Initial Report

Ahmed Ali

June 2017

Contents

0.1 Summary

In the course of auditing the code, no major security flaws were found and what follows are a listing of minor issues and possible criticisms. In general, it is well structured, documented and tested; and so this audit does not attempt to cover possible refactors. Given community response to other ICOs I believe it is important to point out that this ICO is trustful of the multisig and in some setups of individual owners.

1 Inherited Contracts

1.1 SafeMath.sol

Used throughout the other contracts, the best practice for serious contracts.

1.2 VestedToken.sol

While this contract has no major security flaws, it cannot fully be cloned due to revokeGrants erasing grants from history entirely; `transferableTokens` cannot be relied on for a fork

1.2.1 Notes

214 says revoked, while this function merely checks.
65 is not necessary given that the controller is non-transferable from the ICO contract, a claim made there that has a minor flaw.

1.3 MiniMeToken.sol

At line 209, the token implements a workaround for a race condition that is still abusable if the user does not invest the time needed for multiple confirmations. It would be preferable to implement an increment or compare and swap solution though this can be handled by a wrapper contract.
At line 459, there is a reimplementation of the min function and it is used in the contract despite it generally using `SafeMath`.
A minor possible issue is that the ICO contract does not have a way to enable transfers, and so it must be created with transfers enabled.

1.4 LimitedTransfer.sol

1.5 ERC20.sol

A slight modification of the well tested ERC20 that adds additional user-protecting checks; A welcome change.

1.6 ERC20Basic.sol

1.7 Pausable.sol

2 TokenController.sol

3 Shareable.sol

If `confirmAndCheck` is used to protect an operation that can be nullipotent, a malicious shared owner can prevent operation. This is protect against through the use of `keccak256(msg.data)` to wrap functions that do not

depend on the state.
At line 84, the use of nonimplication(`A&-B`) instead of subtraction would be preferable, as it can strictly only unset the bit independent of the state of the bitmap.

3.0.1 Notes

At line 137, using an `==` comparison would further clarify the code-paths.
107 says set, means check.

4 District0xContribution.sol

`setDistrict0xNetworkToken` trusts each of the owners to not act maliciously, at any point in time; I would suggest that such a function be also protected by shareable.
`getRunningContribPeriod` fails to return a sensible value if a period is enabled then canceled. This leads to possible issues such as changing max gas or the minimum contribution being disabled despite no period actually running, as well as the collection of ether for which no tokens can be distributed.
There was a hard cap issue that was resolved; with a single large order disabling the hard cap. This could have led to absurd amounts of ether accumulating in the multisig due to the long tail of ICOs.
At line 116, as contribution periods can overlap and this function depends on the current timestamp there can be a small window where miners can divert funds to whichever period they please. This can be avoided by simply not creating overlapping periods, or switching to blocknumbers instead of timestamp which would have been the advisable solution but at this point might be too much of a refactor.
At line 267, as the length is limited to `CONTRIB_PERIODS` by `setContribPeriod`, and the only path in which `getRunningContribPeriod` can return a value larger than `contribPeriods.length` is when it defaults, it would be preferable to use `CONTRIB_PERIODS` here instead as it simplifies analysis and verification.
At line 176, while it is generally preferable to delay transfers to the end of a function call, transfer is generally considered safe as it does not forward a meaningful amount of gas nor is a contract paying for its own gas on the near horizon. Additionally, once this code path is entered, every other call to contribute will throw near the top; So this can be considered safe.
`compensateContributors` allows owners to censure specific contributors, which further solidifies that this is a trustful ICO.
At line 378, and above the function in the documentation there are checks to

prevent regeneration of the supply, however it is still possible to provide the ICO with a much larger (or smaller, arbitrary) token count and distribution as the MiniMe token can be created, assigned a different controller, then have the controller reassigned to the ICO.

4.0.2 Notes

119 & 122 the hard cap sets the end time, so there is no need to check both. 214 multiplies by a large number, then later divides by it. I do not understand the purpose of this; if the intention is to achieve a higher accuracy for ratio then it does not do this as the division removes any gained accuracy.

5 District0xNetworkToken

As mentioned above, the controller can be changed after construction.