SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2025.04.21, the SlowMist security team received the KiloEx team's security audit application for KiloEx Phase1, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

This is an audit of the relevant permissions in the KiloEx protocol, mainly including the sorting of privileged roles and auditing of the permission architecture to avoid risks caused by lack of permission control from occurring again.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Single point of control permissions	Others	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N2	Price manipulation risk	Design Logic Audit	Critical	Fixed

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/KiloExContract/kilo-contracts

commit: 3b89fb5009b771367a4643050fa0f7b02e7563fb

Fixed Version:

https://github.com/KiloExContract/kilo-contracts

commit: f4fd938aa6dd4b50972888d9bb328aacb475c051

Audit Scope:

This audit mainly focuses on the permission-related issues in the contracts within the scope of this audit.

The main network addresses of the contracts can be found in the following directory:

/scripts/contracts_deploy/settings/

4.2 Privileged Role Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

Contract	Roles
OperatorOwnerGovernable	Gov
	Owner
	Operators
OperatorOwnerGovernableUpgradeable	Gov

Contract	Roles
	Owner
	Operators
OwnerGovernable	Gov
	Owner
OwnerGovernableUpgradeable	Gov
	Owner
Delegate	NULL
DelegateCollection	trustedForwarder
	keepers
KiloPriceFeed	Owner
	Gov
	keeper
KiloStorageManager	Owner
	Gov
	PerpTrade
	pendingReward
MarginFeeManager	Owner
	Gov
	PerpTrade
	Operator
MarketOrderWithTriggerOrder	Owner
	Gov

Contract	Roles
OrderBook	Owner
	Gov
	Keeper
	approvedRouter
PendingReward	Owner
	Gov
	PerpTrade
	protocolRewardReceiver
	vaultRewardReceiver
	liquidationRewardReceiver
PerpTrade	Owner
	Gov
	approvedRouter
	liquidators
PositionRouter	Owner
	Gov
	PositionKeeper
	approvedRouters
	delegateCollectionAddr
PriceImpactLogic	Owner
	Gov
	Operator

Contract	Roles
ProductManager	Owner
	Gov
	Operator
TrustedForwarder	Owner
	Gov
	approvedRouters
	Keeper
VaultStakeReward	Owner
	Gov
	HybridVault
	openTradesPnlFeed
HybridVault	Owner
	Gov
	Operator
	pnlHandler
PriceRouter	Owner
	Gov
	Operator
VUSD	Owner
	Gov
	HybridVault
CommonReward	Gov

Contract	Roles
	Owner
	Operator
KolRewardDistributor	Gov
	Owner
	Operator
TeamContestReward	Gov
	Owner
	Operator
ReferralStorageManager	Gov
	Owner
	Operator
	Handler
ListaDaoWbnbStrategy	Gov
	Owner
	sideVault
SideVaultWithPending	sideVaultEntry
	Gov
	Owner
AaveV3Strategy	Gov
	Owner
	sideVault
SideVault	Gov

Contract	Roles
	Owner
	sideVaultEntry
SideVaultEntry	Gov
	Owner
	Operator
	HybridVault
VenusVTokenStrategy	Gov
	Owner
	sideVault
AirdropRewardDistributor	Gov
	Owner
	Operator
KiloVestingWallet	Gov
	Owner
XKiloDividends	Gov
	Owner
	Operator
	xKiloToken
XKiloToken	Gov
	Owner
	Operator
AffiliateRewardDistributor	Gov

Contract	Roles
	Owner
	Operator
ProtocolReward	Gov
	Owner
	tradeRewardDistributor
	affiliateRewardDistributorAddr
TradeRewardDistributor	Gov
	Owner
	Operator
KTokenLockedDepositNft	KTokenManager
	KToken
KTokenOpenPnlFeed	Gov
	Owner
	Operator

4.3 Vulnerability Summary

[N1] [Suggestion] Single point of control permissions

Category: Others

Content

Please note that in most contracts of the protocol, some roles that can change the core configurations of the contracts, such as Owner, Gov, Operator, etc., are controlled by EOAs (Externally Owned Accounts), which may lead to single point of failure risks. We will explain the more in-depth risks in the Phase 2 audit.

