Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.10.08, the SlowMist security team received the iotube team's security audit application for iotube, 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:

- 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 Audit

- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit Version:

https://github.com/iotubeproject/iotube-contracts/tree/master/contracts

commit: 2de0aedb74c04cd08ecb69a1378e1efd0ae81aaf

Fixed Version:

https://github.com/iotubeproject/iotube-contracts/tree/master/contracts

commit: fbbf1674bd463bcee42e1cf54478a8d2545dd1f7

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Business logic error	Design Logic Audit	High	Fixed
N2	The range of values is not limited	Others	Low	Confirmed
N3	Authority transfer enhancement	Authority Control Vulnerability	Suggestion	Confirmed
N4	unsafe external call risk	Unsafe External Call Audit	Critical	Fixed
N5	Excessive authority issue	Authority Control Vulnerability	High	Confirmed
N6	Business logic flaws	Design Logic Audit	Medium	Confirmed

NO	Title	Category	Level	Status
N7	Excessive authority issue	Authority Control Vulnerability	Medium	Confirmed
N8	Coding optimization	Others	Suggestion	Confirmed
N9	Other safety reminders	Others	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:

AssetRegistry			
Function Name	Visibility	Mutability	Modifiers
assetID	Public	-	-
originalAssetByID	Public	-	-
numOfAssets	Public	-	-
assetOnTubeByID	Public	-	-
assetOnTube	Public	-	-
addOriginalAsset	Public	Can Modify State	onlyOperator

AssetRegistry			
addAssetOnTube	Public	Can Modify State	onlyOperator
activateAsset	Public	Can Modify State	onlyOperator
deactivateAsset	Public	Can Modify State	onlyOperator
activateTube	Public	Can Modify State	onlyOperator
deactivateTube	Public	Can Modify State	onlyOperator
grant	Public	Can Modify State	onlyOwner
revoke	Public	Can Modify State	onlyOwner

AssetUpperBound			
Function Name	Visibility	Mutability	Modifiers
setUpperBound	Public	Can Modify State	onlyOwner

CrosschainCoinRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Fallback>	External	Payable	-
resetAllowance	Public	Can Modify State	-
swapCoinForCrosschainCoin	Public	Payable	-
swapCrosschainCoinForCoin	Public	Can Modify State	-
swapWrappedCoinForCrosschainCoin	Public	Can Modify State	-
swapCrosschainCoinForWrappedCoin	Public	Can Modify State	-

CrosschainCoinRouter			
swapCoinForWrappedCoin	Public	Payable	-
swapWrappedCoinForCoin	Public	Can Modify State	-

CrosschainERC20			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
transferMintership	Public	Can Modify State	onlyMinter
deposit	Public	Can Modify State	-
depositTo	Public	Can Modify State	-
withdraw	Public	Can Modify State	-
withdrawTo	Public	Can Modify State	-
mint	Public	Can Modify State	onlyMinter

CrosschainERC20Factory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createForeignToken	Public	Can Modify State	onlyOwner
createLocalToken	Public	Can Modify State	onlyOwner

CrosschainERC721			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721

CrosschainERC721			
transferMintership	Public	Can Modify State	onlyMinter
deposit	Public	Can Modify State	-
depositTo	Public	Can Modify State	-
withdraw	Public	Can Modify State	-
withdrawTo	Public	Can Modify State	-
safeMint	Public	Can Modify State	onlyMinter

Ledger			
Function Name	Visibility	Mutability	Modifiers
record	Public	Can Modify State	onlyOwner
get	Public	-	-

Lord			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
burn	Public	Can Modify State	onlyOwner
mint	Public	Can Modify State	onlyOwner
mintNFT	Public	Can Modify State	onlyOwner
upgrade	Public	Can Modify State	onlyOwner
_callOptionalReturn	Private	Can Modify State	-

LPToken

LPToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	onlyOwner
_beforeTokenTransfer	Internal	Can Modify State	-

OwnerPausable			
Function Name	Visibility	Mutability	Modifiers
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner

Swap			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	OwnerPausable ReentrancyGuard
getA	External	-	-
getAPrecise	External	-	-
getToken	Public	-	-
getTokenIndex	External	-	-
getDepositTimestamp	External	-	-
getTokenBalance	External	-	-
getVirtualPrice	External	-	-
calculateSwap	External	-	-

