# Smart Contract Security Audit Report

# Table Of Contents

# 1 Executive Summary

On 2023.03.07, the SlowMist security team received the PancakeSwap team's security audit application for PancakeSwap v3, 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 |

| Serial Number | Audit Class | Audit Subclass |
|---|---|---|
| 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 |

| Serial Number | Audit Class | Audit Subclass |
|---|---|---|
| | | Function Return Value Security Audit |
| | | External Call Function Security 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 | - |

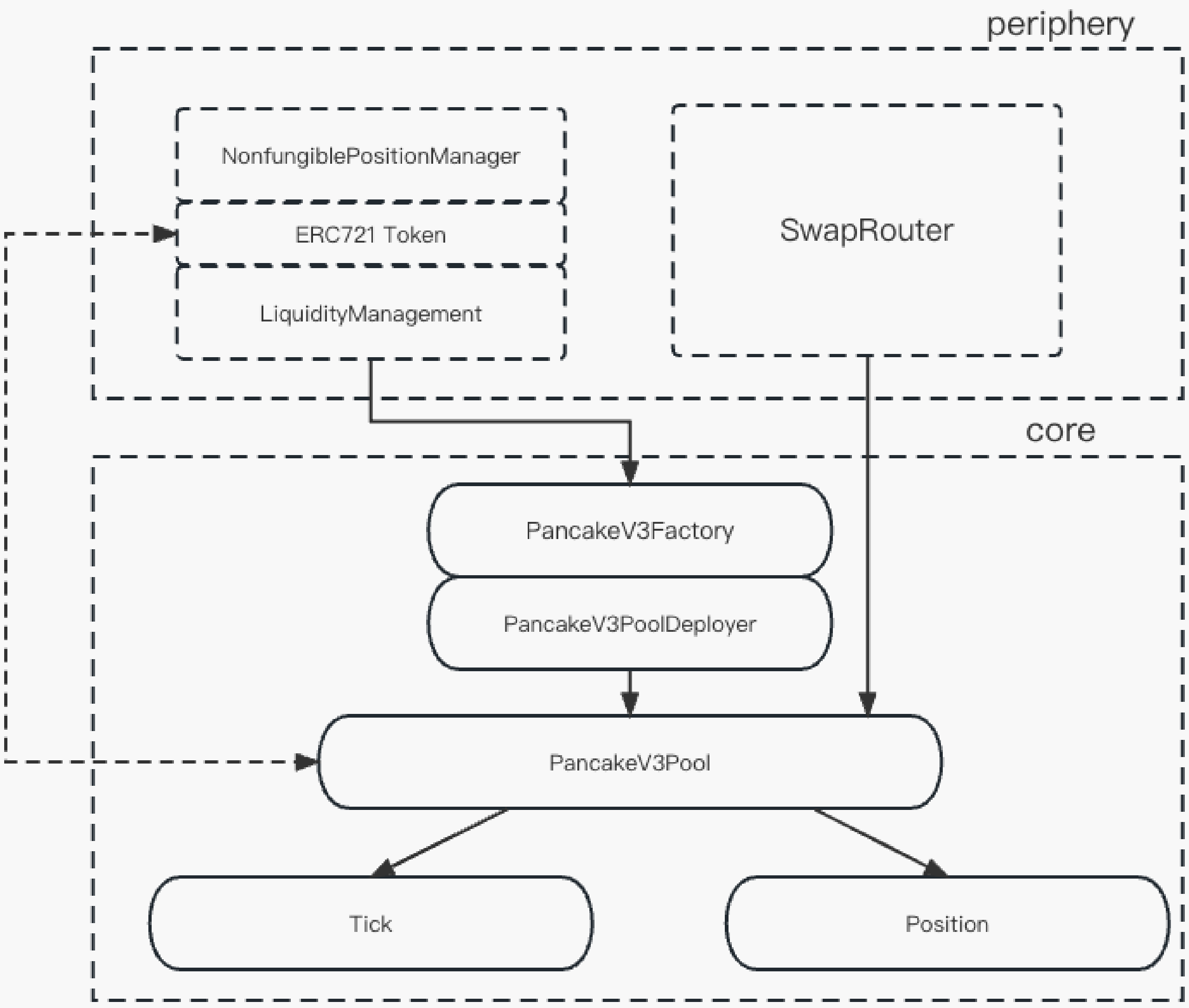
# 3 Project Overview

## 3.1 Project Introduction

PancakeSwap V3 audit section is divided into three parts: core, periphery, and router.

