QuillAudits

Audit Report
June, 2024

For

K-MBIO

Table of Content

Executive Summary

Project Name	Kmbio
Overview	Kmbio is a project centered around the KMBIO Token, an ERC20 token with burnable capabilities, upgradeability via UUPS, and ownership management. It leverages OpenZeppelin's secure, upgradeable contracts to ensure robustness and flexibility. The token is initialized with a supply minted to an admin address and features an owner-restricted mint function, allowing for controlled inflation. The upgradeability aspect ensures that Kmbio can adapt over time without sacrificing security or decentralization.
Timeline	5th August 2024 - 6th August 2024
Updated Code Received	NA
Second Review	NA
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Audit Scope	The scope of this audit was to analyse the Kmbio Token Contract for quality, security, and correctness.
Source Code	https://etherscan.io/address/0x9ffdb2e1f0a55740694a143152ba3ee5fde359c0#code
Branch	NA
Fixed In	NA

Number of Security Issues per Severity

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	2
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC's standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Hardhat, Foundry.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Informational Issues

1. Ownership Transfer should be a Two-way Process

Path

KMBIOToken

Description

Due to the importance of the contract owner being responsible for mint, and burn; it is important to stress the need for the use of a two-way process on the transfer of ownership. When the current owner invokes the transferOwnership function, passing the parameter of the new address, this sets the new address immediately, supposing it is not an address zero; hence, it would revert. However, the issue arises when the address passed was that of a wrong address, and this would not be redeemable anymore.

Recommendation

Use the Openzeppeling Ownable2StepUpgradable to remedy the issue of instantaneous transfer to the wrong address. This way, the assigned address would claim ownership first before the completion of the ownership transfer.

Status

Acknowledged

2. Avoid using the floating pragma

Path

kMBio.sol

Description

Floating pragma should only be used when a contract is intended for consumption by other developers.

Locking the pragma helps ensure that contracts are not accidentally deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

Recommendation

Use locked pragma instead.

Status

Acknowledged

Functional Tests Cases

Some of the tests performed are mentioned below:

- Should get the name of the token
- Should get the symbol of the token
- Should get the total supply of the token when deployed
- Should approve another account to spend token
- Should transfer tokens to other address
- Should not be able to upgrade contract after ownership is renounced.
- Should renounce ownership of the proxy contract.
- Should prevent re-initialization
- Should correctly initialize upgradeable proxy with the implementation logic.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the KMbio codebase. We performed our audit according to the procedure described above.

Kmbio Token Contract Looks good, only informational severity Issues were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in KMbio smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of KMbio smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the KMbio to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2024

For

K-MBIO

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com