Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.09.07, the SlowMist security team received the Memeland team's security audit application for Memecoin, 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

It's a contract to sell tokens and claim tokens.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Preemptive Initialization	Race Conditions Vulnerability	Suggestion	Fixed
N2	Values not adjusted after refund	Design Logic Audit	Low	Acknowledged

NO	Title	Category	Level	Status
N3	Ensure list is sorted to prevent calculation failures	Design Logic Audit	Suggestion	Fixed
N4	Missing event record	Malicious Event Log Audit	Suggestion	Fixed
N5	Risk of over-privilege	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/9gag/memecoin-contract-audit

Audited commit:de9505723fcdf957f67eb11c207300c1a6b32796

Review commit:d496309fb5d2a4c3b1efc5ac7c847467e31799d6

The main network address of the contract is as follows:

MemecoinFiresaleV1

0xB9879cD06c904c2FDbc75d03534929b5E842F3a0

Implementation contract

0x8726e455769d5c58c4ad8453c0c4b7ae116f3113

MemecoinClaim

0xE6f3494E839F3D3Fb36c407eB35cd85D90Dc3704

Implementation contract

0x6fdF393AeC35Cac8125c2E022A0Ad05cCEBBa19B

MemecoinClaim

0xb1911D8FFcC2d8cA6c5EA4F4f18bE6ea675c1Ce7

Implementation contract

0xDC79D2c13Ec218049405836A74454952902eC42d

MemecoinClaim

0x517daba2695244ace417758f72d0Dfb8EfA0Ad59

Implementation contract

0x81804B4FEE3F22DB3cB38e714BFE5fb6E98ad316

MemecoinClaim

0x6f8F1266565d3A7DD05c30EBf64Faf509E4be61a

Implementation contract

0xDC9Aef21d9781B1C1fBf7B107715e23D500ef70C

4.2 Visibility Description

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

MemecoinClaim			
Function Name	Visibility	Mutability	Modifiers
_authorizeUpgrade	Internal	Can Modify State	onlyUpgrader
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
claim	External	Can Modify State	nonReentrant onlyValidClaimSetup
claimInNFTs	External	Can Modify State	nonReentrant onlyValidClaimSetup
claimFromMulti	External	Can Modify State	nonReentrant onlyValidClaimSetup onlyMultiClaim
claimInNFTsFromMulti	External	Can Modify State	nonReentrant onlyValidClaimSetup onlyMultiClaim
_getRequester	Private	-	-
_claim	Internal	Can Modify State	-

MemecoinClaim			
_claimInNFTs	Internal	Can Modify State	-
_executeClaim	Private	Can Modify State	-
_executeClaimInNFTs	Private	Can Modify State	-
_verifyNFTClaim	Private	-	onlyValidCollectionId
_verifyNFTRewardClaim	Private	-	onlyValidCollectionId
_calculateClaimable	Private	-	-
_calculateNFTClaimable	Private	-	-
_calculateNFTRewardsClaimable	Private	-	-
_calculateRemainClaimable	Private	-	-
_calculateUnlockedAmount	Private	-	-
_calculateUnlockedAmountByDaysElapsed	Private	-	-
_calculateNFTUnlockedAmount	Private	-	-
_toUint128	Private	-	-
setClaimables	External	Can Modify State	onlyOwner
setNFTClaimables	External	Can Modify State	onlyOwner
addNFTUnlockedBPAndSetUnlockTs	External	Can Modify State	onlyOwner
setUnclaimedNFTRewards	External	Can Modify State	onlyValidCollectionId onlyOwner
setRevealedCaptainzClaimable	External	Can Modify State	onlyOwner
depositClaimTokenAndStartClaim	External	Can Modify State	onlyOwner

MemecoinClaim			
withdrawClaimToken	External	Can Modify State	onlyOwner onlyClaimNotOpen
withdrawUnclaimedNFTRewards	External	Can Modify State	onlyOwner
setClaimSchedules	External	Can Modify State	onlyOwner onlyClaimNotOpen
setClaimActive	External	Can Modify State	onlyOwner
setClaimStartDate	External	Can Modify State	onlyOwner
setMultiClaimAddress	External	Can Modify State	onlyOwner
setUpgrader	External	Can Modify State	onlyOwner
getClaimInfo	Public	-	onlyValidClaimSetup
getClaimInfoByNFT	Public	-	onlyValidClaimSetup onlyValidCollectionId
getRewardsClaimInfoByNFT	Public	-	onlyValidClaimSetup onlyValidCollectionId
getTotalClaimableAmountsByNFTs	Public	-	-
getUserClaimDataByCollections	Public	-	-
getClaimSchedule	Public	-	-