Solution

If possible, permissions can be transferred to multisig management to mitigate the aforementioned risks.

Status

Acknowledged

[N2] [Critical] Price manipulation risk

Category: Design Logic Audit

Content

In the PriceRouter contract, the priceOfUnderlying function is used to update the price of the underlying token. It selects different price oracle methods based on the price source data parsed from the incoming data parameter. When the price source is from Chainlink, it calls the priceOfChainLink function to update the price. However, in this function, it does not check whether the incoming tokenId corresponds to the token parameter for which the price needs to be obtained. Also, since the priceOfUnderlying function is externally callable by anyone and the data parameter is externally controllable, a malicious user can pass in a token address different from the tokenId to wrongly update the price of the underlying token corresponding to the tokenId, thus affecting the normal operations in the Vault.

Similarly, the same problem exists in the priceOfPyth function.

Code Location: src/hybridvault/PriceRouter.sol

```
    function priceOfUnderlying(bytes calldata data) public override payable {
        OracleSource source = OracleSource(uint8(data[0]));
        if (msg.value > 0) {
            require(source == OracleSource.PYTH, "PriceRouter: updateFee error");
        }
        uint tokenId = uint(uint8(data[1]));
        if (source == OracleSource.KILO_SIGNATURE) {
            priceOfSignature(tokenId, data);
        } else if (source == OracleSource.PYTH) {
            (bytes32 pythId, bytes[] memory priceUpdateData) = abi.decode(data[2:],
(bytes32, bytes[]));
            priceOfPyth(tokenId, pythId, priceUpdateData);
        } else if (source == OracleSource.CHAINLINK) {
            (address token) = abi.decode(data[2:], (address));
            priceOfChainLink(tokenId, token);
        } else if (source == OracleSource.KILO_EX) {
            priceOfKiloEx(tokenId, data);
        } else if (source == OracleSource.MOCK_ORACLE) {
            priceOfMock(tokenId, data);
        }
    }
   function priceOfPyth(uint tokenId, bytes32 pythId, bytes[] memory priceUpdateData)
internal returns (uint) {
        require(oracleSources[tokenId] == OracleSource.PYTH, "PriceRouter: not
allowed");
        require(priceUpdateData.length > 0, "PriceRouter: priceUpdateData is empty");
        uint fee = pyth.getUpdateFee(priceUpdateData);
        pyth.updatePriceFeeds{ value: fee }(priceUpdateData);
        PythStructs.Price memory priceInfo = pyth.getPriceNoOlderThan(pythId,
maxOldAge);
        uint oPrice = uint(uint64(priceInfo.price));
        uint price;
        if (priceInfo.expo >= 0) {
            uint exponent = uint(uint32(priceInfo.expo));
            price = oPrice * PRICE_BASE * (10 ** exponent);
        } else {
            uint exponent = uint(uint32(-priceInfo.expo));
            price = (oPrice * PRICE_BASE) / (10 ** exponent);
        }
        kiloExPrices[tokenId] = PriceInfo(price, priceInfo.publishTime);
        return price;
    }

    function priceOfChainLink(uint tokenId, address token) internal {
        require(oracleSources[tokenId] == OracleSource.CHAINLINK, "PriceRouter: not
allowed");
        (uint price, uint timestamp) = kiloPriceFeed.getChainlinkPrice(token);
        kiloExPrices[tokenId] = PriceInfo(price, block.timestamp);
    }
```

Solution

It is recommended to add a new mapping to check whether the tokenId matches the token for which the price needs to be obtained.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002504230001	SlowMist Security Team	2025.04.21 - 2025.04.23	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical risk and 1 suggestion. All the findings were fixed or acknowledged. The code has been deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

SLOWMIST

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist