SOLIDIFIED

Audit Report for MBAEX on January 16th, 2019.

Summary

Audit Report prepared by Solidified for MBAEX covering the MultiSigRoot 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:

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

Notes

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

- MultiSig:
  - https://rinkeby.etherscan.io/address/0xeaefa4027a1dff7d517c57705f3e74e4f9eab1ff#code

Issues Found

1. MultiSigRoot: onlyWallet modifier can be circumvented (minor)

The onlyWallet modifier works by checking if the given wallet address is registered as type WALLET_TYPE_WALLET. Since this type is set when the wallet is first attached to the node, it can be circumvented in case a contract address that has no contract deployed yet is precomputed and attached as a wallet, then a contract is deployed to the same address at a later stage when WALLET_TYPE_WALLET has already been set.

Recommendation
Instead of relying on checking WALLET_TYPE_WALLET, call the isContract() function directly in the modifier.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

2. MultiSigRoot: MultiSigRoot should not be accepting ETH (minor)

MultiSigRoot has a payable default function, so it can accept Ether transfers.
However, the MultiSigRegulator contract will fail to create a requirement for any transaction which has Ether _value > 0, since the call(..).value(…) function at L570 will fail.

Recommendation
If the described above is an intended behavior, it would be better to explicitly reject transactions transferring Ether value rather than relying on implicit failure (due to MultiSigRegulator's non-payable functions and empty balance)

If MultiSigRegulator is changed to ignore the _value at L570, then any STAKER_CONTROLLER could transfer out Ether from MultiSigRoot by crafting a transaction to some external contract's payable `pause()` function, since this function requires only one `STAKER_CONTROLLER` wallet's approval.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

3. `MultiSigRoot`: `flag` bits do not cover all node wallet indices (minor)

Only 16 `flag` bits are ever set, leaving the rest of the 64 `MAX_WALLET` flags unassigned. If there is any wallet within a node which is created at an index greater than 16, that wallet will not be allowed to sign a transaction – as a consequence the transactions which require all node's wallets approval could never be executed.

Recommendation
Either set `flag` to `2^64-1` instead of `65535`, or remove flags all-together since all flags are presently set to `1`.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract `0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

4. `MultiSigNode`: 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 `0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

5. MultiSigNode: onlyParent modifier is redundant (note)

The onlyParent modifier is not used throughout the scope of the contract.

Recommendation
As it is not utilized, this code can be removed.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract
`0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

6. MultiSigRegulator: Not all StableCoin's functions are implemented in MultiSigRegulator (note)

StableCoin's burn() function is not implemented in MultiSigRegulator and do not define any requirements for execution – thus they cannot be executed from the MultiSigRoot.

Recommendation
These functions should either be implemented or otherwise augmented so that they can be explicitly executed from the MultiSigRoot.

7. MultiSigRegulator: Transaction Limits are not meaningful when transferring other Tokens (note)

MultiSigRegulator allows any token transfer out of MultiSigRoot (because it does not check the destination address of a transaction) as long as the call function matches the ERC-20 standard. If that is intentional, the transaction limit amounts are not very meaningful, since other Tokens might have different USD value and different number of units (decimals) than the StableCoin Token.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract
`0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

8. MultiSigRegulator: Avoid excess computation (note)

Currently Node Labels (e.g. “STAKER”, “STAKER_CONTROLLER”) are defined in multiple places (MultiSigRegulator, MultiSigRoot), and keccak256 is calculated multiple times in every transaction.

Recommendation
Consider moving all Node Class/Labels definitions in a single contract (either Label or a new contract inheriting from Label).

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract
`0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

9. Transaction requirements do not fully match the document provided for Staker and Staker Controller (note)

Transaction requirements do not fully match the document provided for Staker and Staker Controller, namely:

- cannot “disallow to add/remove in Staker group” - document states it can
- can add/remove Signer Controller - documents does not state it can
- can set Transaction Limit - document does not state it can

Recommendation
Augment the presently existing documentation to accurately reflect the transaction requirements.

Amended [24-01-2019]
The issue was fixed and is no longer present in the contract
`0x75D27Fe3Fd376CC21E958bA12FEC53081F01Cb15` (Rinkeby).

SOLIDIFIED

Audit Report for MBAEX on January 16th, 2019.

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

SOLIDIFIED

Audit Report for MBAEX on January 16th, 2019.