MemecoinMultiClaim			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
multiClaim	External	Can Modify State	-
_getRequester	Private	-	-

MemecoinFiresaleV1			
Function Name	Visibility	Mutability	Modifiers
_authorizeUpgrade	Internal	Can Modify State	onlyUpgrader
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
reserve	External	Payable	nonReentrant onlyFiresaleOpening
refund	External	Can Modify State	nonReentrant onlyFiresaleFinished
_getRequester	Private	-	-
_checkValidity	Private	-	-
_withdraw	Private	Can Modify State	-
setFiresaleState	External	Can Modify State	onlyOwner
setRefundStartDate	External	Can Modify State	onlyOwner
setSigner	External	Can Modify State	onlyOwner
setUpgrader	External	Can Modify State	onlyOwner
withdrawSales	Public	Can Modify State	onlyOwner onlyFiresaleFinished
withdrawRefund	Public	Can Modify State	onlyOwner onlyFiresaleFinished
getFiresaleUsersCount	External	-	-
getFiresaleUsers	External	-	-

Memecoin			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit
permit	Public	Can Modify State	-
setTokenPool	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

By calling the initialize functions to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize.

- contracts/claim/MemecoinClaimV1.sol

```
function initialize(address _mvpAddress, address _captainzAddress, address
_potatozAddress) external initializer {
    ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained();
    OwnableUpgradeable.__Ownable_init_unchained();
    UUPSUpgradeable.__UUPSUpgradeable_init();
    upgrader = _msgSender();
    dc = IDelegationRegistry(0x00000000000076A84feF008CDAbe6409d2FE638B);
    setCollectionAddresses(_mvpAddress, _captainzAddress, _potatozAddress);
}
```

- contracts/firesale/MemecoinFiresaleV1.sol

```
function initialize(address _signer, uint256 _unitPrice) public initializer {
    ReentrancyGuardUpgradeable.__ReentrancyGuard_init();
    OwnableUpgradeable.__Ownable_init();
    signer = _signer;
    unitPrice = _unitPrice;
    upgrader = _msgSender();
    dc = IDelegationRegistry(0x00000000000076A84feF008CDAbe6409d2FE638B);
}
```

Solution

It is suggested that the initialize operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

Status

Fixed

[N2] [Low] Values not adjusted after refund

Category: Design Logic Audit

Content

The values of `usersTotalFiresales[requester]` and `firesaleTotal` are not reduced accordingly after the user is refunded, and the data is incorrect if the relevant `usersTotalFiresales[requester]` and `firesaleTotal` values are subsequently used.

- contracts/firesale/MemecoinFiresaleV1.sol

```
function refund(address _vault, uint32 _allocatedAmount, bytes calldata _signature)
    external
    nonReentrant
    onlyFiresaleFinished
{
    address requester = _getRequester(_vault);
    uint256 userTotalFiresale = usersTotalFiresales[requester];
    if (userTotalFiresale == 0) revert NoFiresaleRecord();
    // skip the sig validation & refundAvailable calculation if input is already >=
userTotalFiresale
    if (_allocatedAmount >= userTotalFiresale) revert NoRefundAvailable();
    if (usersRefunded[requester]) revert AlreadyRefunded();

    string memory action = string.concat("meme-firesale-refund-won_amount-",
Strings.toString(_allocatedAmount));
    if (!_checkValidity(requester, _signature, action)) revert InvalidSignature();

    usersRefunded[requester] = true;
    uint256 refundAvailable = (userTotalFiresale - _allocatedAmount) * unitPrice;
    _withdraw(requester, refundAvailable);

    emit UserRefunded(requester, _allocatedAmount);
}
```

Solution

Deduct the refunded portion of the amount.

