Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.11.05, the SlowMist security team received the Coin98 team's security audit application for Coin98 Vault, 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:

- 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:

Coin98Vault.sol (SHA256): 700b72d059920fd7fec56427256900e65cad5872da69c34ffd29dda09380b926

Fixed Version:

Coin98Vault.sol (SHA256): 8922f0317575d24668256ef07d832a08bb5423c6b7ecaff1dce624b761feaf62

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Set admin defect	Authority Control Vulnerability	Suggestion	Fixed

NO	Title	Category	Level	Status
N2	ScheduleData overwritten issue	Authority Control Vulnerability	Low	Fixed
N3	Update schedule data issue	Others	Suggestion	Confirmed
N4	Low-level external call issue	Others	Low	Fixed

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:

Coin98Vault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
admins	Public	-	-
recipients	Public	-	-
schedules	Public	-	-
setAdmins	Public	Can Modify State	onlyOwner

Coin98Vault			
withdraw	Public	Can Modify State	onlyAdmin
withdrawNft	Public	Can Modify State	onlyAdmin
schedule	Public	Can Modify State	onlyAdmin
redeem	Public	Payable	-

Coin98VaultFactory			
Function Name	Visibility	Mutability	Modifiers
fee	External	-	-
ownerReward	External	-	-
createVault	External	Can Modify State	-
setFee	Public	Can Modify State	onlyOwner
withdraw	Public	Can Modify State	onlyOwner
withdrawNft	Public	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Set admin defect

Category: Authority Control Vulnerability

Content

In the Coin98Vault contract, the owner role can modify the status of the admin role through the setAdmins function.

When adding a new admin, it is not checked whether the admin already exists, which will cause the problem of repeatedly adding the existing admin role.

Code location:

```
  function setAdmins(address[] memory nAdmins_, bool[] memory nStatuses_) public
onlyOwner {
    require(nAdmins_.length != 0, "C98Vault: Empty arguments");
    require(nStatuses_.length != 0, "C98Vault: Empty arguments");
    require(nAdmins_.length == nStatuses_.length, "C98Vault: Invalid arguments");

    uint256 i;
    for(i = 0; i < nAdmins_.length; i++) {
      address nAdmin = nAdmins_[i];
      if(nStatuses_[i]) {
        _admins.push(nAdmin);
        _adminStatuses[nAdmin] = nStatuses_[i];
        emit AdminAdded(nAdmin);
      } else {
        uint256 j;
        for(j = 0; j < _admins.length; j++) {
          if(_admins[j] == nAdmin) {
            _admins[j] = _admins[_admins.length - 1];
            _admins.pop();
            delete _adminStatuses[nAdmin];
            emit AdminRemoved(nAdmin);
            break;
          }
        }
      }
    }
  }
```

Solution

It is recommended to check whether this admin already exists when adding a new admin.

Status

Fixed

[N2] [Low] ScheduleData overwritten issue

Category: Authority Control Vulnerability

Content

In the Coin98Vault contract, the admin role can set vesting for the user through the schedule function. If the scheduleKey is the same as before, the current ScheduleData will overwrite the previous ScheduleData.

Code location:

```
  function schedule(address token_, uint256 timestamp_, address[] memory
nRecipients_, uint256[] memory nAmounts_) onlyAdmin public {
    require(nRecipients_.length != 0, "C98Vault: Empty arguments");
    require(nAmounts_.length != 0, "C98Vault: Empty arguments");
    require(nRecipients_.length == nAmounts_.length, "C98Vault: Invalid arguments");

    uint256 i;
    for(i = 0; i < nRecipients_.length; i++) {
      address nRecipient = nRecipients_[i];
      uint256 nAmount = nAmounts_[i];

      bool isRecipientExist = _schedules[nRecipient].length > 0;
      bytes32 scheduleKey = keccak256(abi.encodePacked(nRecipient, token_,
timestamp_));

      ScheduleData memory nSchedule;
      nSchedule.token = token_;
      nSchedule.timestamp = timestamp_;
      nSchedule.amount = nAmount;

      _scheduleDatas[scheduleKey] = nSchedule;
      emit ScheduleUpdated(scheduleKey, nRecipient, token_, timestamp_, nAmount);

      uint256 j;
      uint256 found = 0;
      for(j = 0; j < _schedules[nRecipient].length; j++) {
        if(_schedules[nRecipient][j] == scheduleKey) {
          found = 1;
          break;
        }
      }
```

```
        if(found == 0) {
          _schedules[nRecipient].push(scheduleKey);
        }
        if(!isRecipientExist) {
          _recipients.push(nRecipient);
          emit RecipientAdded(nRecipient);
        }
      }
    }
```

Solution

If the design is unexpected, if the same scheduleKey is set, it is recommended to merge the same ScheduleData.

Status

Fixed

[N3] [Suggestion] Update schedule data issue

Category: Others

Content

In the vault contract, the admin can update the scheduleData of the specified key through the updateSchedule function, but it does not check whether the current time is less than `scheduleData.timestamp` during the update. This may cause the current key to having reached the redeem condition, but the redeem is delayed due to the updateSchedule operation.

Code location:

```
  function updateSchedule(bytes32 key_, uint256 eventId_, uint256 timestamp_, address
receivingToken_, address sendingToken_,
    address recipient_, uint256 receivingTokenAmount_, uint256 sendingTokenAmount_)
onlyAdmin public {
    require(recipient_ != address(0), "C98Vault: Invalid recipient");

    ScheduleData storage scheduleData = _scheduleDatas[key_];
    require(scheduleData.recipient != address(0), "C98Vault: Invalid schedule data");

    scheduleData.eventId = eventId_;
```

```
      scheduleData.timestamp = timestamp_;
      scheduleData.recipient = recipient_;
      scheduleData.receivingToken = receivingToken_;
      scheduleData.receivingTokenAmount = receivingTokenAmount_;
      scheduleData.sendingToken = sendingToken_;
      scheduleData.sendingTokenAmount = sendingTokenAmount_;
      _scheduleDatas[key_] = scheduleData;

      emit ScheduleUpdated(key_, scheduleData);
    }
```

Solution

It is recommended to check whether the current time is less than `scheduleData.timestamp` when performing the updateSchedule operation.

Status

Confirmed

[N4] [Low] Low-level external call issue

Category: Others

Content

In the Coin98Vault contract, when the user claims tokens through the redeem function, if the passed token_ parameter is address(0), then the Coin98Vault contract will transfer native tokens to the target user through call without restricting gas.

Code location:

```
if(fee > 0) {
        uint256 reward = IVaultConfig(_factory).ownerReward();
        uint256 finalFee = fee - reward;
        (bool success, bytes memory data) = _factory.call{value:finalFee}("");
        require(success, "C98Vault: Unable to charge fee");
      }
      if(token_ == address(0)) {
        _msgSender().call{value:totalAmount}("");
      } else {
        IERC20(token_).transfer(_msgSender(), totalAmount);
    }
```

Solution

It is recommended to limit gas usage during low-level calls, or use transfer for native token transfer.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002111090001	SlowMist Security Team	2021.11.05 - 2021.11.09	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 2 low risk, 2 suggestion vulnerabilities. And 1 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.