The periphery part is also divided into two parts: transaction pool (Position) management and SwapRoute management. NonfungiblePositionManager is responsible for the creation of transaction pools and the addition and deletion of liquidity. SwapRouter is the management of swap routes. PancakeV3Factory is the transaction pool ( PancakeV3Pool is deployed by PancakeV3PoolDeployer, and PancakeV3Pool is the core logic, which manages Tick and Position, liquidity management and swap function implementation in a pool. Each Position in the Pool is made into an ERC721 Token, i.e., each Position has an independent ERC721 Token ID.

When users make trades in the liquidity pool of PancakeSwap V3, a fixed percentage of the transaction fee is automatically charged, which is determined by the liquidity providers when creating the liquidity pool. Specifically, the transaction fee rate within each price range is set by the liquidity provider. The transaction fee is automatically calculated and charged based on the trading volume and the set fee rate, and the reward portion of the liquidity provider is allocated to the liquidity pool they provided.



## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
|---|---|---|---|---|
| N1 | Risk of excessive authority | Authority Control Vulnerability Audit | Medium | Fixed |
| N2 | Token compatibility issue | Design Logic Audit | Suggestion | Acknowledged |
| N3 | Missing event records | Others | Suggestion | Fixed |
| N4 | Risk of initial operation | Race Conditions Vulnerability | Suggestion | Acknowledged |
| N5 | Risk of denial of service | Gas Optimization Audit | Suggestion | Acknowledged |

# 4 Code Overview

## 4.1 Contracts Description

### Codebase:

#### Audit Version

https://github.com/pancakeswap/pancake-v3 (preview branch)

commit: c593a9287776c3455fd0eefe3520d98da1f4a5dc

- /v3-core
- /v3-periphery (exclude /v3-periphery/lens)
- /router

#### Fixed Version

https://github.com/pancakeswap/pancake-v3

commit: ac65dc9ad8e40e51d5cae0251e395cff70d63b7c

- /v3-core
- /v3-periphery (exclude /v3-periphery/lens)
- /router

The main network address of the contract is as follows:

**The code was not deployed to the mainnet.**

## 4.2 Visibility Description

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

| PancakeV3Factory | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| createPool | External | Can Modify State | - |
| setOwner | External | Can Modify State | - |
| enableFeeAmount | Public | Can Modify State | - |
| setWhiteListAddress | Public | Can Modify State | - |
| setFeeAmountExtraInfo | Public | Can Modify State | - |

| PancakeV3Pool | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| checkTicks | Private | - | - |
| _blockTimestamp | Internal | - | - |
| balance0 | Private | - | - |
| balance1 | Private | - | - |
| snapshotCumulativesInside | External | - | - |
| observe | External | - | - |
| increaseObservationCardinalityNext | External | Can Modify State | lock |

| PancakeV3Pool | | | |
|---|---|---|---|
| initialize | External | Can Modify State | - |
| _modifyPosition | Private | Can Modify State | - |
| _updatePosition | Private | Can Modify State | - |
| mint | External | Can Modify State | lock |
| collect | External | Can Modify State | lock |
| burn | External | Can Modify State | lock |
| swap | External | Can Modify State | - |
| flash | External | Can Modify State | lock |
| setFeeProtocol | External | Can Modify State | lock onlyFactoryOwner |
| collectProtocol | External | Can Modify State | lock onlyFactoryOwner |
| setLmPool | External | Can Modify State | onlyFactoryOwner |