Status

Acknowledged; Meets design expectations.

[N3] [Suggestion] Ensure list is sorted to prevent calculation failures

Category: Design Logic Audit

Content

Make sure the array in `ClaimSchedule.lockUpBPs` is from smallest to largest, otherwise `_calculateUnlockedAmountByDaysElapsed` will fail.

- contracts/claim/MemecoinClaimV1.sol

```
function setClaimSchedules(ClaimType[] calldata _claimTypes, ClaimSchedule[] calldata
_claimSchedules)
    external
    onlyOwner
    onlyClaimNotOpen
{
uint256 len = _claimSchedules.length;
if (_claimTypes.length != len) revert MismatchedArrays();
for (uint256 i; i < len;) {
    claimScheduleOf[_claimTypes[i]] = _claimSchedules[i];
    unchecked {
        ++i;
    }
}
}
```

Solution

Inside the logic, make sure that the values in the list are from smallest to largest, and that they are reasonable values.

Status

Fixed

[N4] [Suggestion] Missing event record

Category: Malicious Event Log Audit

Content

Key Parameter Settings Unrecorded Events .

- contracts/firesale/MemecoinFiresaleV1.sol

The following functions do not log events `setFiresaleState` , `setFiresalePriceInfo` , `setSigner` , `setUpgrader` , `withdrawSales` .

- contracts/claim/MemecoinClaimV1.sol

The following functions do not log events `setClaimables`, `setNFTClaimables` , `setNFTRewardsClaimables`, `setClaimActive`, `setClaimStartDate`, `setClaimTokenAddress`, `setCollectionAddresses`, `setMultiClaimAddress`, `setUpgrader`.

Solution

Recording events.

Status

Fixed

[N5] [Medium] Risk of over-privilege

Category: Authority Control Vulnerability Audit

Content

The owner can directly collect the token stored in the contract, if the owner's private key is leaked, it will result in the loss of the project's assets.

- contracts/claim/MemecoinMultiClaim.sol

```
    function withdrawClaimToken(uint256 _amount) external onlyOwner onlyClaimNotOpen
{
        address claimTokenAddress = address(claimToken);
        if (claimTokenAddress == address(0)) revert ClaimTokenZeroAddress();

        claimToken.safeTransfer(_msgSender(), _amount);
    }
```

The owner can withdraw the ETH sold through the contract by withdrawing the sales, which could result in a loss of project funds if the owner's private key is compromised.

- contracts/firesale/MemecoinFiresaleV1.sol

```
    function withdrawSales(uint256 _totalFiresaleItems) public onlyOwner
onlyFiresaleFinished {
        if (_totalFiresaleItems > firesaleTotal) revert WithdrawExceedTotalSales();

        uint256 sales = _totalFiresaleItems * unitPrice;
        uint256 available = sales - totalWithdrawnSales;
        if (available == 0) revert NoNewSales();

        totalWithdrawnSales += available;
        _withdraw(_msgSender(), available);
    }
```

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 EOA address can manage the authority involving emergency contract suspension. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged; MemecoinFiresaleV1

0xB9879cD06c904c2FDbc75d03534929b5E842F3a0

owner

0x21e14f503b03f43EBc4B779261D787183a54eC4b

MemecoinClaim

0xE6f3494E839F3D3Fb36c407eB35cd85D90Dc3704

owner

0x21e14f503b03f43EBc4B779261D787183a54eC4b

MemecoinClaim

0xb1911D8FFcC2d8cA6c5EA4F4f18bE6ea675c1Ce7

owner

0x536eCe8Ba00dc1c2c0cc4D477456ce8CB5CecbC3

MemecoinClaim

0x517daba2695244ace417758f72d0Dfb8EfA0Ad59

owner

0x6aC2f83E7a631F353062426e72b350a581837ceA

MemecoinClaim

0x6f8F1266565d3A7DD05c30EBf64Faf509E4be61a

owner

0xb0B89afD0EB04a4BCBa0630A72B40b60387935f5

A timelock will be added to manage the contract.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002309130001	SlowMist Security Team	2023.09.07 - 2023.09.13	Low Risk

Summary conclusion: 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, 1 low risk, 3 suggestion vulnerabilities.

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