SOLIDIFIED

Audit Report for iComply Investor Services Inc. March 19, 2018.

Summary

Audit Report prepared by Solidified for iComply Investor Services Inc. covering the token and crowdsale contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below token sale. The debrief took place on March 19, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- CompliantCrowdsale.sol
- CompliantToken.sol
- Validator.sol
- WhiteListContract.sol

Notes

The audit was performed on commit `40a61985433ccc54ac83f7feeb3c7beb4caf82c7`
The audit was based on the solidity compiler `0.4.21+commit.dfe3193c`

Issues Found

Critical

1. Token can be transferred without a fee

The `CompliantToken` contract inherits from OpenZeppelin's `MintableToken`, which again inherits from `StandardToken`. In the `StandardToken` contract, functionality exists for letting addresses transfer tokens on behalf of other addresses. This means that the whole mechanism of taking a fee per transfer can be circumvented - instead of using \`transfer\` directly and having to pay a fee, an investor can just give an address an allowance of tokens and then use the `transferFrom` function.

Recommendation

It is recommended to override the `transferFrom` function in `CompliantToken` and add fee logic like in `transfer` method to enforce fee for all token transfers.

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

Major

2. Validator approval can be bypassed during token transfer

As per the requirement document, every token transfer should be approved by a validator. This holds true for transfer function. ERC20 also has a `transferFrom` method and the validation is not enforced there. It allows the user to transfer tokens without any approval from the validator.

Recommendation

It is recommended to override the `transferFrom` function in `CompliantToken` and add approval logic.

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

3. Non-whitelisted addresses can hold tokens

Addresses that are not whitelisted can receive tokens and transfer them to another address. Such restrictions can be bypassed by making use of `transferFrom` method defined in the token. Check for whitelisted address is not performed in the method.

Recommendation

It is recommended to override the `transferFrom` function in `CompliantToken` and add validation for whitelisted addresses.

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

Minor

4. Whitelist contract is not initialized in token

As per the document shared, the white listing contract should be initialized during the token creation. Such initialization is not present in the constructor of `CompliantToken` contract. Also, `fee` and `feeRecipient` are not set in a constructor, but in setter methods. This means that it is possible to initialize the contract in such a state where calling other methods will always fail.

Recommendation

It is recommended to initialize white listing contract and other parameters in the constructor of the token itself.

```
function CompliantCrowdsale(address whitelistAddress, address recipient,
uint246 fee) public onlyValidator {
    setWhitelistContract(whitelistAddress);
    setRecipient(recipient);
    setFee(fee);
}
```

If the `whitelistAddress`, `fee` and `feeRecipient` is not intended to be changed later, mark those methods as `internal`.

AMENDED [8 May 2018]
iComply Investor Services Inc. team has added `setWhitelistingContract` to the constructor in the reviewed commit `3cd3b2e73865625105dfc071937df6acacc08099`.

5. Approving transfers from `feeRecipient` will emit event with incorrect values

When a transaction made from the `feeRecipent` address is approved, a `TransferWithFee` event is emitted. However, this event does not contain the correct values, as it always uses the the values for `pendingTransactions[0]` instead of the transaction which is recently approved.

```
TransferWithFee(pendingTransactions[0].from,
                pendingTransactions[0].to,
                pendingTransactions[0].value,
                0);
```

Recommendation

Replace the parameters with correct values

```
TransferWithFee(pendingTransactions[nonce].from,
                pendingTransactions[nonce].to,
                pendingTransactions[nonce].value,
                0);
```

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

6. It is possible to "overdraw" the account

If an investor has a balance of 100 tokens as an example, it is possible for him to call `transfer` with 100 tokens twice, given that no one approves the first transfer before the second is made. When `validator` calls the `approveTransfer` function for the second transfer (given that a transfer transferring enough tokens TO the same address isn't approved in the meantime), that approval will fail, so it isn't actually possible to send more tokens than you have.

However, since `transfer` reverts when the sender does not have sufficient balance, it should probably have some kind of logic in place for checking that the transfer amount is <= investors balance + fee + investor's yet unapproved transactions.

Recommendation
Add logic for checking that the transfer amount is <= investors balance + fee + investor’s yet unapproved transactions.

AMENDED [8 May 2018]

iComply Investor Services inc. team has added a mapping `pendingApprovalAmount` to keep track of the amount in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

Notes

7. Consider checking if `nonce` exists explicitly

`require(pendingTransactions[nonce].to != address(0))` in `CompliantToken #approveTransfer` and `require(pendingMints[nonce].to != address(0))` in `CompliantCrowdsale#rejectMint` are really checking for that there exists a struct with that nounce. This could be written more explicitly in the code (for instance by extracting into a function with a descriptive name) to improve readability.

8. Consider enforcing token ownership transfer

Ownership transfer of the token contract during the sale start and end process is performed manually. It is recommended to automate such transfers in the contract itself.

AMENDED [8 May 2018]
This suggestion has been partially incorporated by the iComply Investor Services Inc. team in commit `3cd3b2e73865625105dfc071937df6acacc08099`. Token ownership is now transferred back to the owner on finalization.

9. Token supply is unlimited

There is no restrictions on the number of tokens that can be minted or sold. The owner can mint as many tokens as he wants at no cost. It is recommended to restrict the number of tokens that can be sold.

10. Consider using withdrawal pattern

While rejecting a request for buying tokens, it won't always work since ether transfer can be rejected by the target address. More details can be found here.

AMENDED [8 May 2018]

This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

11. Using events without emit is deprecated

Using events without `emit` is deprecated in the latest version of solidity. Consider using this pattern when logging an event in solidity.

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

12. Consider using latest version of solidity

The specified minimum compiler version is a bit old. We recommend changing the solidity version pragma to the latest (`0.4.21`).

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`. The current version used is `0.4.23`.

13. Consider using NPM to manage OpenZeppelin

OpenZeppelin contracts were copied from the repository. NPM can be used to manage the library which can help in maintaining and updating the library more easily. Also, consider that even NPM itself can be an attack vector sometime.

14. Use SafeMath for arithmetic operations

In the `transfer` method, a `+` is used to check if the sender has a balance high enough to cover both the transfer itself and the fee for it. It is strongly recommended to replace this with `SafeMath.add` to make sure all arithmetic invariants are valid.

Replace this

```
require(_value + transferFee <= balances[msg.sender]);
```

with this

```
require(_value.add(transferFee) <= balances[msg.sender]);
```

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services Inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

15. Duplicate code can be avoided

The `approveTransfer` function has a lot of duplicated code, which can easily be remedied with refactoring. Removing duplication would reduce risk for oversights when changing the code, and improve readability.

16. Consider using `external`

Consider using `external` for function visibility if the method will only be accessed from outside. This can help save some gas.

AMENDED [8 May 2018]
This issue has been fixed by iComply Investor Services inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

17. Consider renaming the `reference` parameter in `CompliantCrowdsale` constructor

The `reference` parameter in the `CompliantCrowdsale` constructor could be better named. It is the address that references the `Whitelist`, but this is not clear from the name. Consider renaming to something more descriptive, for instance `whitelistAddress`.

AMENDED [8 May 2018]

This issue has been fixed by iComply Investor Services inc. team and is not present in commit `3cd3b2e73865625105dfc071937df6acacc08099`.

Closing Summary

Several critical, major and minor issues were found during the audit which could break the intended behaviour. iComply Investor Services inc. team has addressed most of the issues in the report, as noted by the amendments made. It is furthermore recommended to post the contracts on public bounty following the audit.

Disclaimer

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

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.