Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.08.20, the SlowMist security team received the Cook Finance team's security audit application for Cook Index, 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 version:

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

commit: d0fedf5d4bbf175beb89175b5b1f45bc4cd42411

https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/integration/wrap/VesperWrapAdapter.sol

commit: b82100f18d64fb4accab0b3572408f197d4b6d4f

https://github.com/CookFinance/cook-index/blob/main/cook-protocol-contracts/contracts/protocol/integration/oracles/VesperVaultOracle.sol

commit: 03ceae798002a7e65fe95bfb5e1068a37fdc13e3

Audit scope:

cook-protocol-contracts/contracts/protocol/integration/oracles/VesperVaultOracle.sol

cook-protocol-contracts/contracts/protocol/integration/wrap/VesperWrapAdapter.sol

cook-protocol-contracts/contracts/protocol/modules/IssuanceModule.sol

(***This audit does not include any external calls and import contracts.***)

Fixed version:

https://github.com/CookFinance/cook-index/

commit: aa254c0e2a97939728925202f55456a766a97edd

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Wrong slippage check issue	Design Logic Audit	High	Fixed
N2	Logical redundancy issue	Design Logic Audit	Suggestion	Ignored
N3	Risk of external calls	Unsafe External Call Audit	Suggestion	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:

IssuanceModule			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ModuleBase
issue	External	Can Modify State	nonReentrant onlyValidAndInitializedCK
_issueWithSingleToken	Internal	Can Modify State	-
issueWithSingleToken	External	Can Modify State	nonReentrant onlyValidAndInitializedCK
issueWithEther	External	Payable	nonReentrant onlyValidAndInitializedCK
redeem	External	Can Modify State	nonReentrant onlyValidAndInitializedCK
redeemToSingleToken	External	Can Modify State	nonReentrant onlyValidAndInitializedCK
initialize	External	Can Modify State	onlyCKManager onlyValidAndPendingCK
removeModule	External	Can Modify State	-
setExchanges	External	Can Modify State	onlyOwner
setWrapAdapters	External	Can Modify State	onlyOwner
getRequiredComponentIssuanceUnits	Public	-	-
_getTotalIssuanceUnits	Internal	-	-
_callPreIssueHooks	Internal	Can Modify State	-

IssuanceModule			
_executeExternalPositionHooks	Internal	Can Modify State	-
_exchangeIssueTokenToDefaultPositions	Internal	Can Modify State	-
_exchangeDefaultPositionsToRedeemToken	Internal	Can Modify State	-
_snapshotTargetTokenBalance	Internal	-	-
_validatePostTrade	Internal	-	-
_validatePreTradeData	Internal	-	-
_createTradeInfo	Internal	-	-
_executeTrade	Internal	Can Modify State	-
_trade	Internal	Can Modify State	-
_wrap	Internal	Can Modify State	-
_unwrap	Internal	Can Modify State	-
_validateAndWrap	Internal	Can Modify State	-
_validateAndUnwrap	Internal	Can Modify State	-
_validateInputs	Internal	-	-
_createWrapDataAndInvoke	Internal	Can Modify State	-
_createUnwrapDataAndInvoke	Internal	Can Modify State	-

VesperWrapAdapter

VesperWrapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getDepositUnderlyingTokenAmount	External	-	_onlyValidTokenPair
getWithdrawUnderlyingTokenAmount	External	-	_onlyValidTokenPair
getSpenderAddress	External	-	-
getWrapCallData	External	-	_onlyValidTokenPair
getUnwrapCallData	External	-	_onlyValidTokenPair
validTokenPair	Internal	-	-

VesperVaultOracle			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
read	External	-	-

4.3 Vulnerability Summary

[N1] [High] Wrong slippage check issue

Category: Design Logic Audit

Content

In the IssuanceModule contract, the `_createTradeInfo` function is used to create a structure containing trade data. Among them, it will obtain the thresholdAmounts parameter offset by slippage through the getMinAmountsOut function and the getMaxAmountsIn function. After the trade data is created, the trade operation will be executed

through the `_executeTrade` function, which will use the thresholdAmounts parameter as the minimum amounts to receive for trading on uniswap. However, since slippage check and trade execution are carried out in the same transaction, the thresholdAmounts parameter will still be affected by the last swap transaction of uniswap. Therefore, the slippage check cannot play a protective role.

