SOLIDIFIED

Audit Report for MBAEX on January 16th, 2019.

Summary

Audit Report prepared by Solidified for MBAEX covering the StableCoin smart contracts (and their associated components).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on January 16th, 2019, and the final results are presented here.

Audited Files

The following contracts were covered during the audit:

- StableCoin (published contract, with multiple contracts that compose it)

Notes

The audit was based on the Rinkeby testnet contracts provided at the following addresses:

- StableCoin:
  - https://rinkeby.etherscan.io/address/0x0EaB63161716e2Ea0c0d58B1FdaBFE636Aef9d10#code

Intended Behavior

1. From provided documentation: StableCoin is a cryptocurrency that allows users to make use of digital currency in the same way they use fiat currency. StableCoin enjoys the same value that legacy currency has, as a store of value, a medium of exchange, and is considered as a means of investment. StableCoin is intended to be backed 1:1 to USD.

Issues Found

1. Controllable: A previously frozen new owner account cannot be unfrozen (minor)

Since the `unfreezeAddress()` function restricts allowed addresses to non-owner accounts only, a previously frozen account that becomes the new owner of the contract cannot be unfrozen unless they relinquish contract ownership.

Recommendation
Override the `transferOwnership()` function to make sure a new contract owner is not a frozen address.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

2. Controllable: Function destroy() allows contract owner to confiscate the tokens of all token holders (minor)

In its current implementation, `destroy()` essentially prevents token holders from assuming any reasonable amount of ownership over the USD pegged tokens they have.

Recommendation
`destroy()` should not be able to destroy the contract until `totalSupply` is equal to zero.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

3. StableCoin: approveAndCall() is unexpectedly increasing approved amount instead of only setting it (minor)

Users calling approveAndCall() will be expecting it to assign an allowance of their given amount to their given spender address, and **not** add to the spender's already existing allowance.

Recommendation
Call approve() instead of increaseApproval() in the function's implementation.

For reference, please check the approveAndCall() standard implementation in eip-20.md and the ERC20 API: An Attack Vector on Approve/TransferFrom Methods.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

4. StableCoin: decreaseSupply() allows contract owner to confiscate tokens from any account (note)

In its current implementation, decreaseSupply() essentially prevents token holders from assuming any reasonable amount of ownership over the USD pegged tokens they have.

Recommendation
Designate a burn address (or a group of addresses) that the token holders would send the tokens to in order to redeem their pegged USD. The owner of the burn address(es) would then call the burn() function in order to burn the tokens and decrease the total supply.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

5. StableCoin: Fallback function is redundant (note)

Starting from Solidity 0.4.0, contracts without a fallback function automatically revert payments, making the code redundant.

Recommendation
As it is not utilized, this code can be removed. The contract will reject payments automatically.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

6. StableCoin: burn() function not needed for all token holders (note)

It is highly unlikely that a token holder would need to burn their own USD pegged tokens. Having this function open to the public can potentially pose unnecessary risks for all token holders.

Recommendation
Limit the `burn()` scope to a limited number of accounts whose function is to burn tokens received from other holders (please see the above `decreaseSupply()` recommendation for reference).

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

7. Ownership: Variable name newOwner is confusing (note)

Variable `newOwner` holds the value of an owner pending acceptance, and not the actual new owner of the contract.

Recommendation
Consider renaming `newOwner` to `pendingOwner`.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x96c9cC5Df02a6E5F2987006cC1466B0BF1b6B8CF` (Rinkeby).

SOLIDIFIED

Closing Summary

MBAEX’s contracts contain a few minor issues, along with several areas of note.

We recommend the minor issues are amended, while the notes are up to MBAEX’s discretion, as they mainly refer to improving the operation of the smart contract.

Disclaimer