# Smart Contract Security Audit Report

[2021]

# Table Of Contents

# 1 Executive Summary

On 2021.05.17, the SlowMist security team received the Cook Finance team's security audit application for Cook Finance, 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 party 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. |

| Level | Description |
| --- | --- |
| 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:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Aduit
- Design Logic Audit
- Scoping and Declarations Audit

# 3 Project Overview

## 3.1 Project Introduction

Audit File information:

UniswapV2IndexExchangeAdapter

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/integration/index-exchange/UniswapV2IndexExchangeAdapter.sol

Commit: 6a9b5d9c2fa0001c30164f45e578ab0d7d153a46

BasicIssuanceModule

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/modules/BasicIssuanceModule.sol

Commit: c7c89b78d97e672a2bd8e046de5d0f7bb3643ae8

UniswapPairPriceAdapter

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/pr

otocol/integration/UniswapPairPriceAdapter.sol commit: bdfa6fb4e4261887d6d6c09bc2cd1085e98e9d74

SingleIndexModule

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/modules/SingleIndexModule.sol

commit: c7c89b78d97e672a2bd8e046de5d0f7bb3643ae8

TradeModule

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/modules/TradeModule.sol

commit: c7c89b78d97e672a2bd8e046de5d0f7bb3643ae8

WrapModule

Github: https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/modules/WrapModule.sol

commit: c7c89b78d97e672a2bd8e046de5d0f7bb3643ae8

## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | Risk of external call | Others | Suggestion | Confirmed |
| N2 | Risk of sandwich attack | Design Logic Audit | Low | Confirmed |
| N3 | Risk of re-initialization | Design Logic Audit | Medium | Confirmed |
| N4 | No check the _receiveToken address | Design Logic Audit | Low | Confirmed |

# 4 Code Overview

## 4.1 Contracts Description

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:

| UniswapV2IndexExchangeAdapter | | | |
|---|---|---|---|
| **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| Constructor | Public | can modify state | - |
| getTradeCalldata | External | - | - |
| getSpender | External | - | - |

| BasicIssuanceModule | | | |
|---|---|---|---|
| **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| Constructor | Public | - | ModuleBase |
| issue | External | - | nonReentrant<br>onlyValidAndInitializedCK |
| redeem | External | - | nonReentrant<br>onlyValidAndInitializedCK |
| initialize | External | - | onlyCKManager<br>onlyValidAndPendingCK |
| removeModule | External | - | - |

| BasicIssuanceModule | | | |
|---|---|---|---|
| getRequiredComponentUnitsForIssue | Public | - | onlyValidAndInitializedCK |
| _callPreIssueHooks | Internal | - | - |

| SingleIndexModule | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| Constructor | Public | can modify state | ModuleBase |
| startRebalance | External | can modify state | onlyManagerAndValidCK |
| trade | External | can modify state | nonReentrant onlyAllowedTrader onlyEOA |
| tradeRemainingWHT | External | can modify state | nonReentrant onlyAllowedTrader onlyEOA |
| raiseAssetTargets | External | can modify state | nonReentrant onlyAllowedTrader |
| setTradeMaximums | External | can modify state | onlyManagerAndValidCK |
| setExchanges | External | can modify state | onlyManagerAndValidCK |
| setCoolOffPeriods | External | can modify state | onlyManagerAndValidCK |
| updateTraderStatus | External | can modify state | onlyManagerAndValidCK |
| updateAnyoneTrade | External | can modify state | onlyManagerAndValidCK |
| initialize | External | can modify state | onlyCKManager onlyValidAndPendingCK |
| removeModule | External | can modify state | - |
| getTargetUnits | External | - | - |
| getRebalanceComponents | External | - | - |
| _validateTradeParameters | Internal | - | - |

| SingleIndexModule | | | |
|---|---|---|---|
| _calculateTradeSizeAndDirection | Internal | - | - |
| _buyUnderweight | Internal | can modify state | - |
| _sellOverweight | Internal | can modify state | - |
| _executeTrade | Internal | can modify state | - |
| _updatePositionState | Internal | can modify state | - |
| _getUniswapLikeTradeData | Internal | - | - |
| _noTokensToSell | Internal | - | - |
| _allTargetsMet | Internal | - | - |
| _normalizeTargetUnit | Internal | - | - |
| _isAllowedTrader | Internal | - | - |
| _validateArrays | Internal | - | - |

| TradeModule | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| Constructor> | Public | can modify state | ModuleBase |
| initialize | External | can modify state | onlyValidAndPendingCK onlyCKManager |
| trade | External | can modify state | nonReentrant onlyManagerAndValidCK |
| removeModule | External | can modify state | - |
| _createTradeInfo | Internal | - | - |
| _validatePreTradeData | Internal | - | - |

| TradeModule | | | |
|---|---|---|---|
| _executeTrade | Internal | can modify state | - |
| _validatePostTrade | Internal | - | - |
| _accrueProtocolFee | Internal | can modify state | - |
| _updateCKTokenPositions | Internal | can modify state | - |

| UniswapPairPriceAdapter | | | |
|---|---|---|---|
| **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| Constructor | Public - | can modify state | - |
| getPrice | External | - | - |
| addPool | External | can modify state | onlyOwner |
| removePool | External | can modify state | onlyOwner |
| getAllowedUniswapPools | External | - | - |
| _getUniswapPrice | Internal | - | - |

