TrustedBytes
BLOCKCHAIN SECURITY

Babylon finance

BSC

Public audit report

trustedbytes.org

contact@trustedbytes.org

2023

Contents

Project description

Project name	**Babylon finance**
URL	**n / a**
Blockchain	**BSC**
Logo	**n / a**
Ticker	**$BABY**
Contracts checked	**Babylon (not deployed)**
Date	**2023-01-10**

Introduction

The report was prepared for Babylon finance team. The audit was requested by a development team.

Babylon token is a token with commissions on transfers. Part of the commissions is swapped to BUSD and distributed amongst holders. The other commissions are used for marketing address, adding liquidity and buyback.

The code is located in GitHub repo and was audited after commit b58b942 .

Users must make sure that they are interacting with the same contract as was audited.

Procedure

We perform our audit according to the following procedure:

- Automated analysis;
- Scanning the project's smart contracts with automated Solidity analysis tools of our own development;
- Manual verification all the issues found by the tools.

Manual audit

- Manually analyse smart contracts for well known or trended security vulnerabilities
- Smart contracts' logic check

Automatic check results

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	not passed
DoS With Block Gas Limit	not passed
Presence of unused variables	not passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain	passed
Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed

Issues

High risk	1 issue found
Medium risk	1 issue found
Low risk	8 issues found

Security Score: C

Issues desription:

High risk issues

1. Transfers may run out of block gas (Babylon)

Transfers may run out of block gas limit - iteration over array of unlimited length in L626. A test for the issue can be seen in appendix.

Recommendation: The contract has no documentation and therefore it's hard to provide exact recommendations on how this issue should be fixed. The impact of the issue can be mitigated if buy records are deleted when they are consumed. This will not make it impossible to run out of the block gas limit but will make the possibility of this significantly lower. Another option is to rethink how the contract should work and change the logic to be easier to implement without loops.

Medium risk issues

1. No tests (Babylon)

There are no tests in the repository.

Recommendation: It's crutially important to have 100% unit test coverage. We strongly recommend adding tests to ensure that contract works as expected.

Low risk issues

1. No need to use safemath (Babylon)

The compiler version is ^0.8.0. There is no need to use safemath in this version.

Recommendation: Remove SafeMath library

2. Default variable visibilities (Babylon)

Default variable visibilities, for example, L243, L245, L406, L 410, L420, L428, L432, L445, L454, L461, L470, L474.

Recommendation: Set visibility explicitly.

3. Testnet addresses are hardcoded in the code (Babylon)

Testnet addresses are hardcoded in the code, for example, addresses BUSD and WBNB.

Recommendation: Pass addresses in constructor to avoid hardcoded values and make the contract easier to test.

4. MASK variable is not used (Babylon)

The MASK variable declared in L399 is not used.

Recommendation: Remove unused variable.

5. DEAD and DEAD_NON_CHECKSUM are the same (Babylon)

Address DEAD and DEAD_NON_CHECKSUM addresses are hardcoded in the code and have one value "0×000000000000000000000000000000000000dEaD".

Recommendation: Remove one of the variables.

6. emit events if try-catch fails (Babylon).

In function _transferFrom() and swapBack() emit events if try-catch fails. This will help to track errors.

Recommendation: Add events in the catch block.

7. Floating pragma (Babylon)

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly.

Recommendation: We recommend using a fixed Solidity version to ensure that the contract is deployed with the same Solidity version as it was tested to avoid discrepancies in the behavior of different Solidity versions.

8. Missing input parameter checks in approve() function (Babylon)

Missing input parameter checks in approve() function.

Recommendation: We recommend adding requirements: require(owner != address(0), "ERC20: approve from the zero address"); require(spender != address(0), "ERC20: approve to the zero address");

Disclaimer

This audit report is based on the existing knowledge and skills of our auditors, as well as specialized tools of our own development. Verification is always carried out for already known main types of vulnerabilities and threats, both from the user (investor) and from the developers. An audit of a smart contract doesn't provide a full guarantee that a particular project is completely safe. An audit is just one of the tools to check the honesty of the founders (developers) and their professional skills in developing, along with such methods as liquidity pool locking , KYC and others. Only in the aggregate, you can decide whether to invest in a project or not to do it. First of all DYOR and nothing else.

TrustedBytes

BLOCKCHAIN SECURITY

Trustedbytes.org

Contact@trustedbytes.org

/Trustedbytesorg

@trustedbytes

@trustedbytesorg