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

- Functions should be explicit with their access levels, include public/private for each function
- ☑ Fixed
- Consistency in integer types, e.g uint _amount, uint is used in some places, while uint256 is used in others. uint defaults to uint256 but being consistent and explicit would be best.
- ☑ Fixed
- //TODO: change this comment should be removed, is there some missing functionality that still needs to be implemented?
- ☑ Fixed
- // croudsale statuses small typo in this comment

- ☑ Fixed
- Discrepancy between ether and finney value, //require msg.value >= 0.1 ether require(msg.value >= 10 finney); If you wanted to use 0.1 ether, we wanted to let you know that 10 finney is not equal to 0.1 ether. it's actually 0.01 ether. According to your comments, it's what you intended to declare.
- ☑ Fixed
- canWithdraw naming convention. Usually anything named canXXX, onlyXXX means declaration of a modifier, but in this case it's a variable. We would highly recommend renaming to follow industry conventions.
- ☑ Fixed

Moderate

- Token Sale logic is highly coupled with Token Contract itself, although you can do it this way, we would highly recommend decoupling those 2 separate intentions. You could have a token contract and crowdsale contract which you can manage separately.
- ☑ Fixed

Major

- function withdraw(address participant) { ... //participant.transfer(share);withdraw should be a constant function, because it's called from a constant function. There will need to be an other function that isn't constant, which can be used to change state/withdraw tokens. There is a commented out line which would have allowed token transfer.
- ☑ Fixed

Critical

- None found