Swap			
calculateTokenAmount	External	-	-
calculateRemoveLiquidity	External	-	-
calculateRemoveLiquidityOneToken	External	-	-
calculateCurrentWithdrawFee	External	-	-
getAdminBalance	External	-	-
swap	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
addLiquidity	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidity	External	Can Modify State	nonReentrant deadlineCheck
removeLiquidityOneToken	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidityImbalance	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
updateUserWithdrawFee	External	Can Modify State	-
withdrawAdminFees	External	Can Modify State	onlyOwner
setAdminFee	External	Can Modify State	onlyOwner
setSwapFee	External	Can Modify State	onlyOwner
setDefaultDepositFee	External	Can Modify State	onlyOwner
setDefaultWithdrawFee	External	Can Modify State	onlyOwner
rampA	External	Can Modify State	onlyOwner

Swap			
stopRampA	External	Can Modify State	onlyOwner
setDevAddress	External	Can Modify State	onlyOwner

Tube			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
upgrade	Public	Can Modify State	onlyOwner
count	Public	-	-
pause	Public	Can Modify State	onlyOwner
unpause	Public	Can Modify State	onlyOwner
setFee	Public	Can Modify State	onlyOwner
depositTo	Public	Can Modify State	whenNotPaused
depositNFTTo	Public	Can Modify State	whenNotPaused
deposit	Public	Can Modify State	-
depositNFT	Public	Can Modify State	-
genKey	Public	-	-
genKeyForNFT	Public	-	-
concatKeys	Public	-	-
isSettled	Public	-	-
withdraw	Public	Can Modify State	whenNotPaused

Tube			
withdrawNFT	Public	Can Modify State	whenNotPaused
withdrawCoin	External	Can Modify State	onlyOwner
withdrawToken	External	Can Modify State	onlyOwner

TubeRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setRelayFee	Public	Can Modify State	onlyOwner
relayFee	Public	-	-
depositTo	Public	Payable	-
depositNFTTo	Public	Payable	-
withdrawCoin	External	Can Modify State	onlyOwner
withdrawToken	External	Can Modify State	onlyOwner

TubeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20

ValidatorRegistry			
Function Name	Visibility	Mutability	Modifiers
register	Public	Can Modify State	-
getFile	Public	-	-

Verifier			
Function Name	Visibility	Mutability	Modifiers
size	Public	-	-
get	Public	-	-
addAll	Public	Can Modify State	onlyOwner
removeAll	Public	Can Modify State	onlyOwner
verify	Public	-	-
recover	Internal	-	-

WIOTX			
Function Name	Visibility	Mutability	Modifiers
<Fallback>	External	Payable	-
deposit	Public	Payable	-
withdraw	Public	Can Modify State	-
totalSupply	Public	-	-
approve	Public	Can Modify State	-
transfer	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-

4.3 Vulnerability Summary

[N1] [High] Business logic error

Category: Design Logic Audit

Content

When originalAssetByID gets the data of originalAssets, the _assetID passed in should be reduced by 1, and then readout.

activateAsset, deactivateAsset function directly passes in _assetID, and then read originalAssets[_assetID],

The judgment is `_assetID <= originalAssets.length`.

There are two issues here:

1. When _assetID == originalAssets.length, originalAssets[_assetID] cannot read data.
2. The originalAssets data obtained here is not obtained using the originalAssetByID function, and the business logic needs to be confirmed.

- contracts/AssetRegistry.sol#L101-L133

```
function activateAsset(uint256 _assetID, uint256 _tubeID) public onlyOperator {
       require(_assetID > 0 && _assetID <= originalAssets.length, "invalid asset
id");
       Asset storage oa = originalAssets[_assetID];
       if (_tubeID == 0 || oa.tubeID == _tubeID) {
           if (oa.active == false) {
               oa.active = true;
               emit AssetActivated(_assetID, oa.tubeID);
           }
       } else {
           Asset storage sa = shadowAssets[_assetID][_tubeID];
           if (sa.asset != address(0) && sa.active == false) {
               sa.active = true;
               emit AssetActivated(_assetID, _tubeID);
           }
       }
   }
function deactivateAsset(uint256 _assetID, uint256 _tubeID) public onlyOperator {
       require(_assetID > 0 && _assetID <= originalAssets.length, "invalid asset
id");
       Asset storage oa = originalAssets[_assetID];
       if (_tubeID == 0 || oa.tubeID == _tubeID) {
           if (oa.active == true) {
               oa.active = false;
```

```
                emit AssetDeactivated(_assetID, oa.tubeID);
            }
        } else {
            Asset storage sa = shadowAssets[_assetID][_tubeID];
            if (sa.asset != address(0) && sa.active == true) {
                sa.active = false;
                emit AssetDeactivated(_assetID, _tubeID);
            }
        }
    }
```

