SOLIDIFIED

Audit Report for TEMCO. July 30th, 2018.

Summary

Audit Report prepared by Solidified covering the TEMCO token and crowdsale contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the below contracts. The debrief took place on May 3, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- SafeMath.sol
- ERC20.sol
- Ownable.sol
- Lockable.sol
- TemcoToken.sol
- Crowdsale.sol

Notes

The audit was performed on commit `6d7f4d30390a69c497c31bb3e42e5346423c7773`
The audit was based on the solidity compiler `c0.4.23+commit.124ca40d`

Intended Behavior

The purpose of these contracts is to create and distribute TEMCO tokens in a crowdsale mechanism.

Issues Found

Critical

1. Tokens cannot be distributed if there are many buyers

Tokens are distributed through `distributeCoin` with the help of `balanceList` array. If the array grows large enough, the gas requirement for the function can exceed the block gas limit. During the token distribution call, a large array will cause the iteration to run many times and burn a lot of gas. After a certain point this will exceed the block gas limit and the function can never be executed within a transaction. This can cause the functionality of distribution to fail.

Recommendation
It is recommended to calculate and distribute tokens (with lock period) to the buyer in real-time during the token purchase process OR implement a withdraw pattern where user can claim their tokens after the crowdsale.

Also, batch distribution with gas restriction can be implemented to distribute tokens. Another option is storing indexes in mappings and making the iteration offchain.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

Major

2. Owner of TemcoToken can transfer tokens on behalf of others

The owner of the token contract can transfer tokens on behalf of users, regardless of their consent, by calling `TransferFromWithLockup`. This function creates insecurity for the contributors, because they can loose the ownership of their tokens at any time.

Recommendation
Remove this function or restrict it to accounts that approved the spending, using one of the approve functions.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

3. Owner can deny distributing tokens

Owner needs to proactively call the function to distribute tokens and therefore may choose to never call it denying distribution of tokens to valid investors.

Recommendation
Add a function that allows investors to withdraw their purchased tokens regardless of the owner's actions.

Client's Response
The Temco team stated that this is a known trust point, and that they intend to perform the distribution on behalf of the token buyers. Users will be clearly communicated of this and other trust points present in the sale contracts.

4. Sale can reach the goal with invalid purchases

The crowdsale contract accepts and accounts bids regardless of whitelist status. It's possible that the majority of buys comes from non-whitelisted addresses, meaning that the sale wouldn't reach the goal if the the money is returned.

Recommendation
Check whitelist status before accepting purchases so all raised ETH is valid.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

5. There is no refund system for non-whitelisted purchases

If a non-whitelisted address sends money to the crowdsale contract, it is accepted and there's no way to easily retrieve it, even though he is not getting the tokens. That means that there is no guarantee that a refund will happen.

Recommendation
Implement a refund system(or refuse invalid purchases), for non-whitelisted addresses.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

6. Goal and sale duration is not enforced

Token buying process either checks for sale duration or goal, not both. This can allow the contract to accept payment even after hardcap is reached. User can also buy tokens if hardcap is not reached after the sale duration.

Recommendation
It is recommended to verify all conditions before proceeding with token buying process. It is also recommended to add a feature that can refund remaining change if a user sends excess amount right before reaching sale hard cap.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

7. User can transfer tokens during lock period

User can approve some other address to transfer tokens on their behalf and then call the `transferFrom` function. If the `msg.sender` has no tokens the modifier won't catch and the transfer will be made.

Recommendation
It is recommended to check the lock period for from address in `transferFrom` function to avoid token transfer. It is also recommended to lock the token transfer in common than handling it for individual users

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

Minor

8. Goal can be bypassed

The `onlyWhileOpen` modifier does not account for the value of the incoming transaction, therefore the hardcap can be bypassed.

Recommendation
The modifier should consider the incoming value of the transaction when comparing to the raising goal.

Client's response
The Temco team is aware of this possibility, and will refund the one buyer that reaches the limit manually after the sale.

9. Owner can change sale parameters at any time

Owners can modify some attributes such as conversion rate and goal, which can greatly affect purchases.

