FOR

KOALAS CLUB

WEBSITE

https://www.Koalas.world

TELEGRAM

https://t.me/Koalaworld

SMART CONTRACT AUDIT

Twitter:
twitter.com/vital_blockA

Telegram:
t.me/vital_Block

Disclaimer

Vital Block Solidity reports are not, nor should be considered, an "endorsement" or "disapproval" of any project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Vital Block to perform a security review.

Vital Black Solidity Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analysed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

Vital Block Solidity Reports should not be used in any way to make decisions around investment or involvement with any project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort. Vital Block Solidity Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Vital Block Solidity's position is that each company and individual are responsible for their own due diligence and continuous security. Vital Block Solidity's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyse

What is a Vital Block Audit report?

•A document describing in detail an in-depth analysis of a particular piece(s) of source code provided to Vital Block Solidity by a Client.

•An organized collection of testing results, analysis and inferences made about the structure, implementation, and overall best practices of a particular piece of source code.

•Representation that a Client of Vital Block Solidity has indeed completed a round of auditing with the intention to increase the quality of the company/ product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	**KOALAS WORLD**
Description	**KLUB is community-driven innovative meme token on BSC (Binance Smart Chain) Our token is fully governed and owned by the community. Every decision of KLUB is decided by the community.**
Platform	**Binance Mainnet**
Mainnet Contracts:	**0xe52507731bb1c9953B7772574Bf203C465FC6814** ***Koalas Club (KLUB)***
Files:	**Koalas Club.sol**

Audit Summary

Delivery Date	**July 6 2022**
Method of Audit	**Security Static Analysis**
Timeline	**Story Points 100**

Vulnerability Summary

Total Issues Found	**0**
Total Issues Resolved	**0**
Total Critical	**0**
Total High	**1**
Total Medium	**2**
Total Low	**0**
Total Informational	**2**

Our Audit Methodology

- **STEP 1**
  A manual line-by-line code review to ensure the logic behind each function is safe and secured against common attack vectors.

- **STEP 2**
  Simulation of hundreds of thousands of Smart Contract Interactions on a test and Mainnet blockchain using a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

- **STEP 3**
  Consultation with the project team on the audit report pre-publication to implement recommendations and resolve any outstanding issues.

Grading

The following grading structure is used to assess the level of vulnerability found within all Smart Contracts:

THREAT LEVEL	DEFINITION
Critical	Severe vulnerabilities which compromise the entire protocol and could result in immediate data manipulation or asset loss.
High	Significant vulnerabilities which compromise the functioning of the smart contracts leading to possible data manipulation or asset loss.
Medium	Vulnerabilities which if not fixed within in a set timescale could compromise the functioning of the smart contracts leading to possible data manipulation or asset loss.
Low	Low level vulnerabilities which may or may not have an impact on the optimal performance of the Smart contract.
Informational	Issues related to coding best practice which do not have any impact on the functionality of the Smart Contracts

Description

KOALAS WORLD: (KCLUB) tokenomics are designed with long-term value creation in mind. We believe that great things take time to build properly and that success will require long-term commitment from the team. This is reflected in the tokenomics here below

Buy Trading Fees 10.0% - LP | 3% Reward | 4% Marketing | 1% Buy Back & Burn | 2% LP

Sell Trading Fees 10.0% - LP | 3% Reward | 4% Marketing | 1% Buy Back & Burn | 2% LP

Initial supply : **5,000,000,000 KLUB**

KOALAS WORLD 🌎 TOKENOMICS

BUY TAX

10% Forever

3% Rewards

4% Marketing

2% LP

1% Buyback and burn

SELL TAX

10% Forever

3% Rewards

4% Marketing

2% LP

1% Buyback and burn

LIQUIDITY

Locked

STATS

5 Billion supply

Vulnerability 0: No important security issue detected.
Threat level: Medium

Description:
Not a honeypot transaction simulation is success at the moment. Always DYOR before investing.

INFO! There is no liquidity with BNB. Results with non-BNB pair may differ. If the token is not live yet, results may be different once the token is live. It is common for tokens to have 0% taxes before launching on DEX!

```
ousOwner, address indexed newOwner);


 the deployer as the initial owner.



sgSender();
nder;
wnershipTransferred(address(0), msgSender);


  /**
   * @dev Returns the address of the current owner.
   */
  function owner() public view virtual returns (address) {
    return _owner;
  }

  /**
   * @dev Throws if called by any account other than the
   */
  modifier onlyOwner() {
    require(owner() == _msgSender(), "Owr
    _;
  }

  /**
   * @dev Leaves tl
   * `onlyOwne
   *
```

Vulnerability 1: The owner can change the high fee setting function in the contract.

Threat level: Law

Vulnerability 1: Gas optimisation

Threat level 2: Informational

Description: this smart-contract can be Modified by Deployer
This can always change! Do your own due diligence.
INFO! Owner can change trading tax fee. which is Reall a normal function for most Smart Contract. Removal fee is private and calculate function

KOALAS WORLD (KCLUB)

No trading data available: either trading is disabled, or no Liquidity for the token Yet.

Recommendation:

The contract can be modified so that it can be done via a single call to save gas.

```
* If `_data` is nonempty, it's used as data in a delegate call to `_logic`. This will typic
* function call, and allows initializating the storage of the proxy like a Solidity constru
*/
onstructor(address _logic, bytes memory _data) payable {
   assert(_IMPLEMENTATION_SLOT == bytes32(uint256(keccak256("eip1967.proxy.implementation")
   _upgradeToAndCall(_logic, _data, false);


**
* @dev Returns the current implementation address.
*/
unction _implementation() internal view virtual override returns (address impl) {
   return ERC1967Upgrade._getImplementation();

DX-License-Identifier: MIT

a solidity ^0.8.0;
```

Issues Checking Status

	Issue description	Checking status
1.	**Compiler errors.**	Passed
2.	**Race conditions and Reentrancy. 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 of the contract.**	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 Open Zeppelin contracts implementation and usage.**	Passed
21.	**Fallback function security.**	Passed

Conclusion

During the Vital block Audit process, the KOALAS contract was analysed by manual review and automated testing. All issues identified was after deployment to mainnet. By submitting the contract for audit after Deployment, the team have displayed a strong commitment to security.

Whilst there are no obvious vulnerabilities or security risks identified within the main net contract, it is beyond the scope of this Vital Block Audit to comment upon any risks associated with tokenomics, adoption or platform longevity. Before placing funds in any defi protocol Vital Block encourages potential investors to exercise due diligence and research all projects thoroughly to assess plans for ongoing development and financial sustainability.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectely on certain edge cases that may result in avulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a structassignment operation affecting an in-memory struct rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Appendix

Inconsistency
Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers
Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error
Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code
Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

VitalBlock

Making Defi And Web3 a Safer place

WWW.VITALBLOCK.XYZ

Decentralized Smart Contract Auditing Firm.