Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2024.07.24, the SlowMist security team received the Cicada Protocol team's security audit application for rMner, 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
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

Serial Number	Audit Class	Audit Subclass
7	Security Design 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

The project contains two token contracts, an exchange contract for exchanging two tokens and a transaction contract for automatic trading with the pool contract.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing zero address check	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N2	Redundant code	Others	Suggestion	Acknowledged
N3	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N4	Variable type not set correctly	Others	Suggestion	Fixed
N5	Unchecked return value	Others	Low	Fixed
N6	Contract interface import risks	Others	Suggestion	Fixed
N7	The _rate parameter is not checked	Design Logic Audit	Low	Fixed
N8	Event logging error	Design Logic Audit	Low	Fixed
N9	Redundant code	Others	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/mineral-devlop/rMner

Initial audit version: 0473bef932213ce2319c4bb716fc0464e5c4dd34

Final audit version: 9d57ded923255c97d2023d7c973f51e9fd91d661

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:

rMner			
Function Name	Visibility	Mutability	Modifiers

rMner			
<Constructor>	Public	Can Modify State	Ownable
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-
mint	External	Can Modify State	onlyOwner
burn	External	Can Modify State	onlyOwner
transferFrom	Public	Can Modify State	-
increaseAllowance	Public	Can Modify State	-
decreaseAllowance	Public	Can Modify State	-
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_spendAllowance	Internal	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-
_afterTokenTransfer	Internal	Can Modify State	-

r2MNER			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
mintTo	Public	Can Modify State	onlyExchangePolicy
setMonetaryPolicy	External	Can Modify State	onlyOwner
setExchangePolicy	External	Can Modify State	onlyOwner
rebase	Public	Can Modify State	onlyMonetaryPolicy
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
burn	Public	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
increaseAllowance	Public	Can Modify State	-
decreaseAllowance	Public	Can Modify State	-
getTotalShares	External	-	-
sharesOf	External	-	-
getSharesByR2Mner	Public	-	-
getrMnerShares	Public	-	-

r2MNER			
transferShares	External	Can Modify State	-
transferSharesFrom	External	Can Modify State	-
_transfer	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_spendAllowance	Internal	Can Modify State	-
_getTotalShares	Internal	-	-
_sharesOf	Internal	-	-
_transferShares	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_mintShares	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_burnShares	Internal	Can Modify State	-
_emitTransferEvents	Internal	Can Modify State	-
_emitTransferAfterMintingShares	Internal	Can Modify State	-
_mintInitialShares	Internal	Can Modify State	-

rMnerBlack			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	initializer
isRMnerBlack	Public	-	-
isR2MnerBlack	Public	-	-
isR2MnerRebaseBlack	Public	-	-
setRMnerBlack	Public	Can Modify State	onlyOwner

rMnerBlack			
setR2MnerBlack	Public	Can Modify State	onlyOwner
setR2MnerRebaseBlack	Public	Can Modify State	onlyOwner

rMnerExchange			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
swap	Public	Payable	nonReentrant
_takeBtcFee	Internal	Can Modify State	-
_getRMnerFee	Internal	-	-
withdrawTokensSelf	External	Can Modify State	onlyAssetManager
setBtcFee	External	Can Modify State	onlyAdmin
setFee	External	Can Modify State	onlyAdmin
setSwapRate	External	Can Modify State	onlyAdmin
setStopSwap	External	Can Modify State	onlyOwner
setPriceAddress	External	Can Modify State	onlyOwner
setFeeReceive	External	Can Modify State	onlyOwner
setAdmin	External	Can Modify State	onlyOwner
setAssetManager	External	Can Modify State	onlyOwner
<Receive Ether>	External	Payable	-

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

rMnerPrice			
getPrice	External	-	-
getAmountsOut	Public	-	-

rMnerRebase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
buyrMner	Public	Payable	onlyOwner
withdrawTokensSelf	External	Can Modify State	onlyOwner
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing zero address check

Category: Others

Content

1.In the rMner contract, the zero address check of the `_blackContract` parameter address is not performed in the constructor function.

- rMner.sol#L24-L34

```
constructor
```

2.In the rMnerBlack contract, the `setRMnerBlack` function, `setR2MnerBlack` function, and `setR2MnerRebaseBlack` function did not perform a zero address check on the `user` parameter address.

- rMnerBlack.sol#L31-L34, L36-L39, L41-L44

```
function setRMnerBlack
function setR2MnerBlack
```

```
function setR2MnerRebaseBlack
```

3.In the r2MNER contract, the `constructor` function lacks a zero address check for the `_blackContract` parameter address, the `setMonetaryPolicy` function lacks a zero address check for the `monetaryPolicy_` parameter address, and the `setExchangePolicy` function lacks a zero address check for the `exchangePolicy_` parameter address.

