Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.06.20, the SlowMist security team received the Rango team's security audit application for Rango, 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.

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:

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
		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

Audit Version

https://github.com/rango-exchange/rango-contracts

commit: 4d6051e11f037214f00d3196d5d656c1f415b888

Fixed Version

https://github.com/rango-exchange/rango-contracts

commit: f273dfdb4065ef022e60954cef2ca6282ac61c79

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	Low	Confirmed
N2	Redundant code	Others	Suggestion	Fixed
N3	Compatibility reminder	Others	Suggestion	Fixed
N4	Business logic issue	Design Logic Audit	High	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

chain ID	contract name	contract address
56	rangoV1	proxy:0x2a7813412b8da8d18Ce56FE763B9eb264D8e28a8,implementation:0x50d018594f18435fa1d40e4f1f6ec73ac7f2e621
56	cbridge	0x69456B55E0868E7C4012d288aD6F92f89eD7697e
56	multichain	0x4A17056B30B1c50155fA33dB4AbC656aBc31b909
137	rangoV1	proxy:0x38F7Aa5370439E879370E24AdD063a11Bd74610D,implementation:0xf3404bbc05003192349dc02fd27287d6ddce13d8
137	cbridge	0x103ea2aBF8c85f2bb77a80Db3F425B1B00ba8Bc0
137	multichain	0xbdff5a098323FeDECF9Bf937E4536A5A804e7Ae6
1285	rangoV1	proxy:0x0e3EB2eAB0e524b69C79E24910f4318dB46bAa9c,implementation:0x8d6ca235fd71fff6df231b6a7823a90a7d40971f
1285	cbridge	0xDAf5A411eAd333A452330092396004d7e5b75aF6
1285	multichain	0xf3404bBC05003192349DC02FD27287d6DDcE13D8
43114	rangoV1	proxy:0xc4300bE7878F42B39Bdfb6A57D0f88eB87b842C3,implementation:0xe569d45803c76a6651f466f6b16da4c8705d4257
43114	cbridge	0xe0b244D909E3E595002ff067c41Be4239586C97F
43114	multichain	0xefaA4e4dF9E4eC3cB3812Ac74120DD0EF5FFf6b4
250	rangoV1	proxy:0x2a7813412b8da8d18Ce56FE763B9eb264D8e28a8,implementation:0x3f603d09af1a55a5aa1a94e52f63f392d5423cb1
250	cbridge	0x66aDa3270d0B5C47Ba0eEa9EEfe4D18017bFB2A9
250	multichain	0x59c984B6234BE41F8a7797f31fC3a9d46c5F1B89
42161	rangoV1	proxy:0x0e3EB2eAB0e524b69C79E24910f4318dB46bAa9c,implementation:0xefaa4e4df9e4ec3cb3812ac74120dd0ef5fff6b4
42161	cbridge	0x2B2E0eA5e0B698252Ae3C740b6E73415223Fb5AF
42161	multichain	0xe3FCd45676c626c39F1C6201Ad32cdFa3dD9293B

chain ID	contract name	contract address
10	rangoV1	proxy:0x0e3EB2eAB0e524b69C79E24910f4318dB46bAa9c,implementation:0x0D71D18126E03646eb09FEc929e2ae87b7CAE69d
10	cbridge	0x688A62D79aAb9628783383b9faBB46C2Dc6c2FAf
10	multichain	0x72aA25d1c70ab1423a6d65BF2Cc6E4270a6E0224
1	rangoV1	proxy:0x0e3EB2eAB0e524b69C79E24910f4318dB46bAa9c,implementation:0x8d6ca235fd71fff6df231b6a7823a90a7d40971f
1	cbridge	0x7F913c6E98bd3119759788Bd2E7cc377FCB908a0
1	multichain	0xB5e9c880cd27E947CbD4EC2de78B043E34a3543d
1	thorchain	0xfB55A962355748a15DD07C3F30E6a257241b9E68
1	RangoThorchainOutputAggUniV2	0x4A7Ed08b2715a575eb719c756f8FA28541458b18,0xff2818c7Da1bF08F3C5ca404Abf09c5CD252FdAf
1	RangoThorchainOutputAggUniV3	0xa45AeFB7f8943693B37267a99FEd88D732Dccb8a

4.2 Visibility Description

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

MessageBusAddress			
Function Name	Visibility	Mutability	Modifiers
setMessageBus	Public	Can Modify State	onlyOwner