Recommendation

Consider only allowing the functions to be called during the stages preceding the auction start, so users can reliably know the conditions they're buying in. Most of those parameters are already defined in the intended behaviour, so there's no reason to alter them during the auction.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

10. Owner can mint tokens as required

The current mechanism allows owner to mint tokens as and when required. Since the contract can have multiple owners, anyone can exploit it easily.

Recommendation

It is recommended to remove minting functionality completely or to allow the contract to mint only during token distribution and call finish minting once tokens are distributed.

Client's Response

The mint functionality was included in the smart contracts due to a design decision, as Temco wants to ensure all buyers get their tokens. Once the sale is finished minting will be disabled with the use of the finishMinting function.

11. Unblocking an address can fail

If there are many addresses in the `kycBlockedMapList`, iteration can fail due to gas limit. This can cause user to not receive distributed tokens. After token distribution, if you unlock an account, there is no mechanism to distribute the tokens to that user.

Recommendation

It is recommended to follow a different data structure or mechanism to handle the KYC and whitelisting.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

12. Function distributeCoin won’t save progress

If there’s any error during the call of this function, the whole transaction will be reverted. This includes previous transfer functions as well as the variable `nextPayeeIndex`

Recommendation
There is no way to partially execute a function on Ethereum, therefore saving the progress isn’t useful in case of errors. Consider making this function accept a range (begin index and end index) for the iteration to be more useful.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

13. Bonus and mint features not implemented

The intended behavior mentions both a bonus scheme for early investors and that the minting of new tokens will be closed after sale, but neither of those functionalities are implemented in the contracts.

Recommendation
Implement such features in the contracts or remove them altogether from the intended behavior.

Client's response
The Temco team is aware of the reported issue, but chooses to perform the bonus payout manually. Although a valid decision, we strongly encourage the Temco team to implement the specs as they are defined as this ensures the outcome expected by users and mitigates the risk of human error during the sale, while also reducing the effort necessary to operate the smart contracts.

Notes

14. Transfer event called twice in transfer

The ERC20 transfer function is emitting the Transfer event twice for the same transaction. Consider removing the first call.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

15. Owner can remove themself making the contract unusable

Owners have the option to remove their ownership from the contract. If there is only one owner and if removed accidentally, most of the functions can become unusable. Consider adding a restriction for this.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

16. Wrong event raised during owner removal

Current implementation of `removeOwner` function in ownable emits a wrong event. Consider changing the event to the right one.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

17. Balance check can cause DoS

Consider checking the balance using `if` condition in `distributeCoin` function. If any user has zero balance, then the function will always fail causing a DoS to the user. This may not occur in the current implementation. But as per the recommendation for other issues, this condition may arise and it is recommended to avoid using require.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

18. Consider transferring ether for each sale

Consider transferring ether for tokens sold to the wallet for each buy. This can avoid storing the amount in the crowdsale contract and avoid any loss of funds in case of a failure.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

19. Handling allowance during transfer can be removed

Adding and removing allowance in the function `transferFromWithLockup` doesn't change any state or log event. This can be removed.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

20. Duplicate getter function for public variable

Public state variables will generate getter methods automatically. Avoid writing explicit getter methods or declare variables as private to avoid duplicate functions. Beware the generated getters won't be able to return full arrays, so if that return is needed, is recommended to manually write the getters.

AMEND[24.08.2018]
Temco team has fixed this issue in the latest code provided

Closing Summary

Several issues were found in Temco contracts, some of which are Critical and Major and can severely break intended behavior. All critical security issues were fixed, but Temco chose to maintain functions that allow central control of the contract in the name of user convenience.

These do not represent security vulnerabilities, but users should be advised that Temco can centrally pause the contracts, mint tokens and have to take actions in order for the execution of bonus and token payouts.

Beyond the issues mentioned, the contracts were also checked for overflow/underflow issues, DoS, and re-entrancy vulnerabilities. None were discovered.

OpenZeppelin contracts such as Ownable/SafeMath/etc. have been widely audited and secured, as such, they were not prioritized for auditing.

Disclaimer