- r2Mner.sol#L72-L82, L89-L92, L95-L98

```
constructor
function setMonetaryPolicy
function setExchangePolicy
```

4.In the rMnerRebase contract, the `constructor` function lacks a zero address check for the `_r2MNER` parameter address and the `_exchangeAddress` parameter address, and the `withdrawTokensSelf` function lacks a zero address check for the `to` parameter address.

- rMnerRebase.sol#L18-L22, L45-L58

```
constructor
function withdrawTokensSelf
```

5.In the rMnerExchange contract, the `constructor` function lacks a zero address check for the `_rMNER` parameter, `_r2MNER` parameter, and `_price` parameter addresses; the `setPriceAddress` function lacks a zero address check for the `rMnerPrice_` parameter address;the `withdrawTokensSelf` function lacks a zero address check for the `to` parameter address; and the `setFeeReceive` function lacks a zero address check for the `feeReceive_` parameter address.

- rMnerExchange.sol#L48-L56, L137-L150, L176-L180, L182-L186

```
constructor
function withdrawTokensSelf
function setPriceAddress
function setFeeReceive
```

Solution

It is recommended to add a zero address check to the address parameter in the function.

Status

Fixed

[N2] [Suggestion] Redundant code

Category: Others

Content

1.In the r2MNER contract, rMnerExchange contract, rMnerPrice contract, and rMnerRebase contract, the `SafeMath` library is imported to check the overflow risk during the calculation process. However, the solidity version 0.8.7 in the contract can automatically check the overflow risk, so there is no need to use the `SafeMath` library.

- r2Mner.sol#L5, L12

```
import "./utils/SafeMath.sol";
using SafeMath for uint256;
```

- rMnerExchange.sol#L4, L18

```
import "./utils/SafeMath.sol";
using SafeMath for uint256;
```

- rMnerPrice.sol#L4, L7

```
import "./utils/SafeMath.sol";
using SafeMath for uint256;
```

- rMnerRebase.sol#L7, L10

```
import "./utils/SafeMath.sol";
using SafeMath for uint256;
```

2.In the r2MNER contract, the `_burnShares` function is an internal function and is not called by other functions.

- r2Mner.sol#L370-L394

```
    function _burnShares(address _account, uint256 _sharesAmount)
        internal
        returns (uint256 newTotalShares)
    {
    ```
    }
```

3.In the rMnerBlack contract, the `isR2MnerRebaseBlack` function is not used.

- rMnerBlack.sol#L27-L29

```
    function isR2MnerRebaseBlack(address user) public view returns (bool) {
        return r2MnerRebaseBlacks[user];
    }
```

Solution

It is recommended to delete the redundant code.

Status

Acknowledged

[N3] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the r2MNER contract, the `Owner` role can modify important parameters in the contract.

- r2Mner.sol#L89-L92, L95-L98

```
    function setMonetaryPolicy
    function setExchangePolicy
```

2.In the r2MNER contract, the `ExchangePolicy` role and the `Owner` role can mint tokens arbitrarily through the `mintTo` function.

- r2Mner.sol#L84-L86

```
    function mintTo
```
```

3.In the r2MNER contract, the `MonetaryPolicy` role and the `Owner` role can rebase tokens through the `rebase` function.

- r2Mner.sol#L100-L118

```
    function rebase
```

4.In the rMner contract, the `Owner` role can mint and burn rMner tokens arbitrarily through the `mint` function and `burn` function.

- rMner.sol#L94-L97, L98-L100, L172-L185, L199-L215

```
    function mint
    function burn
    function _mint
    function _burn
```

5.the rMnerBlack contract, the `Owner` role can set the blacklist address through the `setRMnerBlack` function, `setR2MnerBlack` function, and `setR2MnerRebaseBlack` function.

- rMnerBlack.sol#L31-L34, L36-L39, -L41-L44

```
    function setRMnerBlack
    function setR2MnerBlack
    function setR2MnerRebaseBlack
```

6.In the rMnerRebase contract, the `Owner` role can purchase rMner tokens through the `buyrMner` function, transfer them to `exchangeAddress`, and rebase r2MNER tokens.

- rMnerRebase.sol#L24-L41

```
    function buyrMner
```

7.In the rMnerRebase contract, the `Owner` role can transfer out any ERC20 tokens and native tokens in the contract through the `withdrawTokensSelf` function.

- rMnerRebase.sol#L45-L58

```
function withdrawTokensSelf
```

8.In the rMnerExchange contract, the `Owner` role and the `admin` role can modify important parameters in the contract.

- rMnerExchange.sol#L152-L156, L158-L162, L164-L168, L170-L174, L176-L180, L182-L186, L217-L222, L224-L229

```
function setBtcFee
function setFee
function setStopSwap
function setSwapRate
function setPriceAddress
function setFeeReceive
function setAdmin
function setAssetManager
```

9.In the rMnerExchange contract, the `assetManager` role can transfer out any ERC20 tokens and native tokens in the contract through the `withdrawTokensSelf` function.

- rMnerExchange.sol#L137-L150

