BAIL security

BAILSEC.IO

EMAIL : OFFICE@BAILSEC.IO

TWITTER : @BAILSECURITY

TELEGRAM : @HELLOATBAILSEC

FINAL REPORT:

Kuma Protocol

June 2023

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

Executive Overview

1. Project Details

Project	Cairo Finance
Website	https://cairofinance.app/
Type	Decentralized Financial Services (DEFI)
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/mimo-capital/kuma-staking/tree/fd613ffd4c0ae49cf9b65e9e6844dbad5aa32fdf

2. Detections Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1		
Medium	0			
Low	1			1
Informational	2	2		
Total	0			

2.1 Detections Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low level risk to the project or users. Nevertheless the issue should be fixed immediately
Informational	Effects are small and do not post an immediate danger to the project or users

3. Detections

KumaUniswapStaking.sol

The KumaUnswapStaking smart contract is a staking contract which allows users to stake and withdraw wKIB tokens. Users have the possibility to stake tokens via a simple deposit or zap a specific, pre-determined stablecoin into the contract using a UniswapV3Pool. Similar to the staking interaction, users can withdraw KIB tokens directly or withdraw and swapping them atomically using UniswapV3 to the pre-determined stablecoin. Moreover, users can internally assign their balance to other addresses, which effectively reduces a user's share and increases the corresponding address' share. It is important to note that the wKIB token is the wrapped version of a yield-bearing token which increases its supply during each transaction, respectively increasing each unique token holder's balance, the wrapped version is just a standard ERC20 token which can be used for this staking purpose. This contract is solely meant to take custody of users' tokens and does not include any incentives for staking.

Any privileged address can change the UniswapV3Router, UniswapV3Fee and UniswapV3Factory without any limitations.

The access control mechanism is handled externally by the AccessController contract.

Issue	Malicious Manager can steal all tokens
Severity	High
Description	Any address which has the *KUMA_MANAGER_ROLE* granted is privileged to change the *UniswapRouter*, *UniswapFactory* and UniswapFee. At first sight these privileges do not seem to have any massive impact since one could think the only bad thing that could happen is a DoS of the zapping functionalities. However, that is not true, a malicious manager could set the router to a malicious contract which a) returns

	a huge *_wKIBTBought* amount and b) transfers the token to another address than the staking contract. Using this methodology, a malicious manager can artificially increase their own shares, effectively draining all users' tokens within the contract. Moreover, the freedom of these variable changes can result in a DoS state for the zapping functionality and can also be abused by the router address to consume any unconsumed approvals.
Recommendations	The most secure mitigation would be to make the router and fee immutable, however, since we assume flexibility is desired, we simply recommend to only add kyc-ed multisig contracts as managers.
Comments	Fixed, the Kuma team stated that a multisig will be used for this purpose. This fix solely relies on the trustworthiness of the Kuma team.

Issue	Possibly low liquidity on UniswapV3
Severity	Low
Description	Since the wKIB token is just a wrapped version of the KIB token, we assume that this token might only be used for staking purposes. If this is true, we assume that the liquidity for this token is relatively low on UniswapV3, which will result in a suboptimal swap experience.
Recommendations	We highly recommend to think about this issue and if this is in fact an issue, we recommend offering incentives for users to create such a pair.
Comments	Acknowledged, the team is exploring methodologies to incentivize such a pair.

Issue	*AccessControl* address can be EOA
Severity	Low
Description	The *AccessControl* address is set during the contract deployment. While there is a check against *address(0)*, there is no check that the *AccessControl* address is in fact a contract. If this address is accidentally set to an EOA, all privileged functions will be callable by anyone.
Recommendations	We recommend executing an *isContract* check during the setting of the *AccessControl* address.
Comments	Acknowledged.

Issue	*Factory* is router dependent
Severity	Informational
Description	The *UniswapFactory* is always dependent on the router, therefore whenever the *UniswapRouter* is changed, the factory should automatically be changed as well. *abstract contract PeripheryImmutableState is IPeripheryImmutableState {* */// @inheritdoc IPeripheryImmutableState* *address public immutable override factory;* */// @inheritdoc IPeripheryImmutableState* *address public immutable override WETH9;* *constructor(address _factory, address _WETH9) {* *factory = _factory;*

	WETH9 = _WETH9; *}*
Recommendations	We recommend to simply fetching the factory directly from the router via *UniswapRouter.factory()*
Comments	Resolved, the factory will now always be fetched from the current router.

Issue	Various unused code sections
Severity	**Informational**
Description	The codebase contains various unused code sections which do not serve any purpose and thus can be removed to make the code cleaner and more readable: Line 4: import {WadRayMath} from "./libraries/WadRayMath.sol"; Line 19: using WadRayMath for uint256; Line 288 The getShares function contains the same logic as getWithdrawableWrappedKIBT and is therefore redundant and can be removed
Recommendations	We recommend removing the mentioned code sections.

Comments	Resolved, the team stated that the getWithdrawableWrappedKIBT function will be kept to achieve a standard interface.