MessageReceiverApp			
Function Name	Visibility	Mutability	Modifiers
executeMessageWithTransfer	External	Payable	onlyMessageBus

MessageReceiverApp			
executeMessageWithTransferFallback	External	Payable	onlyMessageBus
executeMessageWithTransferRefund	External	Payable	onlyMessageBus
executeMessage	External	Payable	onlyMessageBus

MessageSenderApp			
Function Name	Visibility	Mutability	Modifiers
sendMessage	Internal	Can Modify State	-
sendMessageWithTransfer	Internal	Can Modify State	-
sendTokenTransfer	Internal	Can Modify State	-

MessageSenderLib			
Function Name	Visibility	Mutability	Modifiers
sendMessage	Internal	Can Modify State	-
sendMessageWithTransfer	Internal	Can Modify State	-
sendMessageWithLiquidityBridgeTransfer	Internal	Can Modify State	-
sendMessageWithPegVaultDeposit	Internal	Can Modify State	-
sendMessageWithPegBridgeBurn	Internal	Can Modify State	-

RangoCBridge			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

RangoCBridge			
<Receive Ether>	External	Payable	-
updateCBridgeAddress	External	Can Modify State	onlyOwner
updateCBridgeMessageBusSenderAddress	External	Can Modify State	onlyOwner
computeCBridgeSgnFee	External	-	-
send	External	Can Modify State	whenNotPaused nonReentrant
cBridgeIM	External	Payable	whenNotPaused nonReentrant
executeMessageWithTransferRefund	External	Payable	onlyMessageBus
executeMessageWithTransfer	External	Payable	onlyMessageBus whenNotPaused nonReentrant
executeMessageWithTransferFallback	External	Payable	onlyMessageBus whenNotPaused nonReentrant
_computeSwapRequestId	Private	-	-
_trySwap	Private	Can Modify State	-

RangoMultichain			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
addMultichainRouters	External	Can Modify State	onlyOwner

RangoMultichain			
removeMultichainRouters	External	Can Modify State	onlyOwner
multichainBridge	External	Payable	whenNotPaused nonReentrant

RangoThorchain			
Function Name	Visibility	Mutability	Modifiers
<Receive Ether>	External	Payable	-
swapInToThorchain	External	Payable	whenNotPaused nonReentrant

RangoThorchainOutputAggUniV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
swapIn	Public	Can Modify State	nonReentrant
swapOut	Public	Payable	nonReentrant

RangoThorchainOutputAggUniV3			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
swapIn	Public	Can Modify State	nonReentrant
swapOut	Public	Payable	nonReentrant

BaseContract			
Function Name	Visibility	Mutability	Modifiers

BaseContract			
addWhitelist	External	Can Modify State	onlyOwner
removeWhitelist	External	Can Modify State	onlyOwner
refund	External	Can Modify State	onlyOwner
refundNative	External	Can Modify State	onlyOwner
approve	Internal	Can Modify State	-
_sendToken	Internal	Can Modify State	-
_sendNative	Internal	Can Modify State	-
getBaseContractStorage	Internal	-	-
_getRevertMsg	Internal	-	-

BaseProxyContract			
Function Name	Visibility	Mutability	Modifiers
addWhitelist	External	Can Modify State	onlyOwner
removeWhitelist	External	Can Modify State	onlyOwner
updateFeeContractAddress	External	Can Modify State	onlyOwner
refund	External	Can Modify State	onlyOwner
refundNative	External	Can Modify State	onlyOwner
onChainSwaps	External	Payable	whenNotPaused nonReentrant
onChainSwapsInternal	Internal	Can Modify State	-
callSwapsAndFees	Private	Can Modify State	-

BaseProxyContract			
approve	Internal	Can Modify State	-
_sendToken	Internal	Can Modify State	-
_sendNative	Internal	Can Modify State	-
getBaseProxyContractStorage	Internal	-	-

RangoCBridgeProxy			
Function Name	Visibility	Mutability	Modifiers
updateRangoCBridgeAddress	External	Can Modify State	onlyOwner
cBridgeSend	External	Payable	whenNotPaused nonReentrant
cBridgeIM	External	Payable	whenNotPaused nonReentrant
getCBridgeProxyStorage	Internal	-	-

RangoMultichainProxy			
Function Name	Visibility	Mutability	Modifiers
updateRangoMultichainAddress	External	Can Modify State	onlyOwner
multichainBridge	External	Payable	whenNotPaused nonReentrant
getMultichainProxyStorage	Internal	-	-

RangoThorchainProxy			
Function Name	Visibility	Mutability	Modifiers
updateRangoThorchainAddress	External	Can Modify State	onlyOwner

RangoThorchainProxy			
swapInToThorchain	External	Payable	whenNotPaused nonReentrant
getThorchainProxyStorage	Internal	-	-

RangoV1			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

Owner can modify the cBridgeAddress address, and there is no event record. If the Owner's private key is stolen, an attacker can modify cBridgeAddress into a malicious contract. Malicious cBridgeAddress contract address affects user funds.

