Gifto Audit Report

Preamble

This audit report was undertaken by BlockchainLabs.nz for the purpose of providing feedback to Gifto. It has subsequently been shared publicly without any express or implied warranty.

Solidity contracts were sourced from the public Github repo gifto-io/GiftoSmartContract prior to commit 50e1f1895dd91a0dad0d4f0b2ea620e5827ed1fa - we would encourage all community members and token holders to make their own assessment of the contracts.

Scope

All Solidity code contained in /contracts was considered in scope along with the tests contained in /test as a basis for static and dynamic analysis.

Focus Areas

The audit report is focused on the following key areas - though this is not an exhaustive list.

Correctness

- No correctness defects uncovered during static analysis?
- No implemented contract violations uncovered during execution?
- No other generic incorrect behaviour detected during execution?
- Adherence to adopted standards such as ERC20?

Testability

- Test coverage across all functions and events?
- Test cases for both expected behaviour and failure modes?
- Settings for easy testing of a range of parameters?
- No reliance on nested callback functions or console logs?
- Avoidance of test scenarios calling other test scenarios?

Security

- No presence of known security weaknesses?
- No funds at risk of malicious attempts to withdraw/transfer?
- No funds at risk of control fraud?
- Prevention of Integer Overflow or Underflow?

Best Practice

- Explicit labeling for the visibility of functions and state variables?
- Proper management of gas limits and nested execution?

- Latest version of the Solidity compiler?

Classification

Defect Severity

- Minor - A defect that does not have a material impact on the contract execution and is likely to be subjective.
- Moderate - A defect that could impact the desired outcome of the contract execution in a specific scenario.
- Major - A defect that impacts the desired outcome of the contract execution or introduces a weakness that may be exploited.
- Critical - A defect that presents a significant security vulnerability or failure of the contract across a range of scenarios.

Findings

Minor

- **Tokens are not automatically transferred to investors** - `Best practice` To receive a token, the investor must `buyGifto`, and then the owner of the contract must run the `deliveryToken` function to send the investor tokens. This is a manual process so there is no guarantee for the investor that they will receive their tokens other than trust. View on GitHub
  - ☐ Not Fixed
- **Removal of setMinimumBuy without removing relevant variables** - `Best practice` We would recommend that if the intention is not to make use of the removed function setMinimumBuy, that you remove these unused variables. View on GitHub
  - ☐ Not Fixed
- **The 'onlyNotOwner' modifier is not used in the contracts** - `Best practice` There is a modifier `onlyNotOwner` which is not used by any function, this can be removed. View on GitHub
  - ☐ Not Fixed

- **Use .transfer instead of .send** - `Best practice` This is a very minor issue because `.send` is still value, but `.transfer` has a richer interface and allows you to override the gas limit, which `.send` does not. There is some discussion on `.send` vs `.transfer` here: View on GitHub
  - ☐ Not Fixed
- **Explicitly declare your variable types** - `Best practice` `uint` will default to `uint256` but it is recommended to explicitly declare it as `uint256` View on GitHub
  - ☐ Not Fixed
- **Explicitly declare your variables access modifiers** - `Best practice` You should explicitly declare `public` on the variables that are meant to be `public`. This can help to avoid errors, but it can also cause unexpected behaviour. View on GitHub
  - ☐ Not Fixed
- **Convention is to use capital letters for the token "symbol"** - View on GitHub
  - ☐ Not Fixed
- **Format repository to follow standard convention (add folders, separate files)** - `Best practice` We strongly recommend restructuring the files in your repo to follow conventional approach of other token launches. This is so that relevant files can be more easily found and increases trransparency See here for examples: View on GitHub
  - ☐ Not Fixed
- **Comment needs updating to reflect commit** - `Best practice` As a result of the update in the following line #44 we would recommend also updating the comment to reflect this accordingly for added clarity. View on GitHub
  - ☐ Not Fixed
- **The 'validInvestor' modifier is not used in the contracts** - `Best practice` There is a modifier `validInvestor` which is not used by any function, this can be removed. View on GitHub
  - ☑ Fixed 02ca7007

Moderate

- **Missing SafeMath Library** - `Best practice`, Correctness` For calculations we recommend using SafeMath.sol http://zeppelin-solidity.readthedocs.io/en/latest/safemath.html This ensures against and prevents the unsigned integer overflow issue. View on GitHub
  - ☐ Not Fixed

