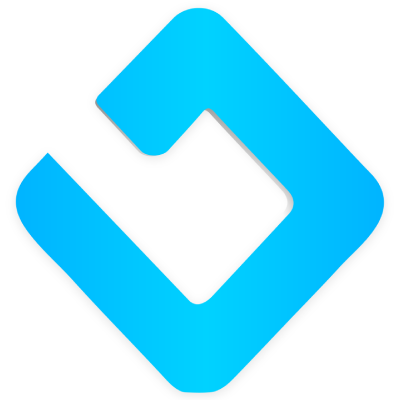


­­­­­

A Secure Place For Web3

**SMART CONTRACT AUDIT OF**

**Bloggercube Fair Launch**



Contract Address:

**0x43B121Af0fE2085D72c544e4B6f163C5A8a15D9F**



www.expelee.com | Page 1 |



**­Audit Summary**

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smartcontract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED (WITH MEDIUM SEVERITY)**

KYC Verification: **YES**

Audit Date: 20/05/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren’t resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.



www.expelee.com | Page 2 |



**DISCLAMER**

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.



www.expelee.com | Page 3 |



**Contract Review**

|  |  |
| --- | --- |
| **Contract Name** | **BloggercubeToken** |
| **Compiler Version** | **v0.8.10+commit.fc410830** |
| **Optimization** | **Yes with 200 runs** |
| **License** | **MIT license** |
| **Explorer** | **https://bscscan.com/address/0x43B121Af0fE2085D72c544e4B6f163C5A8a15D9F#code** |
| **Symbol** | **BCC** |
| **Decimals** | **9** |
| **Total Supply** | **10,000,000,000** |
| **Domain** | **https://dapp.bloggercube.com/** |



www.expelee.com | Page 4 |



**Project Review**

Token Name: Bloggercube

Web Site: https://dapp.bloggercube.com/

Twitter: https://twitter.com/\_bloggercube

Telegram: https://t.me/+Y5o18KI0E1FiNjky

Contract Address: 0x43B121Af0fE2085D72c544e4B6f163C5A8a15D9F

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY



www.expelee.com | Page 5 |



Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

- Unhandled Exceptions - Transaction Order Dependency Smart Contract - Integer Overflow Vulnerabilities - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation

Source Code - Gas Limit and Loops Review - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation

Functional - Operations Trail & Event Generation Assessment - Assets Manipulation - Liquidity Access



www.expelee.com | Page 6 |



Vulnerability Checklist

|  |  |  |
| --- | --- | --- |
| № | Description. | Result |
| 1 | Compiler warnings. | Passed |
| 2 | Race conditions and Re-entrancy. Cross-function race conditions. | Passed |
| 3 | Possible delays in data delivery. | Passed |
| 4 | Oracle calls. | Passed |
| 5 | Front running. | Passed |
| 6 | Timestamp dependence. | Passed |
| 7 | Integer Overflow and Underflow. | Passed |
| 8 | DoS with Revert. | Passed |
| 9 | DoS with block gas limit. | Passed |
| 10 | Methods execution permissions. | Passed |
| 11 | Economy model. | Passed |
| 12 | The impact of the exchange rate on the logic. | Passed |
| 13 | Private user data leaks. | Passed |
| 14 | Malicious Event log. | Passed |
| 15 | Scoping and Declarations. | Passed |
| 16 | Uninitialized storage pointers. | Passed |
| 17 | Arithmetic accuracy. | Passed |
| 18 | Design Logic. | Passed |
| 19 | Cross-function race conditions. | Passed |
| 20 | Safe Zeppelin module. | Passed |
| 21 | Fallback function security. | Passed |



www.expelee.com | Page 7 |



Manual Audit

* **Low-Risk**
* 4 low-risk code issues found
* **Medium-Risk**
* 0 medium-risk code issues found
* **High-Risk**
* 0 high-risk code issues found



www.expelee.com | Page 8 |



* **Low-Risk**

1. **Too many digits**

Literals with many digits are difficult to read and review

\_mint(msg.sender, 10000000000 \* 10\*\*decimals());

1. **No zero address validation for some functions**

Detect missing zero address validation.

function setMarketingWalletAddress(address \_addr) public

onlyOwner

returns (bool)

{

marketingWalletAddress = \_addr; return true;

}

**Recommendation**

Check that the new address is not zero.



www.expelee.com | Page 9 |



1. **Divide before Multiply**

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

if (marketingWalletAddress != address(0) &amp;&amp; marketingFeePercent &gt; 0) { marketingFee = (amount / 100) \* marketingFeePercent;

}

if (developmentWalletAddress != address(0) &amp;&amp; developmentFeePercent &gt; 0) { developmentFee = (amount / 100) \* developmentFeePercent;

}

if (donationWalletAddress != address(0) &amp;&amp; donationFeePercent &gt; 0) { donationFee = (amount / 100) \* donationFeePercent;

}

**Recommendation**

Consider ordering multiplication before division



www.expelee.com | Page 10 |



1. **Missing events arithmetic**

Detect missing events for critical arithmetic parameters

function setMarketingFee(uint8 \_fee) public

onlyOwner

onlyValidFee(\_fee) returns (bool)

{

marketingFeePercent = \_fee; return true;

}

**Recommendation**

Emmit an event for critical parameter change



www.expelee.com | Page 11 |



Manual Audit (Contract Function)

contract BloggercubeToken is ERC20, ERC20Snapshot, Ownable {

// State Variables

uint8 public marketingFeePercent = 0; uint8 public developmentFeePercent = 0; uint8 public donationFeePercent = 0;

address public marketingWalletAddress;

address public developmentWalletAddress; address public donationWalletAddress;

address public \_owner;



www.expelee.com | Page 12 |



Important Points To Consider

✔ The owner cannot mint tokens after Initial

✔ The owner cannot stop Trading.

✔ Verified contract source

✔ Token is sellable (not a honeypot) at this time

✘ Ownership renounced or source does not contain an owner contract

✘ Source does not contain a fee modifier (Owner can change buy/sell fees up to 100%.)

✘ Owner/creator wallet contains less than 10% of circulating token supply (12.6%)



www.expelee.com | Page 13 |

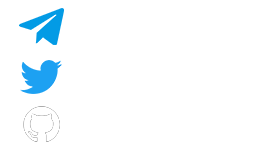
About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through ourservices so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

* Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
* Manually detailed examination of the code line by line by experts.
* Live test by multiple clients using Test net.
* Analysing failure preparations to check how the Smart
* Contract performs in case of any bugs and vulnerabilities.
* Checking whether all the libraries used in the code are on the latest version.
* Analysing the security of the on-chain data.

Social Media





www.expelee.com | Page 14|