SOLIDIFIED

Audit Report for iComply v2 October 10, 2018.

Summary

Audit Report prepared by Solidified for iComply v2 covering the token and crowdsale contracts.

Process and Delivery

Independent Solidified experts performed an unbiased and isolated audit of the below token sale. The debriefing took place on October 10, 2018, and the final results are presented here. Report was updated at December 11, 2018, with focus on new features.

Audited Files

The following files were covered during the audit:

- Crowdsale (no cap, hard cap, token cap) contracts
  - ComplaintCrowdsale.sol
  - Crowdsale.sol
  - FinalizableCrowdsale.sol
  - ComplaintCrowdsaleHardCap.sol
  - ComplaintCrowdsaleTokenCap.sol
- Token contracts
  - CompliantToken.sol
  - CompliantTokenEthFee.sol
  - CompliantTokenRemintableEthFee.sol
  - CompliantTokenRemintable.sol
  - CompliantTokenSwitchEthFee.sol
  - CompliantTokenSwitchRemintableEthFee.sol
  - CompliantTokenSwitchRemintable.sol
  - CompliantTokenSwitch.sol
  - BasicTokenEthFee.sol
  - ERC20EthFee.sol
  - ReMintableTokenEthFee.sol
  - ERC20BasicEthFee.sol
  - MintableTokenEthFee.sol
  - StandardTokenEthFee.sol
  - ReMintableToken.sol
  - Whitelist.sol
  - WhitelistedToken.sol

SOLIDIFIED

Audit Report for iComply v2 October 10, 2018.

Notes

The audit was based on the commit hash `f94ae0f829d347ba8f2228c17156b30fea9ade3e`, update based in commit `848c27d5464ea3d92806dadac84e26ad1386c846`, Solidity compiler `0.4.25+commit.59dbf8f1`

Follow up

Minor

1. No way to specify the whitelist contract address

The Whitelisted token has no method to specify the whitelist contract address referred in the BasicToken contract. The current implementation points it to an empty address.

Recommendation

It is recommended to use the constructor to specify the whitelist contract. This will ensure that the modifier checks the permission with required whitelist contract.

Amended [09-01-2019]

The issue was fixed and is no longer present in commit `bf7a6b7dd07e73c6e6f8c5e2145b5dbfaf672a67`.

2. Use enums to effectively represent reject reasons (note)

If there are a finite number of predefined reasons to reject a transaction, it is better to represent them as part of an enum. This will avoid the confusion which may arise if random numbers are used to represent the reason.

Auditee's response:

"We are intentionally not using enums for Reject Reason as we intend to embed a fair amount of transaction-specific rejection data in that data field."

Issues Found

Minor

1. ETH fee might get stuck in the contract

Fee recipient can do a transfer with the fee and lose it forever. This is possible since there is no validation that checks whether the fee recipient address is sending any ETH as a fee along with the transfer. This can cause the fee to get stuck in the contract.

Recommendation
It is recommended to restrict the fee recipient from sending any ETH along with the transfer. This can be done by including a `require` statement in the `isFeeNeeded` modifier.

Amended [22-10-2018]
The issue was fixed and is no longer present in commit `3bcbef7d0e19364454ddf4935405f85042ee4e72`.

2. Rejecting a transfer can cause DoS

In the EthFee token contracts, rejecting a token transfer returns the ETH fee collected using the `transfer` method. This will work fine if the sender is an EOA.
However, a malicious sender can use an contract to do the transfer which will also deny any incoming ETH. This will cause the _rejectTransfer function to fail for that specific address.

Recommendation
It is recommended to use withdraw pattern while rejecting a token transfer. Since this issue will not affect any other user except the sender itself, leaving it as such is also an option.

Amended [22-10-2018]
The issue was fixed and is no longer present in commit `3bcbef7d0e19364454ddf4935405f85042ee4e72`.

3. Changing the fee recipient can cause inconsistencies

If some address with pending transactions is added as a fee recipient or vice-versa the pendingApprovalAmount mapping will have incorrect values. This is due to the addition and subtraction of fee amount based on the current fee recipient.

Recommendation
It is recommended to not allow modifications to the fee recipient if there are any pending transactions in the queue.

Amended [22-10-2018]
The issue was fixed and is no longer present in commit `3bcbef7d0e19364454ddf4935405f85042ee4e72`.

4. Allowing contract address updates during crowdsale is not recommended

The token contract and whitelist contract can be changed to a different one during any point in time. It is recommended not to allow the change of these contracts during the crowdsale process to increase the trust in the system.

Auditee's Response
"We decided not to change the functionality around this one. IT's an upgradability feature, and the responsibility ultimately falls on the ICO Issuer to only use it when appropriate."

5. Token sale can fail due to insufficient permission

The contracts for crowdsale expects the token owner to be the contract itself. If it was not assigned before the minting process in `buyTokens` function, the process will fail. Verification of the token ownership during the initialization itself can be a good idea.

Auditee's Response
"There doesn't appear to be any meaningful place to put that check unfortunately. We decided the problem was minor enough not to complicate things further to address this edge case."

6. Gas recommendations

The following methods can be followed to better optimize the gas usage. Using them is subject to the requirements and needs.

- DetailedERC20 - Using hardcoded public CONSTANT values instead of setting state variables can save some gas.
- Duplicate code can be avoided to save a lot of gas. Consider moving those to the parent or helper contracts to increase reusability.

Auditee's Response
"Updating variables use a negligible amount of gas to update.
Duplicate code is written only in the repo. While deploying the contracts for a single crowdsale and token, no code is repeated so there will be no additional gas usage."

7. Fee representation is wrong

During the token transfer method, the mapping `pendingTransactions` and the event `RecordedPendingTransaction` stores the transfer fee as one of the parameters. This wrong when a fee recipient performs the transaction with zero fees. It is recommended to update the parameters accordingly.

Amended [22-10-2018]
The issue was fixed and is no longer present in commit `3bcbef7d0e19364454ddf4935405f85042ee4e72`.

8. Token ownership can be transferred before finalization

Transferring token ownership is possible before finalization and doing it will render the `finalization` function useless. It is recommended to restrict this functionality if it is not intended.

Auditee's Response
"After some internal debate, we decided not to change this functionality. This is essential for fixing any issues that might arise during a crowdsale - errors in the crowdsale code, wrong

parameters being set, etc. We wouldn't be able to fix those issues otherwise without restarting the entire crowdsale and token."

9. Token renaming

The functions `updateName`/`updateSymbol` are not compliant with the ERC20 standard. It may cause investor distrust or problems with listing on an exchange.

Auditee's Response
"There have been projects that changed their token name and symbol in the past, both on the smart contract and exchange side of things, without any problems."

Audit Report for iComply v2 October 10, 2018.

Closing Summary

Several issues were found during the audit which could break the intended behaviour. It is recommended for the iComply Investor Services inc. team to address the issues. It is furthermore recommended to post the contracts on public bounty following the audit.

Follow up [12-11-2018]

One minor deficiency was identified in the contracts' new features, as well as an informational issue. We recommend the minor issue is amended before production use.

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.

© 2018 Solidified Technologies Inc.