| WrapModule | | | |
|---|---|---|---|
| Constructor | Public | can modify state | ModuleBase |
| wrap | External | can modify state | nonReentrant onlyManagerAndValidCK |
| wrapWithHT | External | can modify state | nonReentrant onlyManagerAndValidCK |
| unwrap | External | can modify state | nonReentrant onlyManagerAndValidCK |
| unwrapWithHT | External | can modify state | nonReentrant onlyManagerAndValidCK |

| WrapModule | | | |
|---|---|---|---|
| initialize | External | can modify state | onlyCKManager |
| removeModule | External | can modify state | - |
| _validateInputs | Internal | - | - |
| _validateWrapAndUpdate | Internal | can modify state | - |
| _validateUnwrapAndUpdate | Internal | can modify state | - |
| _createWrapDataAndInvoke | Internal | can modify state | - |
| _createUnwrapDataAndInvoke | Internal | can modify state | - |
| _updatePosition | Internal | can modify state | - |
| _snapshotTargetAssetsBalance | Internal | - | - |

## 4.3 Vulnerability Summary

**[N1] [Suggestion] Risk of external call**

**Category: Others**

**Content**

BasicIssuanceModule has many external call, it suggested to be aware of the risk of external call

e.g. initialize function

```
    function initialize(
        ICKToken _ckToken,
        IManagerIssuanceHook _preIssueHook
    )
        external
        onlyCKManager(_ckToken, msg.sender)

        onlyValidAndPendingCK(_ckToken)
```

```
    {
        managerIssuanceHook[_ckToken] = _preIssueHook;
        //SlowMist// Please be aware of the risk of external call
        _ckToken.initializeModule();
    }
```

**Solution**

**Status**

Confirmed; The project party confirms that risk and will pay attention to the risk of external call

### [N2] [Low] Risk of sandwich attack

**Category: Design Logic Audit**

**Content**

In SingleIndexModule, the trade function does not set a slippage protection when trade, which may results in sandwich attack.

```
    function trade(address _component) external nonReentrant
onlyAllowedTrader(msg.sender) onlyEOA() virtual {

        _validateTradeParameters(_component);
        //@audit not check assetInfo[component] before trade.
        (
            bool isBuy,
            uint256 tradeAmount
            //SlowMist// There is no slippage protection inside
_calculateTradeSizeAndDirection
        ) = _calculateTradeSizeAndDirection(_component);

        if (isBuy) {
            //@audit no check if tradeAmount is larger than component
            _buyUnderweight(_component, tradeAmount);
        } else {
            _sellOverweight(_component, tradeAmount);
        }

        assetInfo[_component].lastTradeTimestamp = block.timestamp;
    }
```

**Solution**

Check the swap sliappage

**Status**

Confirmed; The project side confirmed that there is a trade amount limitation that guarantee the trade amount is small enough to ignore the slippage issue.

### [N3] [Medium] Risk of re-initialization

**Category: Design Logic Audit**

**Content**

The initialize function of SingleIndexModule, TradeModule and WarpModule were not restricted to can only be called once.

```
//SlowMist// No restriction on initialization
function initialize(ICKToken _index)
        external
        onlyCKManager(_index, msg.sender)
        onlyValidAndPendingCK(_index)
    {
        require(address(index) == address(0), "Module already in use");

        ICKToken.Position[] memory positions = _index.getPositions();

        for (uint256 i = 0; i < positions.length; i++) {
            ICKToken.Position memory position = positions[i];
            assetInfo[position.component].targetUnit = position.unit.toUint256();
            assetInfo[position.component].lastTradeTimestamp = 0;
        }

        index = _index;
        _index.initializeModule();
    }
```

**Solution**

Restrict the re-initialization

**Status**

Confirmed; The project side confirmed that the initialize functoion can not be called again without removing the module first.

### [N4] [Low] No check the _receiveToken address

**Category: Design Logic Audit**

**Content**

The trade function of TradeModule does not check if _receiveToken is allowed to trade.

```
    function trade(
        ICKToken _ckToken,
        string memory _exchangeName,
        address _sendToken,
        uint256 _sendQuantity,
        address _receiveToken,
        uint256 _minReceiveQuantity,
        bytes memory _data
    )
        external
        nonReentrant
        onlyManagerAndValidCK(_ckToken)
    {
        TradeInfo memory tradeInfo = _createTradeInfo(
            _ckToken,
            _exchangeName,
            _sendToken,
            _receiveToken,
            _sendQuantity,
            _minReceiveQuantity
        );

        _validatePreTradeData(tradeInfo, _sendQuantity);

        _executeTrade(tradeInfo, _data);

        uint256 exchangedQuantity = _validatePostTrade(tradeInfo);

        uint256 protocolFee = _accrueProtocolFee(tradeInfo, exchangedQuantity);

        (
```

```
            uint256 netSendAmount,
            uint256 netReceiveAmount
        ) = _updateCKTokenPositions(tradeInfo);

        emit ComponentExchanged(
            _ckToken,
            _sendToken,
            _receiveToken,
            tradeInfo.exchangeAdapter,
            netSendAmount,
            netReceiveAmount,
            protocolFee
        );
    }
```

**Solution**

Check the _receiveToken

**Status**

Confirmed; The project side confirmed that this is a feature, the address is up the stack with a manager contract.

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002105200003 | SlowMist Security Team | 2021.05.17 - 2021.05.20 | Passed |

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 issue, 2 low risk issue and 1 suggestion were confirmed and being fixed; All other 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