defined in `ICOStartPromo` which would be expected to be checked against when airdropping promotional tokens. Consider checking the total supply when airdropping tokens to addresses, if that is the intended behavior, otherwise remove the `totalSupply` variable from the contract.

11. Whitelist implementation in ICOStartSale is inefficient

All conditional checks in `addAddressToWhitelist` and `removeAddressFromWhitelist` are redundant. Consider calling `delete` on the mapping when removing an address from whitelist. In case there is a need to communicate the whitelist change to a UI, also consider emitting an `Event`.

AMENDED [17 May 2018]

ICOStart response:

Not inefficient. Matter of taste mostly; return value is used (which makes the test necessary), not sure an event would be easier or less costly. Not changed for the time being as it's close to irrelevant in our case.

12. Gas costly pattern in loops

Due to how loops are implemented in Solidity, the length of an array is read on every iteration in, if it's included in the loop's body, which results in 200 gas being wasted per iteration, given that the array's length is not modified. Consider assigning the length of the array outside the loop and use that variable instead. This occurs both when adding and removing addresses from the whitelist and on every `_getCurrentPeriod()` call. Paper for reference: https://arxiv.org/abs/1703.03994

AMENDED [17 May 2018]

This suggestion has been partially incorporated by the ICOStart team in commit `771ba898cccb2fe82a2dda46af3245fe22b5338f`. Length of the array is calculated outside the loop in `_getCurrentPeriod()` function.

13. Redundant assertions in airdrop and multi transfer functions

According to the `ERC20` specification, any successful call to `transfer` always returns `true`. If the execution fails then it reverts, which bubbles up to the original calling function. As a result, any assertions are unneeded.

AMENDED[17 May 2018]

This issue has been incorporated by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645`.

14. Include a null address check for manager

There is no null address check in `ICOStartReservation.sol` constructor about the address of manager. Consider adding a null address check for manager address.

AMENDED[17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645`.

15. Replace if - revert patterns with require

Consider replacing all `if(falsestatement) { … revert()  }` patterns to `require(truestatement).`

AMENDED[17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645`.

16. Consider using latest version of solidity

The contracts use solidity version 0.4.19. It is suggested to use the latest version (`0.4.23`) and fix all compiler warnings that arise.

17. Using constructors without the constructor keyword is deprecated

Using constructors that have the same function name as the contract is deprecated in the latest version of solidity (`0.4.23`). Consider using the `constructor` keyword instead.

18. Add error strings to require statements

Since version `0.4.22` of solidity, `require` statements can include an error string. Consider adding appropriate error messages to the `require` statement.

AMENDED [17 May 2018]

ICOStart response, regarding issues 16 to 18:

Latest released version of truffle at the time the contracts were developed used solidity 0.4.21 and latest released version of zeppelin-solidity used 0.4.19, so we have used 0.4.21. No reason to update the toolchain at this point as there are no critical fixes - will be done for the next development cycle for sure.

19. Remove return variable from unlock function

The definition of `unlock` expects a boolean return value, but no value is returned. Consider removing the `returns` declaration or adding a return value.

AMENDED[17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645`.

20. Redundant function `isWhitelisted`

The function `isWhiteListed` in `LockableWhitelisted.sol` is not needed since whitelist variable is defined as `public` in `Whitelist.sol`

AMENDED [17 May 2018]

ICOStart response:

Kept for now because it’s already documented and used in tests. To be removed eventually.

21. Fix code comments to match implementation

The comments in cases such as function `getToken()` in `ICOStartReservation` do not match the code. Consider fixing comments to correctly describe the implemented function.

22. Remove unused code

Remove unused variables such as `INITIAL_SUPPLY` in `ICOStartToken.sol`

23. Remove duplicate SafeMath declaration

`SafeMath` is assigned to the type `uint` twice. It is recommended to remove the second assignment.

AMENDED[17 May 2018]

This issue has been fixed by the ICOStart team and is no longer present in commit `6f796f912995dd3e832d13f04be2bdb00c59e645`.

Closing Summary

Beyond the issues reported, the contracts were also checked for overflow/underflow issues, DoS, and re-entrancy vulnerabilities. None were discovered. The code was found to be well tested for many different scenarios.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of ICOStart or its products. This audit does not provide a security or correctness guarantee of the audited smart contracts. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

SOLIDIFIED

Audit Report for ICOStart. May 14, 2018.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

© 2018 Solidified Technologies Inc.