Major

- **Token does not follow ERC20 Token Standard - Missing `approve` function** - `Correctness` The token standard can be seen here: https://theethereum.wiki/w/index.php/ERC20_Token_Standard Tests created for ERC20 Standard: The `approve` function must be implemented for the Gifto token to be compatible with ERC20. Failing to meet the ERC20 Token Standard can mean you won't get accepted on exchanges and may be incompatible with some Ethereum wallets. View on GitHub
  - ☑ Fixed 84602fbe
- **function createCoin() should not be allowed to be called more than once** - `Correctness` We recommend adding a modifier so that createCoin() cannot be called by anybody more than once. This ensures that the total supply cannot be increased anymore than what is originally minted. View on GitHub
  - ☑ Fixed 63108870

Critical

- None found

Test Results

Test Coverage


```
--------------------------|----------|----------|----------|----------|------------
----|
File                      |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered
Lines |
--------------------------|----------|----------|----------|----------|------------
----|
 contracts/               |    41.28 |    25.64 |    50.82 |    40.61 |
|
  ERC20Interface.sol      |      100 |      100 |      100 |      100 |
|
  GiftoCrowdsale.sol      |    94.67 |     62.5 |    91.18 |    93.02 |...
,83,408,409 |
```

```
  GiftoMultisigWallet.sol |        0 |        0 |        0 |        0 |...
369,370,371 |
--------------------------|----------|----------|----------|----------|------------
----|
All files                 |    41.28 |    25.64 |    50.82 |    40.61 |
|
--------------------------|----------|----------|----------|----------|------------
----|
```

```
.-------------------------------------------------------------------------------
----------------------.
|                                    contracts/GiftoCrowdsale.sol
|
|-------------------------------------------------------------------------------
---------------------|
| Function                           | Constant |  Returns  |
Modifiers               |
|------------------------------------|----------|-----------|------------------
---------------------|
| ()                                 | false    |           | payable
|
| buyGifto()                         | false    |           |
payable,onSale,validValue,validInvestor |
| Gifto()                            | false    |           |
|
| totalSupply()                      | true     | uint256   |
|
| turnOnSale()                       | false    |           | onlyOwner
|
| turnOffSale()                      | false    |           | onlyOwner
|
| setIcoPercent(uint256)             | false    |           | onlyOwner
|
| setMaximumBuy(uint256)             | false    |           | onlyOwner
|
| setBuyPrice(uint)                  | false    |           | onlyOwner
|
| balanceOf(address)                 | true     | uint256   |
|
| isApprovedInvestor(address)        | true     | bool      |
|
| getBuyers()                        | true     |           |
|
| getDeposit(address)                | true     | uint256   |
|
| addInvestorList(address)           | false    |           | onlyOwner
|
| removeInvestorList(address)        | false    |           | onlyOwner
|
| deliveryToken(uint,uint)           | false    |           |
onlyOwner,validRange                |
| transfer(address,uint256)          | false    | bool      |
|
| transferFrom(address,address,uint256) | false    | success   |
|
| approve(address,uint256)           | false    | success   |
```

```
|
| allowance(address,address)            | true     | remaining |
|
| withdraw()                            | false    | bool      | onlyOwner
|
'-------------------------------------------------------------------------------
----------------------'
```

Gas Consumption

Contracts were assessed on the gas usage of each function to ensure there aren't any unforeseen issues with exceeding the block size GasLimit. A detailed report can be found in Gas_Consumption.md.

Addendum

Upon finalization of the contracts to be used by Gifto, we have diligently enumerated each function within the contracts including static and dynamic analysis. Deployment testing results can be viewed at Kovan_Tests.md

We have reviewed this document to ensure that there are no ommisions and that the developers' comments are a fair summary of each function.

Conclusion

The developers demonstrated an understanding of Solidity and smart contracts. They were receptive to the feedback provided to help improve the robustness of the contracts.

We would have preferred to see more follow through on resolving minor issues and focus on following best practice prior to the deployment and operation of these contracts.

We took part in carefully reviewing all source code provided, including both static and dynamic testing methodology. We were also required to create a test suite using the Truffle Framework to fully satisfy coverage in all areas.

Overall we consider the resulting contracts following the audit feedback period adequate and have not identified any potential vulnerabilities.