```
function withdrawTokensSelf
```

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. 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. When updating a new contract, be careful to maintain compatibility with the old contract in terms of storage structure, do not reorder the state variables in the old contract, and do not insert new variables between the old ones.

Status

Acknowledged

[N4] [Suggestion] Variable type not set correctly

Category: Others

Content

1.In the r2MNER contract, the `blackContract` variable should be immutable and the `_decimals` variable should be a constant.

- r2Mner.sol#L25, L36

```
uint8 _decimals = 18;
address blackContract;
```

2.In the rMner contract, the `blackContract` variable should be immutable.

- r2Mner.sol#L22

```
address blackContract;
```

3.In the rMnerRebase contract, the `r2MNER` variable and the `exchangeAddress` variable should be immutable.

- rMnerRebase.sol#L15, L16

```
address r2MNER;
address exchangeAddress;
```

4.In the rMnerExchange contract, the `rMNER` variable and the `r2MNER` variable should be immutable.

- rMnerExchange.sol#L23, L24

```
address public rMNER;
address public r2MNER;
```

Solution

It is recommended to set the type of contract variables correctly.

Status

Fixed

[N5] [Low] Unchecked return value

Category: Others

Content

1.In the rMnerRebase contract, the `withdrawTokensSelf` function did not check the return value when calling the `transfer` function of the `token` contract.

- rMnerRebase.sol#L45-L58

```
    function withdrawTokensSelf(address token, address to) external onlyOwner {
        ```
         IERC20(token).transfer(to, bal);
        ```
    }
```

2.n the rMnerExchange contract, the `withdrawTokensSelf` function did not check the return value when calling the `transfer` function of the `token` contract.

- rMnerExchange.sol#L137-L150

```
    function withdrawTokensSelf(address token, address to) external onlyOwner {
        ```
         IERC20(token).transfer(to, bal);
        ```
    }
```

Solution

It is recommended to check the return value of the transfer function or use the safeTransfer function.

Status

Fixed

[N6] [Suggestion] Contract interface import risks

Category: Others

Content

1.In the rMnerRebase contract, the `ISwap` contract interface is directly imported from the github repository, which may lead to security risks.

- rMnerRebase.sol#L3

```
import "https://github.com/izumiFinance/iZiSwap-
periphery/blob/main/contracts/interfaces/ISwap.sol";
```

2.In the rMnerPrice contract, the `IiZiSwapPool` contract interface is directly imported from the github repository, which may lead to security risks.

- rMnerPrice.sol#L3

```
import "https://github.com/izumiFinance/iZiSwap-
core/blob/main/contracts/interfaces/IiZiSwapPool.sol";
```

Solution

It is recommended to import the contract interface from a local file.

Status

Fixed

[N7] [Low] The _rate parameter is not checked

Category: Design Logic Audit

Content

In the rMnerExchange contract, the `setSwapRate` function does not check that the value of the `_rate` parameter is not 0.

- rMnerExchange.sol#L170-L174

```
    function setSwapRate(uint256 _rate) external onlyOwner {
        uint256 prev = rate;
        rate = _rate;
        emit UpdateRate(prev, _rate);
    }
```

Solution

It is recommended to check that the _rate parameter is not 0.

Status

Fixed

[N8] [Low] Event logging error

Category: Design Logic Audit

Content

In the `swap` function of the rMnerExchange contract, when `tokenA` is the r2MNER token, the third parameter in the `Swap` event should be the rMNER token address, because `tokenB` is the rMNER token at this time.

- rMnerExchange.sol#L58-L99

```
    function swap(address tokenA, uint256 _amount) public payable nonReentrant {
        ```
        } else {
            uint256 _outAmount = _amount.div(rate).mul(10000);

            uint256 _feeAmount = _getRMnerFee(_outAmount, outFee);
            _takeBtcFee(_outAmount, btcOutFee);

            TransferHelper.safeTransferFrom(
                r2MNER,
                msg.sender,
                address(this),
                _amount
            );
            uint256 r2MnerBalance = IR2MNER(r2MNER).balanceOf(address(this));
            IR2MNER(r2MNER).burn(r2MnerBalance);

            if (_feeAmount > 0) {
                IERC20(rMNER).safeTransfer(feeReceive, _feeAmount);
            }
            IERC20(rMNER).safeTransfer(msg.sender, _outAmount - _feeAmount);

            emit Swap(msg.sender, tokenA, r2MNER, _amount, _outAmount);
        }
    }
```

**Solution**

It is recommended to modify the third parameter in the second Swap event to the rMNER token address.
```

Status

Fixed

[N9] [Suggestion] Redundant code

Category: Others

Content

In the rMner contract, the `_beforeTokenTransfer` function and the `_afterTokenTransfer` function do not implement specific logic.

- rMner.sol#L234-L238, L240-L244

```
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}

    function _afterTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
}
```

Solution

It is recommended to delete the redundant code.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002407250003	SlowMist Security Team	2024.07.24 - 2024.07.25	Medium 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, 3 low risk, 5 suggestion.

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