SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Client: BabyLonia Team
Date: 21/04/2022
Audit By: Vaibhav Gupta

Disclaimer:

This document may contain confidential information about the proprietary code along with intellectual property of the client as well as information about potential vulnerabilities which can be used for exploitation.

The Auditor does not guarantee the authenticity of the client's project or the organization or any individual associated with the project.

All representations, warranties, undertakings related to the client's project are excluded, particularly concerning - but not limited to - the qualities of the assessed project and products.

The auditor may not be held responsible for the use that may be made of the information contained herein.

The document containing confidential information can be used internally by Client, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Client.

Document Information:

Name	Smart contract code review and security analysis report for BabyloniaV2Token
Audit by	Vaibhav Gupta
Type	ERC20/BEP20 Token
Platform	Solidity
File	Token.sol
MD5 Hash	dfa2d2575b84f144ebd77cbfaf31bf0e
Website	https://www.babylonia.app/
Date	21st April 2022

Introduction

The auditor (Vaibhav Gupta) was contracted to audit the contracts for the Babylonia's V2 Token and the Presale contracts.

This report provides an overview of the checks performed and the discoveries made throughout the process of auditing the BabyV2Token contract.

Complete manual deployment and evaluation of the business logic and security was done along with automated analysis by tools like

Coverage

The following smart contracts were covered in this audit:

Filename: Token.sol
MD5 Hash: dfa2d2575b84f144ebd77cbfaf31bf0e
Contract Name: BABYV2Token

The following checks were performed on the specified contract:

Code based checks:

- Overflows and Underflows
- Reentrancy
- Ownership takeover
- Denial of service attacks
- ERC20 API conformance
- Costly loops
- Transaction ordering
- Unsafe Self Destructs
- Unchecked external calls
- Transaction ordering
- Visibility Levels
- Consistency checks
- Event generation
- Ownership security
- Use of Unsafe Libraries
- Access control

Functional checks:

- Basic Functionality Checks
- Advanced Functionality Checks
- Business logic review
- Token supply manipulation
- User balance manipulation
- Escrow manipulation
- Data consistency
- Data integrity
- Unsafe Minting
- Asset integrity
- Authorization control

Audit Summary:

The audit summary highlights the issues found during analyzing the contracts and completing all the code and functional checks.

Severity	Issues detected	Description
Critical	0	Critical vulnerabilities are the vulnerabilities that can cause direct loss of funds.
High	0	These vulnerabilities can be complex to exploit but still a motivated attacker can exploit these to cause loss of funds.
Medium	1	These issues should be fixed but usually cannot directly cause loss of funds but can affect the data integrity.
Low	3	These issues do not affect any functionality and cannot cause any direct data or fund loss as they do not significantly impact the execution of the contract.

During the audit a total of 0 Critical , 0 High , 1 Medium and 3 Low Severity issues were detected, overall the contract is well secured.

Audit Overview

Critical Severity Issues

No critical severity issues were found

High Severity Issues

No high severity issues were found

Medium Severity Issues

1) **Possibly incorrect comparison in the “mint” function**

   Currently the variable **_maxSupply** is set to as 888888888 BABY , but it is only possible to mint only 888888887 BABY, if the total number of mintable tokens should be 888888888 BABY the **_maxSupply** should be set to 888888889 BABY or the comparison should be changed to greater than equal to

   Contract: BABYV2Token
   Require Condition: require(_maxSupply > _mintamount * 10 ** _decimals + _totalSupply, "Cannot be more than MaxSupply");
   Recommendation: Change the comparison the greater than equal require(_maxSupply >= _mintamount * 10 ** _decimals + _totalSupply, "Cannot be more than MaxSupply");

Low Severity Issues

1) **Variables can be set as constants**

   Some variables whose values are constant their values can be set directly when declaring the constants

   Contract: BABYV2Token
   Variables: _name, _symbol, _decimals
   Recommendation: Add the attribute constant to these variables and the functions decimals() , symbol(), name() as pure

2) **Variables can be set as immutable**

   Variable can be set as immutable whose value is set in constructor and is not

changed after declaring in the constructor

Contract: BABYV2Token
Variables: _maxSupply
Recommendation: Add the attribute immutable to the variable or change to constant by setting the value as a constant

3) **Conformance to Solidity naming conventions**

Solidity defines a standard naming convention that is recommended

Contract: BABYV2Token
Variables: _mintamount
Recommendation: Follow the Solidity Naming convention (https://docs.soliditylang.org/en/v0.8.13/style-guide.html)

Conclusion

The smart contract “BABYV2Token” was tested manually along with automated analysis with multiple tools.

A total of 3 Low Severity and 1 Medium Severity issues were found, this report contains all the found issues and their recommended solutions.

The checks performed during the audit suggest the contract is well secured.