Back To Earth Token Audit

Preamble

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

Solidity contracts were sourced directly from the developers - 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

- Latest version of solidity compiler is recommended (Addressed in #5)
- `burn` `transfer` and `burnFrom` functions accept a value of 0 tokens, which is not correct as this will spend gas on a transaction that does not alter the state (First addressed in #3 and again in #5)
- ERC20 spec correctness
- `transfer` should return boolean true/false (Addressed in #4)
- `Approval` event, which is triggered when `approve` is called: `event Approval(address indexed _owner, address indexed _spender, uint256 _value)` (Addressed in #4)
- `mintToken` emits the `Transfer` event twice, this can be simplified to `Transfer(0, target, mintedAmount);` (Addressed in #5)
- `throw`ing will consume all gas, consider returning in some cases to preserve the users gas (Addressed in #5)

Moderate

- `burn` function doesn’t check whether the address is frozen, therefore it is possible to ‘melt’ the frozen tokens and thereby decrease the total supply (Addressed in #4)
- `mintToken` isn't checking for overflows (Addressed in #5)

Major

- `burnFrom` never decreases the allowance amount, this would allow someone with an allowance to burn all tokens belonging to that token holder. (Addressed in #3)
- `burnFrom` should only be callable by the contract owner, otherwise anyone with an `allowance` from a token holder could `burnFrom` instead of `transferFrom`

thereby decreasing the `totalSupply` (Addressed in #3)

Critical

None found