- contracts/bridges/cbridge/RangoCBridge.sol#L41-L43

```
    function updateCBridgeAddress(address _address) external onlyOwner {
        cBridgeAddress = _address;
    }
```

Owner can modify the messageBus address, Malicious messageBus contract address affects user funds.

- contracts/bridges/cbridge/RangoCBridge.sol#L45-L47

```
    function updateCBridgeMessageBusSenderAddress(address _address) external
onlyOwner {
        setMessageBus(_address);
    }
```

- contracts/bridges/cbridge/im/message/framework/MessageBusAddress.sol#L12-L15

```
    function setMessageBus(address _messageBus) public onlyOwner {
        messageBus = _messageBus;
        emit MessageBusUpdated(messageBus);
    }
```

Owner can add and remove the whitelist update fee contract address, and there is no event record.

- contracts/libs/BaseProxyContract.sol#L47-L61

```
    function addWhitelist(address _factory) external onlyOwner {
        BaseProxyStorage storage baseProxyStorage = getBaseProxyContractStorage();
        baseProxyStorage.whitelistContracts[_factory] = true;
    }

    function removeWhitelist(address _factory) external onlyOwner {
        BaseProxyStorage storage baseProxyStorage = getBaseProxyContractStorage();
        require(baseProxyStorage.whitelistContracts[_factory], 'Factory not found');
        delete baseProxyStorage.whitelistContracts[_factory];
    }

    function updateFeeContractAddress(address payable _address) external onlyOwner {
        BaseProxyStorage storage baseProxyStorage = getBaseProxyContractStorage();
        baseProxyStorage.feeContractAddress = _address;
    }
```

- contracts/libs/BaseContract.sol#L42-L60

```
    function addWhitelist(address _factory, bool isMessagingDApp) external onlyOwner
{
        BaseContractStorage storage baseStorage = getBaseContractStorage();
        if (isMessagingDApp)
            baseStorage.whitelistMessagingContracts[_factory] = true;
        else
            baseStorage.whitelistContracts[_factory] = true;
    }

    function removeWhitelist(address _factory, bool isMessagingDApp) external
onlyOwner {
        BaseContractStorage storage baseStorage = getBaseContractStorage();

        if (isMessagingDApp) {
            require(baseStorage.whitelistMessagingContracts[_factory], 'Factory not
found');
            delete baseStorage.whitelistMessagingContracts[_factory];
        } else {
            require(baseStorage.whitelistContracts[_factory], 'Factory not found');
            delete baseStorage.whitelistContracts[_factory];
        }
    }
```

- contracts/rango/bridges/cbridge/RangoCBridgeProxy.sol#L16-L19

```
    function updateRangoCBridgeAddress(address _address) external onlyOwner {
        CBridgeProxyStorage storage cbridgeProxyStorage = getCBridgeProxyStorage();
        cbridgeProxyStorage.rangoCBridgeAddress = _address;
    }
```

The Rango project adopts an upgradeable model, so the ProxyAdmin contract and the TransparentUpgradeableProxy contract have excessive authority issue too.

Solution

It is recommended to transfer the ownership to a governance contract or a timelock contract, at least a multi-signature contract. and add event records to facilitate review by community users.

It is recommended to transfer the authority of admin to the governance contract or timelock contract.

Status

Confirmed;

Fix Status: An incomplete fix has been made by adding event logging;

The project team response: We will surely update our security model to multi-sig or governance in midterm, and as we are still integrating more than 10 new bridges in coming weeks, we'll improve the security after our next audit for the new codes which is going to happen very soon. For now, we prefer to stick with the current model, since the security issue level is evaluated as "Low" based on the audit report.

[N2] [Suggestion] Redundant code

Category: Others

Content

The updateCBridgeMessageBusSenderAddres function has the same effect as the setMessageBus function, and only the Owner can call it.