originalAssetIDs[_tubeID][_asset] = id; The record is originalAssets.length;, so to take the value of originalAssets, the index should be originalAssetIDs[_tubeID][_asset]-1, combined with the processing logic here, you need to confirm the issues with the developer.

- contracts/AssetRegistry.sol#L74-L84

```
function addOriginalAsset(uint256 _tubeID, address _asset) public onlyOperator
returns (uint256) {
        require(_tubeID > 0 && _asset != address(0), "invalid parameter");
        uint256 id = assetID(_tubeID, _asset);
        if (id == 0) {
            originalAssets.push(Asset(_tubeID, _asset, true));
            id = originalAssets.length;
            originalAssetIDs[_tubeID][_asset] = id;
            emit NewOriginalAsset(_tubeID, _asset, id);
        }
        return id;
    }
```

Solution

It is recommended to communicate with the developer to confirm the business logic.

Status

Fixed; After communication with the iotube team, the issue has been fixed in the commit:

2aef5cd8f72f84fdedc4f10039330d598892162f.

[N2] [Low] The range of values is not limited

Category: Others

Content

The owner can set the value corresponding to upperBounds[_asset], but there is no limit on the value range.

- contracts/AssetUpperBound.sol#L12-L15

```
function setUpperBound(address _asset, uint256 _upperBound) public onlyOwner {
    upperBounds[_asset] = _upperBound;
    emit UppserBoundSet(_asset, _upperBound);
}
```

There is no restriction on the fees range, and there is an issue of conditional competition. If the Owner changes the Fees, the expected Fees of the user will be inconsistent with the actual ones.

- contracts/Tube.sol#L92-L94

```
function setFee(uint256 _tubeID, uint256 _fee) public onlyOwner {
    fees[_tubeID] = _fee;
}
```

Solution

It is recommended to suspend the business first and then change the fee configuration to avoid the actual fee inconsistent with the user's expectations and add the limit of the value range of fees and upperBound.

Status

Confirmed

[N3] [Suggestion] Authority transfer enhancement

Category: Authority Control Vulnerability

Content

There is no pending and accept mechanism for authority transfer to avoid loss of authority

- contracts/CrosschainERC20.sol#L34-L37

```
function transferMintership(address _newMinter) public onlyMinter {
    minter = _newMinter;
```

```
        emit MinterSet(_newMinter);
    }
```

- contracts/CrosschainERC721.sol#L29-L32

```
    function transferMintership(address _newMinter) public onlyMinter {
        minter = _newMinter;
        emit MinterSet(_newMinter);
    }
```

Solution

It is recommended to have a pending and accept operation when transferring minting authority to avoid losing authority.

Status

Confirmed

[N4] [Critical] unsafe external call risk

Category: Unsafe External Call Audit

Content

The withdraw function in the contract code does not check the whitelist of _recipient and _data.

There is an unsafe external call `(success,) = _recipient.call(_data);`.

The attacker can use _recipient.call(_data); to call any function of the lord contract, or transfer the token approved by the user to the Tube contract.

- contracts/Tube.sol#L179-L207

```
    function withdraw(
        uint256 _srcTubeID,
        uint256 _txIdx,
        address _token,
        address _recipient,
        uint256 _amount,
        bytes memory _data,
        bytes memory _signatures
    ) public whenNotPaused {
        require(_amount != 0, "amount is 0");
```

```
        require(_recipient != address(0), "invalid recipient");
        require(_signatures.length % 65 == 0, "invalid signature length");
        bytes32 key = genKey(_srcTubeID, _txIdx, _token, _recipient, _amount, _data);
        ledger.record(key);
        (bool isValid, address[] memory signers) = verifier.verify(key, _signatures);
        require(isValid, "insufficient validators");
        bool success = true;
        if (_data.length > 0) {
            lord.mint(_token, address(this), _amount);
            IERC20(_token).safeApprove(_recipient, _amount);
            (success, ) = _recipient.call(_data);
            if (!success) {
                IERC20(_token).safeDecreaseAllowance(_recipient, _amount);
            }
        } else {
            lord.mint(_token, _recipient, _amount);
        }
        emit Settled(key, signers, success);
    }
```

