Beam Smart Contract Audit Report

Preamble

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

Solidity contracts were sourced directly from the HubbleLand team at this commit hash 11b086ca757f1, we would encourage all community members and token holders to make their own assessment of the contracts once they are deployed and verified.

Scope

The following contract was a subject for static, dynamic and functional analyses:

Contracts

- BeamCrowdsale.sol
- BeamToken.sol

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?

Analysis

- Test coverage
- Dynamic tests
- Gas usage
- Functional tests

Issues

Severity Description

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.

Minor

- Typo in event name MenegerUpdated - `Correctness` #L16 and #L70 View on GitHub
  - [x] Fixed: 468b86
- Prefer explicit declaration of variable types - `Best practice` Prefer to use explicit variables types. It is recommended to explicitly define your variable types and keep consistency. This confirms your intent and safeguards against a future when the default type changes. #L411 Prefer `uint256` instead of `uint` . View on GitHub
  - [x] Fixed: 468b86
- Function docstring not accurate to function - `Best practice` The `_postValidatePurchase` docstring mentions that the function should use `revert` statements to rollback when valid conditions are not met. The `_checkSeed` and `_checkSoftCap` functions do not use any reverts. #L873-L874 View on GitHub
  - [x] Fixed: 468b86
- Avoid magic numbers - `Best practice` In `BeamCrowdsale.sol` there are some hard coded values, this code could be more readable/maintainable if the values were saved to a variable instead. The two `oraclize_query` functions contain this line: `if (price > 1 ether + tx.gasprice*200000) return 0; // unexpectedly high price` View on GitHub
  - [x] Fixed: 468b86
    - Recommend externalising shared contracts e.g Ownable, SafeMath, Whitelist `Best Practice` There are shared contracts which are imported by CustomContract.sol, BeamCrowdsale.sol, and BeamToken.sol which are duplicated in each file. These shared contracts could be put into their own file

so they only need to be modified once. This would make it less likely to introduce mistakes if the contracts need changed. View on GitHub

Moderate

- Race condition found when user claiming their ETH from the contract - `Best practice` , Security` #L643-L644 This is a typical race condition. It can cause you lose all the ETHs deposited in the contract. Fixing is required! More infor about Race Condition View on GitHub
  - [x] Fixed: 468b86

Major

- None found

Critical

- None found

Observations

- The function `setPublicRound` allows you to start/finish a public or private round. There is no check that the public or private round has already been finished, so it would be possible to start either round multiple times.
- The usage for the `buyForFiat` function is not well documented, this function is only for owners so this is not a huge issue. How to calculate `_usdUnits` is not clear unless you hunt around the contract for other usages.
- The crowdsale contract has a `withdrawFunds` function that allows the contract owner to withdraw all ETH from the contract at any time. https://github.com/BlockchainLabsNZ/beam-contracts-audit/blob/master/contracts/BeamCrowdsale.sol#L605-L611

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 took part in carefully reviewing all source code provided.

Overall we consider the resulting contracts following the audit feedback period adequate and any potential vulnerabilities have now been fully resolved. These contracts have a low level risk of ETH and BEAM being hacked or stolen from the inspected contracts

—

Disclaimer

Our team uses our current understanding of the best practises for Solidity and Smart Contracts. Development in Solidity and for Blockchain is an emerging area of software engineering which still has a lot of room to grow, hence our current understanding of best practise may not find all of the issues in this code and design.

We have not analysed any of the assembly code generated by the Solidity compiler. We have not verified the deployment process and configurations of the contracts. We have only analysed the code outlined in the scope. We have not verified any of the claims made by any of the organisations behind this code.

Security audits do not warrant bug-free code. We encourge all users interacting with smart contract code to continue to analyse and inform themselves of any risks before interacting with any smart contracts.