| PancakeV3FactoryOwner | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| setOwner | External | Can Modify State | onlyOwner |
| setLmPoolDeployer | External | Can Modify State | onlyOwner |
| setFactoryOwner | External | Can Modify State | onlyOwner |
| enableFeeAmount | External | Can Modify State | onlyOwner |
| setFeeProtocol | External | Can Modify State | onlyOwner |
| collectProtocol | External | Can Modify State | onlyOwner |
| setLmPool | External | Can Modify State | onlyOwnerAndLmPoolDeployer |

| PancakeV3PoolDeployer | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| setFactoryAddress | External | Can Modify State | - |
| deploy | External | Can Modify State | onlyFactory |

| StableSwapRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| setStableSwap | External | Can Modify State | onlyOwner |
| _swap | Private | Can Modify State | - |
| exactInputStableSwap | External | Payable | nonReentrant |
| exactOutputStableSwap | External | Payable | nonReentrant |

| SmartRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ImmutableState PeripheryImmutableState StableSwapRouter |

| V2SwapRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _swap | Private | Can Modify State | - |
| swapExactTokensForTokens | External | Payable | nonReentrant |
| swapTokensForExactTokens | External | Payable | nonReentrant |

| V3SwapRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |

| V3SwapRouter | | | |
|---|---|---|---|
| pancakeV3SwapCallback | External | Can Modify State | - |
| exactInputInternal | Private | Can Modify State | - |
| exactInputSingle | External | Payable | nonReentrant |
| exactInput | External | Payable | nonReentrant |
| exactOutputInternal | Private | Can Modify State | - |
| exactOutputSingle | External | Payable | nonReentrant |
| exactOutput | External | Payable | nonReentrant |

| NFTDescriptorEx | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| constructTokenURI | Public | - | - |
| escapeQuotes | Internal | - | - |
| generateDescriptionPartOne | Private | - | - |
| generateDescriptionPartTwo | Private | - | - |
| generateName | Private | - | - |
| generateDecimalString | Private | - | - |
| tickToDecimalString | Internal | - | - |
| sigfigsRounded | Private | - | - |
| adjustForDecimalPrecision | Private | - | - |
| abs | Private | - | - |
| fixedPointToDecimalString | Internal | - | - |
| feeToPercentString | Internal | - | - |

| NFTDescriptorEx | | | |
|---|---|---|---|
| addressToString | Internal | - | - |
| generateSVGImage | Internal | - | - |
| overRange | Private | - | - |
| scale | Private | - | - |
| tokenToColorHex | Internal | - | - |
| getCircleCoord | Internal | - | - |
| sliceTokenHex | Internal | - | - |
| setOwner | External | Can Modify State | onlyOwner |
| toggleSwitchAndUpdateNFTDomain | External | Can Modify State | onlyOwner |

| NonfungiblePositionManager | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ERC721Permit PeripheryImmutableState |
| positions | External | - | - |
| cachePoolKey | Private | Can Modify State | - |
| mint | External | Payable | checkDeadline |
| tokenURI | Public | - | - |
| baseURI | Public | - | - |
| increaseLiquidity | External | Payable | checkDeadline |
| decreaseLiquidity | External | Payable | isAuthorizedForToken checkDeadline |
| collect | External | Payable | isAuthorizedForToken |
| burn | External | Payable | isAuthorizedForToken |

| NonfungiblePositionManager | | | |
|---|---|---|---|
| _getAndIncrementNonce | Internal | Can Modify State | - |
| getApproved | Public | - | - |
| _approve | Internal | Can Modify State | - |

| NonfungibleTokenPositionDescriptor | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| nativeCurrencyLabel | Public | - | - |
| tokenURI | External | - | - |
| flipRatio | Public | - | - |
| tokenRatioPriority | Public | - | - |

| SwapRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | PeripheryImmutableState |
| getPool | Private | - | - |
| pancakeV3SwapCallback | External | Can Modify State | - |
| exactInputInternal | Private | Can Modify State | - |
| exactInputSingle | External | Payable | checkDeadline |
| exactInput | External | Payable | checkDeadline |
| exactOutputInternal | Private | Can Modify State | - |
| exactOutputSingle | External | Payable | checkDeadline |
| exactOutput | External | Payable | checkDeadline |

| V3Migrator | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | PeripheryImmutableState |
| <Receive Ether> | External | Payable | - |
| migrate | External | Can Modify State | - |