Code location: contracts/protocol/modules/IssuanceModule.sol

```
    function _createTradeInfo(
        ICKToken _ckToken,
        IExchangeAdapter _exchangeAdapter,
        address _sendToken,
        address _receiveToken,
        uint256 _exactQuantity,
        bool _isSendTokenFixed,
        uint256 _slippage
    )
        internal
        view
        returns (TradeInfo memory)
    {
        uint256 thresholdAmount;
        address[] memory path;
        if (_sendToken == address(weth) || _receiveToken == address(weth)) {
            path = new address[](2);
            path[0] = _sendToken;
            path[1] = _receiveToken;
            uint256[] memory thresholdAmounts = _isSendTokenFixed ?
_exchangeAdapter.getMinAmountsOut(_exactQuantity, path, _slippage) :
_exchangeAdapter.getMaxAmountsIn(_exactQuantity, path, _slippage);
            thresholdAmount = _isSendTokenFixed ? thresholdAmounts[1] :
thresholdAmounts[0];
        } else {
            path = new address[](3);
            path[0] = _sendToken;
            path[1] = address(weth);
            path[2] = _receiveToken;
            uint256[] memory thresholdAmounts = _isSendTokenFixed ?
_exchangeAdapter.getMinAmountsOut(_exactQuantity, path, _slippage) :
_exchangeAdapter.getMaxAmountsIn(_exactQuantity, path, _slippage);
            thresholdAmount = _isSendTokenFixed ? thresholdAmounts[2] :
thresholdAmounts[0];
        }
```

```
        TradeInfo memory tradeInfo;
        tradeInfo.ckToken = _ckToken;
        tradeInfo.exchangeAdapter = _exchangeAdapter;
        tradeInfo.sendToken = _sendToken;
        tradeInfo.receiveToken = _receiveToken;
        tradeInfo.totalSendQuantity = _isSendTokenFixed ? _exactQuantity :
thresholdAmount;
        tradeInfo.totalReceiveQuantity = _isSendTokenFixed ? thresholdAmount :
_exactQuantity;
        tradeInfo.preTradeSendTokenBalance = _snapshotTargetTokenBalance(_ckToken,
_sendToken);
        tradeInfo.preTradeReceiveTokenBalance = _snapshotTargetTokenBalance(_ckToken,
_receiveToken);
        tradeInfo.data = _isSendTokenFixed ? _exchangeAdapter.generateDataParam(path,
true) : _exchangeAdapter.generateDataParam(path, false);
        return tradeInfo;
    }
```

```
    function getMinAmountsOut(uint256 amountIn, address[] memory path, uint256
slippage) external view returns (uint256[] memory amounts) {
        amounts = new uint256[](path.length);
        amounts = IUniswapV2Router02(router).getAmountsOut(amountIn, path);
        for (uint i = 1; i < path.length; i++) {
            amounts[i] =
amounts[i].preciseMul(PreciseUnitMath.PRECISE_UNIT.sub(slippage));
        }
    }

    function getMaxAmountsIn(uint256 amountOut, address[] memory path, uint256
slippage) external view returns (uint256[] memory amounts) {
        amounts = new uint256[](path.length);
        amounts = IUniswapV2Router02(router).getAmountsIn(amountOut, path);
        for (uint i = 0; i < path.length - 1; i++) {
            amounts[i] =
amounts[i].preciseMul(PreciseUnitMath.PRECISE_UNIT.add(slippage));
        }
    }
```

Solution

It is recommended to use an oracle with delayed price feed for slippage check.

Status

Fixed; After communicating with the project party, the project party decided that the final price will be compared with the price provided by the delayed price feeder. If the price deviation is too large, it will not be allowed to execute.

[N2] [Suggestion] Logical redundancy issue

Category: Design Logic Audit

Content

In the VesperWrapAdapter contract, getSpenderAddress is used to obtain the source token address of the wrap token, but the actual function logic directly returns the passed _wrappedToken parameter. This seems to be different from what the function comments indicate.

Code location: contracts/protocol/integration/wrap/VesperWrapAdapter.sol

```
    function getSpenderAddress(address /* _underlyingToken */, address _wrappedToken)
external view returns(address) {
        return _wrappedToken;
    }
```

Solution

It is recommended to self-check the design logic here to ensure that it meets the expected design. If it is an expected design, it is recommended to use _wrappedToken directly.

Status

Ignored; After communicating with the project party, the project party stated that this is an interface function, and different protocols have different implementations. vespperwrapadapter will return _wrappedToken directly, but AaveWrapAdapter will return the address of aavelendingpoolcore. The project needs a common method to obtain the sender address of different protocols.

[N3] [Suggestion] Risk of external calls

Category: Unsafe External Call Audit

Content

There are a large number of external calls in the IssuanceModule contract, but the external call part is not within the scope of this audit. It is necessary to pay attention to the unknown risks of external calls.

Solution

None.

Status

Confirmed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002109010001	SlowMist Security Team	2021.08.20 - 2021.08.27	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 high risk, 2 suggestion vulnerabilities. And 1 suggestion vulnerabilities were confirmed and being fixed; 1 suggestion vulnerabilities were ignored; 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