Solution

It is recommended to add a whitelist or blacklist mechanism for externally called contracts to prevent attackers from making arbitrary external calls through the Tube contract, and add an nonReentrant lock.

Status

Fixed; The issue has been fixed in commit: fbbf1674bd463bcee42e1cf54478a8d2545dd1f7.

[N5] [High] Excessive authority issue

Category: Authority Control Vulnerability

Content

The Owner of the Tube contract has too much authority. The owner of the Lord contract can be changed, and the owner of the Lord contract can execute mint and burn arbitrarily. This will affect the user's assets.

- contracts/Tube.sol#L71-L78

```
    function upgrade(address _newTube) public onlyOwner {
        if (ledger.owner() == address(this)) {
            ledger.transferOwnership(_newTube);
        }
```

```
        if (lord.owner() == address(this)) {
            lord.transferOwnership(_newTube);
        }
    }
```

- contracts/Lord.sol#L94-L131

```
function burn(
        address _token,
        address _sender,
        uint256 _amount
    ) public onlyOwner {
        if (address(standardTokenList) != address(0) &&
standardTokenList.isAllowed(_token)) {
            // transfer token to standardTokenList
            _callOptionalReturn(
                _token,
                abi.encodeWithSelector(IToken(_token).transferFrom.selector, _sender,
tokenSafe, _amount)
            );
            return;
        }
        if (address(proxyTokenList) != address(0) &&
proxyTokenList.isAllowed(_token)) {
            _callOptionalReturn(
                _token,
                abi.encodeWithSelector(IToken(_token).transferFrom.selector, _sender,
address(this), _amount)
            );
            _callOptionalReturn(_token,
abi.encodeWithSelector(IToken(_token).burn.selector, _amount));
            return;
        }
        _callOptionalReturn(_token,
abi.encodeWithSelector(IToken(_token).burnFrom.selector, _sender, _amount));
    }

    function mint(
        address _token,
        address _recipient,
        uint256 _amount
    ) public onlyOwner {
        if (address(standardTokenList) != address(0) &&
```

```
standardTokenList.isAllowed(_token)) {
            require(tokenSafe.mint(_token, _recipient, _amount), "token safe mint
failed");
            return;
        }
        if (address(proxyTokenList) != address(0) &&
proxyTokenList.isAllowed(_token)) {
            require(minterPool.mint(_token, _recipient, _amount), "proxy token mint
failed");
        }
        _callOptionalReturn(_token,
abi.encodeWithSelector(IToken(_token).mint.selector, _recipient, _amount));
    }

    function mintNFT(
        address _token,
        uint256 _tokenID,
        address _recipient,
        bytes memory _data
    ) public onlyOwner {
        IERC721Mintable(_token).safeMint(_recipient, _tokenID, _data);
    }

    function upgrade(address _newLord) public onlyOwner {
        if (minterPool.owner() == address(this)) {
            _callOptionalReturn(
                address(tokenSafe),
                abi.encodeWithSelector(minterPool.transferOwnership.selector,
_newLord)
            );
        }
        if (tokenSafe.owner() == address(this)) {
            _callOptionalReturn(
                address(tokenSafe),
                abi.encodeWithSelector(tokenSafe.transferOwnership.selector,
_newLord)
            );
        }
    }
```

Solution

It is recommended to transfer the authority of the Owner role to a timelock contract or a self-care contract. At least

the Owner role should be a multi-signature address.

Status

Confirmed; The project team response: the owner of the lord is the Tube contract.

[N6] [Medium] Business logic flaws

Category: Design Logic Audit

Content

The swapCoinForCrosschainCoin function will call cerc20.depositTo.