## 4.3 Vulnerability Summary

### [N1] [Medium] Risk of excessive authority

**Category: Authority Control Vulnerability Audit**

**Content**

In the StableSwapRouter, the Owner role can set the stableSwapFactory to any address. If a fake stableSwapFactory address is passed in, the SmartRouterHelper will obtain a malicious transaction pair from the getStableInfo function, resulting in loss of funds. And this function is also missing the event log and 0 address check.

Code location:

router/contracts/StableSwapRouter.sol#33-39

```
    function setStableSwap(
        address _factory,
        address _info
    ) external onlyOwner {
        stableSwapFactory = _factory;
        stableSwapInfo = _info;
    }
```

**Solution**

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

**Status**

Fixed; After communication with the project team, they expressed that they will follow this and transfer the ownership to the multisig after deploying.

### [N2] [Suggestion] Token compatibility issue

**Category: Design Logic Audit**

**Content**

1.In the StableSwapRouter, the exactOutputStableSwap and exactInputStableSwap functions will call swap to transfer by the pay function via the input data amountIn and the input data is recorded directly into it. In the *swap function, although amountIn* is re-recorded. However, the amountIn_ data is still recorded in the _swap function when the exchange is performed in the swapContract of the transferring third-party contract. If the third-party contract token balance is used to directly participate in the calculation, the contract cannot be compatible with the rebase token.

Code location:

router/contracts/StableSwapRouter.sol#41-56

```
    function _swap(
        address[] memory path,
        uint256[] memory flag
    ) private {
        require(path.length - 1 == flag.length);

        for (uint256 i; i < flag.length; i++) {
            (address input, address output) = (path[i], path[i + 1]);
            (uint256 k, uint256 j, address swapContract) =
SmartRouterHelper.getStableInfo(stableSwapFactory, input, output, flag[i]);
            uint256 amountIn_ = IERC20(input).balanceOf(address(this));
            TransferHelper.safeApprove(input, swapContract, amountIn_);
            IStableSwap(swapContract).exchange(k, j, amountIn_, 0);
        }

        refund();
    }
```

2.In the PancakeV3Pool contract, during the mint and flash operations, unexpected ERC20 token interest behavior might allow token interest to count toward the amount of tokens required for the UniswapV3Pool.mint and flash functions, enabling the user to avoid paying in full.

Code location:

v3-core/contracts/PancakeV3Pool.sol#496-502, 829–941

```
        uint256 balance0Before;
        uint256 balance1Before;
        if (amount0 > 0) balance0Before = balance0();
        if (amount1 > 0) balance1Before = balance1();
        IPancakeV3MintCallback(msg.sender).pancakeV3MintCallback(amount0, amount1,
data);
        if (amount0 > 0) require(balance0Before.add(amount0) <= balance0(), 'M0');
        if (amount1 > 0) require(balance1Before.add(amount1) <= balance1(), 'M1');
```

**Solution**

It is recommended that the project team conduct a compatibility review when listing tokens, and the third-party contract interface is to pay attention to this compatibility issues.

**Status**

Acknowledged; After communication with the project team, they expressed that they won't list any rebase tokens in our stable swap, this sc is only for our stable pairs like BUSD/USDT/USDC/DAI.

### [N3] [Suggestion] Missing event records

**Category: Others**

**Content**

1.In the PancakeV3FactoryOwner contract, the owner role can set the lmPoolDeployer address, but there is no event log and 0 address check.

Code location:

v3-core/contracts/PancakeV3FactoryOwner.sol#49-51

```
    function setLmPoolDeployer(address _lmPoolDeployer) external onlyOwner {
        lmPoolDeployer = _lmPoolDeployer;
    }
```

2.In the PancakeV3PoolDeployer contract, the owner role can set the factoryAddress address, but there is no event log.