- contracts/bridges/cbridge/RangoCBridge.sol#L45-L47

```
    function updateCBridgeMessageBusSenderAddress(address _address) external
onlyOwner {
        setMessageBus(_address);
    }
```

- contracts/bridges/cbridge/im/message/framework/MessageBusAddress.sol#L12-L15

```
    function setMessageBus(address _messageBus) public onlyOwner {
        messageBus = _messageBus;
        emit MessageBusUpdated(messageBus);
    }
```

Solution

It is recommended to check business logic and remove redundant code.

Status

Fixed

[N3] [Suggestion] Compatibility reminder

Category: Others

Content

Contracts are not compatible with deflationary tokens.When deflationary tokens are transferred, the amount received by the contract may be less than request.amountIn.

- contracts/libs/BaseProxyContract.sol#L89-L114

```
    function onChainSwapsInternal(SwapRequest memory request, Call[] calldata calls)
internal returns (bytes[] memory, uint) {

        IERC20 ercToken = IERC20(request.toToken);
        uint balanceBefore = request.toToken == NULL_ADDRESS
            ? address(this).balance
            : ercToken.balanceOf(address(this));

        bytes[] memory result = callSwapsAndFees(request, calls);

        uint balanceAfter = request.toToken == NULL_ADDRESS
            ? address(this).balance
            : ercToken.balanceOf(address(this));

        uint secondaryBalance;
        if (calls.length > 0) {
            require(balanceAfter - balanceBefore > 0, "No balance found after
swaps");

            secondaryBalance = balanceAfter - balanceBefore;
            emit DexOutput(request.toToken, secondaryBalance);
        } else {
            secondaryBalance = balanceAfter > balanceBefore ? balanceAfter -
balanceBefore : request.amountIn;
        }

        return (result, secondaryBalance);
    }
```

- contracts/libs/BaseProxyContract.sol#L189-L217

```
function callSwapsAndFees(SwapRequest memory request, Call[] calldata calls) private
returns (bytes[] memory) {
        bool isSourceNative = request.fromToken == NULL_ADDRESS;
        BaseProxyStorage storage baseProxyStorage = getBaseProxyContractStorage();

        // validate
        require(baseProxyStorage.feeContractAddress != NULL_ADDRESS, "Fee contract
address not set");

        for(uint256 i = 0; i < calls.length; i++) {
            require(baseProxyStorage.whitelistContracts[calls[i].target], "Contact
not whitelisted");
        }

        // Get all the money from user
        uint totalInputAmount = request.feeIn + request.affiliateIn +
request.amountIn;
        if (isSourceNative)
            require(msg.value >= totalInputAmount, "Not enough ETH provided to
contract");

        // Check max fee/affiliate is respected
        uint maxFee = totalInputAmount * MAX_FEE_PERCENT_x_10000 / 10000;
        uint maxAffiliate = totalInputAmount * MAX_AFFILIATE_PERCENT_x_10000 / 10000;
        require(request.feeIn <= maxFee, 'Requested fee exceeded max threshold');
        require(request.affiliateIn <= maxAffiliate, 'Requested affiliate reward
exceeded max threshold');

        // Transfer from wallet to contract
        if (!isSourceNative) {
            for(uint256 i = 0; i < calls.length; i++) {
                approve(request.fromToken, calls[i].target, totalInputAmount);
            }
            SafeERC20.safeTransferFrom(IERC20(request.fromToken), msg.sender,
address(this), totalInputAmount);
        }
```

Solution

It is recommended to add a reminder on the webpage to avoid users interacting with deflationary tokens. record the balance of the contract before the transfer, execute the transfer, and then check the balance after the transfer, make sure afterBalance - beforeBalance = amount, amount is the parameter input to the transfer function.

Status

Fixed

[N4] [High] Business logic issue

Category: Design Logic Audit

Content

The Owner can withdraw the assets retained in the contract through the refund and refundNative functions.The onChainSwapsInternal function does not check the balanceBefore and balanceAfter values of request.fromToken in the BaseProxyContract contract, so the attacker can use this issue to transfer the assets retained in the contract. so it can bypass onlyOwner restrictions.

