SOLIDIFIED

Audit Report for Coder Inc. February 25, 2018.

Summary

Audit Report prepared by Solidified for Coder Inc. covering token and crowdsale contract schemes.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below token sale. The debrief took place on February 25, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- CoderCoin.sol
- CoderCrowdsale.sol
- TokenTimelockMod.sol

Notes:

- The audit was conducted on commit a44b3420ceee32ef89f1e6ab7b4c10ad20482b3b
- The audit was based on the solidity compiler 0.4.20+commit.3155dd80

Intended Behavior

The purpose of these contracts is to create the CoderCoin and distribute it to the public, in a crowdsale process.

Issues Found

1. Token Decimal cases not considered in CoderCrowdsale.sol

The CoderCoin.sol contract defines that the token has 18 decimal cases, but the crowdsale contract doesn't take that into consideration when calculating token amounts to distribute. This leads to offering 10^{-18} less tokens.

Recommendation

When calculating token amount, multiply it by 10^{18} to account for decimal cases.

2. FinishCrowdsale() is exploitable by owner to mint more tokens than specified by the whitepaper

The provided intended behaviour states that an equal amount of tokens sold will be held by the owners, but the finishCrowdsale() allows for more tokens to be minted to the project. This happens because the founder percentage is added before the beneficiary distribution, making it so that the project gets the same amount as sold + founders percentage, violating the specification.

Recommendation:
Include the owner percentage in the beneficiaries share, so no more than what was sold in tokens is distributed.

3. Owner can still claim below the goal ether

If the presale does not reach its minimum goal, the owner can simply call the destroy() function and get the crowdsale ether, without needing to return the amount raised.

Recommendation
Remove the destroying mechanism or make it check if the goal has been reached.

4. Owner can avoid distributing bonuses

The arbitrary state changing from presale to ICO allows the owner to deny the bonuses specified in the behaviour.

How it works
As soon as presale begins, the owner can change the state to ICO and buyers are contemplated with the lower tier bonuses from the ICO(20%) instead of the correct range from the presale(between 75% and 30%)

Recommendation:
Consider allowing the state change to ICO only if a minimum amount was raised in the pre-sale. And make it so that it is not possible to revert to a previous state

5. Durations of sales are not enforced by contracts

The intended behaviour states that both the presale and the ICO have fixed duration, but neither of those values are enforced by the contract, making it possible for both to last forever (or as long as the owner wants). As per the document provided the duration for presale is 60 days and crowdsale is 30 days. It is recommended to add such restrictions to the contract.

6. Token Distribution specified is not enforced

The intended behaviour specifies a clear distribution of the 50% percent held by the team, but such scheme is not enforced by the contract and can be divided arbitrarily. Consider enforcing such behaviour.

7. Possible confusion of units between goal and totalCollected

The code and the comments in it give the impression that the 'goal' variable holds an amount in ether(eg 100), but is always compared with an amount in wei. This difference might lead to a very low goal.

8. Consider removing the destruction mechanism

When a contract is destruct from the blockchain, its bytecode is reverted to 0, but transactions made to this address are still accepted. That means that any amount transferred to the contract after its destruction is lost forever.
A better approach is to implement a soft ending mechanism, where no function can be called and sent ether can be rejected. Take a look at the Pausable.sol from open Zeppelin.

9. If presale minimum is not reached, buyers will still hold the tokens

If the presale goal is not reached, buyers can reclaim their invested ether, but will remain as token holders. The intended behavior does states what should be done in such scenario, but that might be problematic depending on the team decision.

10. Last transaction must be exact

If the contracts are close to reaching both the presale and ICO caps, the last transaction must be of the exact remaining amount, otherwise it will fail. A more user friendly approach is to accept any amount and give a change back to the buyer.

11. Make sure that the gas used is within the limit

CoderCrowdsale.sol

Some methods in the contract have a lot of logic to execute. Make sure that each execution will not cross the limited gas available.

Consider splitting the functionality into multiple methods to avoid gas limit issues.

12. Consider using view/pure

CoderCrowdsale.sol

Use `view` if your function does not modify storage and `pure` if it does not even read any state information.

Replace `constant` with `view` in methods.

13. Older non-fixed compiler version

CoderCrowdsale.sol | Line 1

The specified minimum compiler version is old. We recommend changing the solidity version pragma to the latest version (`0.4.20`) to enforce the use of an up to date compiler.

14. Consider using `memory`

CoderCrowdsale.sol | Line 115, 236, 274

`storage` can be replaced with `memory` if the values are not modified during use. It can help save some gas.

// Current usage

```
Bonus storage b = bonuses[i];
```

// Recommended usage

```
Bonus memory b = bonuses[i];
```

15. Install OpenZeppelin via NPM

OpenZeppelin's MIT license requires the license and copyright notice to be included if its code is used. Copying the files makes it difficult and error-prone to update to a more recent version.

Consider following the recommended way to use OpenZeppelin contracts, which is via the zeppelin-solidity NPM package. This allows for any bug fixes to be easily integrated into the codebase.

16. Consider removing `require(false)`

CoderCrowdsale.sol | Line 357

Consider replacing the `if-require` with a `require` or a `modifier`.

17. Replace `manager` with `_manager`

CoderCrowdsale.sol | Line 326

Wrong input validation in `setManager` method. Replace `manager` with `_manager`.

// Current implementation

```
require(manager != address(0));
```

// Recommended

```
require(_manager != address(0));
```

18. Unchecked math

CoderCrowdsale.sol

It is a good practice to use SafeMath throughout the contract.

19. Input validations

CoderCrowdsale.sol

Input validations are missing in some methods. Consider validating inputs such as address and other parameters in all methods.

Closing Summary

Several major and minor issues were found during the audit, some of which can break the intended behaviour or mislead the token buyers. It's strongly advised that these issues are corrected before proceeding with development. At the minimum, we recommend fixing the majors and minors, denoted by orange and yellow color codes.

Audit Report for Coder Inc. February 25, 2018.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Coder Inc platform. 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.

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.

Solidified Technologies Inc.