Code location:

v3-core/contracts/PancakeV3PoolDeployer.sol#27-30

```
function setFactoryAddress(address _factoryAddress) external {
    require(factoryAddress == address(0), "already initialized");
    factoryAddress = _factoryAddress;
}
```

3.In the StableSwapRouter, the owner role can set the stableSwapFactory and stableSwapInfo addresses, but there is no event log and 0 address check.

Code location:

router/contracts/StableSwapRouter.sol#33-39

```
function setStableSwap(
    address _factory,
    address _info
) external onlyOwner {
    stableSwapFactory = _factory;
    stableSwapInfo = _info;
}
```

**Solution**

It is recommended to record events and 0 address checks when sensitive parameters are modified for self-inspection or community review, and add the 0 address check.

**Status**

Fixed

### [N4] [Suggestion] Risk of initial operation

**Category: Race Conditions Vulnerability**

**Content**

In the PancakeV3Pool contract, by calling the initialize function to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize and there is no access control verification for the initialize functions

Code location:

v3-core/contracts/PancakeV3Pool.sol#277-305

```
    function initialize(uint160 sqrtPriceX96) external override {
        require(slot0.sqrtPriceX96 == 0, 'AI');

        int24 tick = TickMath.getTickAtSqrtRatio(sqrtPriceX96);

        (uint16 cardinality, uint16 cardinalityNext) =
observations.initialize(_blockTimestamp());

        slot0 = Slot0({
            sqrtPriceX96: sqrtPriceX96,
            tick: tick,
            observationIndex: 0,
            observationCardinality: cardinality,
            observationCardinalityNext: cardinalityNext,
            feeProtocol: 209718400,
            unlocked: true
        });

        if (fee == 100) {
            slot0.feeProtocol = 216272100;
        } else if (fee == 500) {
            slot0.feeProtocol = 222825800;
        } else if (fee == 2500) {
            slot0.feeProtocol = 209718400;
        } else if (fee == 10000) {
            slot0.feeProtocol = 209718400;
        }

        emit Initialize(sqrtPriceX96, tick);
    }
```

**Solution**

It is suggested that the initialize operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

**Status**

Acknowledged

### [N5] [Suggestion] Risk of denial of service

**Category: Gas Optimization Audit**

**Content**

In the UniswapV3Pool contract, the swap function can be disrupted by forcing the loop to go through too many operations, potentially trapping the swap due to a lack of gas.

Code location:

v3-core/contracts/PancakeV3Pool.sol#660-881

```
    while (state.amountSpecifiedRemaining != 0 && state.sqrtPriceX96 !=
sqrtPriceLimitX96) {
            StepComputations memory step;
            step.sqrtPriceStartX96 = state.sqrtPriceX96;
        ...
        state.tick = zeroForOne ? step.tickNext - 1 : step.tickNext;
            } else if (state.sqrtPriceX96 != step.sqrtPriceStartX96) {
                // recompute unless we're on a lower tick boundary (i.e. already
transitioned ticks), and haven't moved
                state.tick = TickMath.getTickAtSqrtRatio(state.sqrtPriceX96);
            }
        }
```

**Solution**

It's recommended to determine a reasonable minimum tick spacing requirement, consider setting a minimum for liquidity per position or make sure that all parameters that the owner can enable (likes fee level and tick spacing) have bounds that lead to expected behavior.

**Status**

Acknowledged; After communication with the project team, they expressed that their minimum fee is 100 with the minimum tickSpacing 1 for the latest upgrade.

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
|---|---|---|---|
| 0X002303130001 | SlowMist Security Team | 2023.03.07 - 2023.03.13 | Low Risk |

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 4 suggestion vulnerabilities. All the findings were fixed. The code was not 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.

**Official Website**

www.slowmist.com

**E-mail**

team@slowmist.com

**Twitter**

@SlowMist_Team

**Github**

https://github.com/slowmist