- contracts/CrosschainCoinRouter.sol#L37-L40

```
function swapCoinForCrosschainCoin(uint256 _amount) public payable {
    wrappedCoin.deposit{value: _amount}();
    cerc20.depositTo(msg.sender, _amount);
}
```

cerc20.depositTo will call safeTransferFrom, where msg.sender is the CrosschainCoinRouter contract, but CrosschainCoinRouter has authorized CrosschainERC20 contract operation assets.

- contracts/CrosschainERC20.sol#L43-L47

```
function depositTo(address _to, uint256 _amount) public {
    require(address(coToken) != address(0), "no co-token");
    coToken.safeTransferFrom(msg.sender, address(this), _amount);
    _mint(_to, _amount);
}
```

Although the allowance is set to -1, under extreme conditions, the continuous consumption quota will still be reduced to no quota. At this time, the contract cannot be used without re-approve.

- contracts/CrosschainCoinRouter.sol#L20-L25

```
constructor(CrosschainERC20 _cerc20) public {
    ERC20 ct = _cerc20.coToken();
    cerc20 = _cerc20;
    ct.safeApprove(address(cerc20), uint256(-1));
```

```
        wrappedCoin = WrappedCoin(address(ct));
    }
```

Solution

It is recommended to set up a re-approve function, which can be re-approve after the allowance is used up

Status

Confirmed

[N7] [Medium] Excessive authority issue

Category: Authority Control Vulnerability

Content

Owner can set the relayfee arbitrarily, and there is no limit on the value range. When the relayfee is set to a large value, most of the user's funds will be used to pay the relayFees.

- contracts/TubeRouter.sol#L49-L55

```
function setRelayFee(uint256 _tubeID, uint256 _fee) public onlyOwner {
    if (_fee == 0) {
        relayFees[_tubeID].exists = false;
    } else {
        relayFees[_tubeID] = RelayFee(_fee, true);
    }
}
```

Solution

It is recommended to transfer the authority of Owner to the timelock contract, and transfer the authority of the management role of timelock to the multi-signature address, and set the value range of the relayFees.

Status

Confirmed

[N8] [Suggestion] Coding optimization

Category: Others

Content

Owner can withdraw ETH and token in the contract. There are relayFees in the TubeRouter contract, and the Owner can withdraw the relayFees through the withdrawToken function, However, withdrawCoin and withdrawToken are used to extract the assets that were unexpectedly credited into the contract.

- contracts/TubeRouter.sol#L97-L106

```
function withdrawCoin(address payable _to) external onlyOwner {
        _to.transfer(address(this).balance);
    }

 function withdrawToken(address _to, IERC20 _token) external onlyOwner {
        uint256 balance = _token.balanceOf(address(this));
        if (balance > 0) {
            _token.safeTransfer(_to, balance);
        }
    }
```

- contracts/Tube.sol#L229-L238

```
    function withdrawCoin(address payable _to) external onlyOwner {
        _to.transfer(address(this).balance);
    }

    function withdrawToken(address _to, IERC20 _token) external onlyOwner {
        uint256 balance = _token.balanceOf(address(this));
        if (balance > 0) {
            _token.safeTransfer(_to, balance);
        }
    }
```

Solution

It is recommended that the operation of withdrawing the fee be implemented as a separate function. And the role of fee management should be set to multi-signature to avoid theft of the private key of the fee management role and the team's revenue will be transferred away.

Status

Confirmed

[N9] [Suggestion] Other safety reminders

Category: Others

Content

Note that when signing, make sure that the K value is not the same in the signature implementation. In the elliptic curve signature algorithm, if the random number is not safe enough and the same K value random number is used, there will be two transactions with the same R value. , So that the private key can be calculated, please pay attention to investigate similar cryptographic implementations.

Reference: https://panzhibiao.com/2019/03/13/important-random-k-and-fake-signatures/

To capture events in the cross-chain bridge, the implementation of subscribing to the events of the specified contract should be adopted to avoid the attacks of fake contract events

Solution

It is recommended to ensure that the K value cannot be the same and that the contract address to which the event belongs is iotube's project contract.

Status

Confirmed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002110180001	SlowMist Security Team	2021.10.08 - 2021.10.18	High Risk

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, 2 high risk, 2 medium risk, 1 low risk, 3 suggestion vulnerabilities. And 1 high risk, 2 medium risk, 1 low risk, 3 suggestion vulnerabilities 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