- contracts/libs/BaseProxyContract.sol#L78-L87

```
    function onChainSwaps(
        SwapRequest memory request,
        Call[] calldata calls,
        bool nativeOut
    ) external payable whenNotPaused nonReentrant returns (bytes[] memory) {
        (bytes[] memory result, uint outputAmount) = onChainSwapsInternal(request,
calls);

        _sendToken(request.toToken, outputAmount, msg.sender, nativeOut, false);
        return result;
    }
```

- contracts/libs/BaseProxyContract.sol#L89-L113

```
    function onChainSwapsInternal(SwapRequest memory request, Call[] calldata calls)
internal returns (bytes[] memory, uint) {

        IERC20 ercToken = IERC20(request.toToken);
        uint balanceBefore = request.toToken == NULL_ADDRESS
            ? address(this).balance
            : ercToken.balanceOf(address(this));
        bytes[] memory result = callSwapsAndFees(request, calls);

        uint balanceAfter = request.toToken == NULL_ADDRESS
            ? address(this).balance
            : ercToken.balanceOf(address(this));

        uint secondaryBalance;
        if (calls.length > 0) {
            require(balanceAfter - balanceBefore > 0, "No balance found after 
swaps");

            secondaryBalance = balanceAfter - balanceBefore;
            emit DexOutput(request.toToken, secondaryBalance);
        } else {
            secondaryBalance = balanceAfter > balanceBefore ? balanceAfter - 
balanceBefore : request.amountIn;
        }

        return (result, secondaryBalance);
    }
```

Transfer out the assets in the contract through calls[i].target.call.

- contracts/libs/BaseProxyContract.sol#L115-L170

```
    function callSwapsAndFees(SwapRequest memory request, Call[] calldata calls) 
private returns (bytes[] memory) {
        bool isSourceNative = request.fromToken == NULL_ADDRESS;
        BaseProxyStorage storage baseProxyStorage = getBaseProxyContractStorage();

        // validate
        require(baseProxyStorage.feeContractAddress != NULL_ADDRESS, "Fee contract 
address not set");

        for(uint256 i = 0; i < calls.length; i++) {
            require(baseProxyStorage.whitelistContracts[calls[i].target], "Contact 
not whitelisted");
        }

        // Get all the money from user
        uint totalInputAmount = request.feeIn + request.affiliateIn +
request.amountIn;
        if (isSourceNative)
            require(msg.value >= totalInputAmount, "Not enough ETH provided to
contract");

        // Check max fee/affiliate is respected
        uint maxFee = totalInputAmount * MAX_FEE_PERCENT_x_10000 / 10000;
        uint maxAffiliate = totalInputAmount * MAX_AFFILIATE_PERCENT_x_10000 / 10000;
        require(request.feeIn <= maxFee, 'Requested fee exceeded max threshold');
        require(request.affiliateIn <= maxAffiliate, 'Requested affiliate reward
exceeded max threshold');

        // Transfer from wallet to contract
        if (!isSourceNative) {
            for(uint256 i = 0; i < calls.length; i++) {
                approve(request.fromToken, calls[i].target, totalInputAmount);
            }
            SafeERC20.safeTransferFrom(IERC20(request.fromToken), msg.sender,
address(this), totalInputAmount);
        }

        // Get Platform fee
        if (request.feeIn > 0) {
            _sendToken(request.fromToken, request.feeIn,
baseProxyStorage.feeContractAddress, isSourceNative, false);
            emit FeeReward(request.fromToken, baseProxyStorage.feeContractAddress,
request.feeIn);
        }

        // Get affiliator fee
        if (request.affiliateIn > 0) {
            require(request.affiliatorAddress != NULL_ADDRESS, "Invalid
affiliatorAddress");
            _sendToken(request.fromToken, request.affiliateIn,
request.affiliatorAddress, isSourceNative, false);
            emit AffiliateReward(request.fromToken, request.affiliatorAddress,
request.affiliateIn);
        }

        bytes[] memory returnData = new bytes[](calls.length);
        for (uint256 i = 0; i < calls.length; i++) {
            (bool success, bytes memory ret) = isSourceNative
                ? calls[i].target.call{value: request.amountIn}(calls[i].callData)
                : calls[i].target.call(calls[i].callData);

            emit CallResult(calls[i].target, success, ret);
            require(success, string(abi.encodePacked("Call failed, index:", i)));
            returnData[i] = ret;
        }

        return returnData;
    }
```

Solution

It is recommended to add a check for the balance before and after request.fromToken.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206290002	SlowMist Security Team	2022.06.20 - 2022.06.29	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 high risk, 1 low risk, 2 suggestion vulnerabilities. And 1 low risk vulnerability was confirmed; All other findings were fixed. The code was deployed.

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.