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 Presale Contract
Audit by	Vaibhav Gupta
Type	Crowdsale Contract
Platform	Solidity
File	Presale.sol
MD5 Hash	7f6a573bbaff78ed6edcc93eca9daf48
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 Presale contract.

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

Coverage

The following smart contracts were covered in this audit:

Filename: Presale.sol
MD5 Hash: 7f6a573bbaff78ed6edcc93eca9daf48
Contract Name: Presale

The audit assumes that the payment tokens (BUSD, USDT and USDC) , the BABY Token and the masterWallet are already verified and are trusted..

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	0	These issues should be fixed but usually cannot directly cause loss of funds but can affect the data integrity.
Low	4	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 , 0 Medium and 4 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

No medium severity issues were found

Low Severity Issues

1) **Event Should be emitted when changing a critical variable**

   Whenever changing a core variable's value an event should be emitted to notify the change so that the change can be tracked off chain.

   Contract: Presale
   Functions: updatePresaleRate, updatebabyToken, updateMasterWalletAddress
   Recommendation: Emit an event in the specified function

2) **Conformance to Solidity naming conventions**

   Solidity defines a standard naming convention that is recommended

   Contract: Presale
   Functions,Events and Variables: PresalepresaleEnabledUpdated, _enabled, DepositBusd, _amount, _type, _baby, _newWalletAddr
   Recommendation: Emit an event in the specified function

3) **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: Presale
Variables: busd, usdt, usdc, startTime, endTime
Recommendation: Add the attribute immutable to the variable

4) **Improve calculation while adding amounts into the mapping deposits**

Value is added into mapping **deposits** through the function **DepositBusd**, the statement can be updated to present correct values if the contracts busd, usdt, and usdc have different numbers of decimals.

Currently the value can show incorrect values if payment tokens have different
number of decimals in their notation thus causing issues when using that data externally.

Contract: Presale
Function: DepositBusd
Statement: deposits[msg.sender] = deposits[msg.sender] + _amount;
Recommendation: Change the statement to
deposits[msg.sender] = deposits[msg.sender] + _amount.div(10 ** _tokenDecimals);

Conclusion

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

A total of 4 Low 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.