Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.09.13, the SlowMist security team received the MultiDAO team's security audit application for MultiDAO, 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

Project address: https://github.com/MultichainDAO/SBT-contracts

Commit: feb66aa41a864073e111de3b03e2db46cb9a4383

Audit scope:

- MultiDAO-contracts/contracts/IDNFT.sol
- MultiDAO-contracts/contracts/MultiHonor.sol
- MultiDAO-contracts/contracts/VEPowerOracleSender.sol
- MultiDAO-contracts/contracts/VEPowerOracleReceiver.sol

Fixed Version

Project address: https://github.com/MultichainDAO/SBT-contracts

Commit: 5950338475ebb51f6f4479118bbb658681ed1837

Audit scope:

- MultiDAO-contracts/contracts/IDNFT.sol
- MultiDAO-contracts/contracts/MultiHonor.sol
- MultiDAO-contracts/contracts/VEPowerOracleSender.sol
- MultiDAO-contracts/contracts/VEPowerOracleReceiver.sol

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing event record	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N2	Design logic issue	Design Logic Audit	Suggestion	Fixed
N3	Arithmetic accuracy deviation vulnerability	Arithmetic Accuracy Deviation Vulnerability	Suggestion	Confirmed
N4	Possible spillover risk	Integer Overflow and Underflow Vulnerability	Suggestion	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:

IDNFT_v1			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setHonor	External	Can Modify State	onlyOwner
allowTransfer	External	Can Modify State	onlyOwner
_beforeTokenTransfer	Internal	Can Modify State	-
claim	External	Can Modify State	-

IDNFT_v1			
burn	External	Can Modify State	-
tokenURI	Public	-	-
_tokenURI	Internal	-	-
toString	Internal	-	-

MultiHonor_V1			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
__initRole	Internal	Can Modify State	-
__initSBT	Internal	Can Modify State	-
__initVEEpoch	Public	Can Modify State	-
setIDCard	External	Can Modify State	-
currentVEEpoch	Public	-	-
POC	External	-	-
POC	External	-	-
VEPower	External	-	-
VEPoint	External	-	-
EventPoint	External	-	-
levelRequire	Public	-	-
Level	External	-	-

MultiHonor_V1			
TotalPoint	External	-	-
setPOC	External	Can Modify State	-
addPOC	External	Can Modify State	-
setVEPower	External	Can Modify State	-
setEventPoint	External	Can Modify State	-
addEventPoint	External	Can Modify State	-
vePower2vePoint	Public	-	-
log_2	Public	-	-
balanceOf	Public	-	-

Administrable			
Function Name	Visibility	Mutability	Modifiers
setAdmin	Internal	Can Modify State	-
transferAdmin	External	Can Modify State	onlyAdmin
acceptAdmin	External	Can Modify State	-

AnyCallReceiver			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setSenders	Public	Can Modify State	onlyAdmin
setAnyCallProxy	Public	Can Modify State	onlyAdmin

AnyCallReceiver			
onReceive	Internal	Can Modify State	-
anyExecute	External	Can Modify State	onlyExecutor

VEPowerOracleReceiver			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	AnyCallReceiver
currentEpoch	Public	-	-
veKey	Public	-	-
_initDaoId	Internal	Can Modify State	-
onReceive	Internal	Can Modify State	-

Administrable			
Function Name	Visibility	Mutability	Modifiers
setAdmin	Internal	Can Modify State	-
transferAdmin	External	Can Modify State	onlyAdmin
acceptAdmin	External	Can Modify State	-

AnyCallSender			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setReceivers	Public	Can Modify State	onlyAdmin

AnyCallSender			
setAnyCallProxy	Public	Can Modify State	onlyAdmin
_anyCall	Internal	Can Modify State	-

VEPowerOracleSender			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	AnyCallSender
currentEpoch	Public	-	-
delegateVEPower	External	Payable	-
calcAvgVEPower	Public	-	-
getPower	Public	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event record

Category: Others

Content

There is a lack of event records when modifying sensitive parameters of the contract, which is not conducive to the supervision of users and the community.

Code location:MultiDAO-contracts/contracts/MultiHonor.sol #L64-67

```
function setIDCard(address IDCard_) external {
    _checkRole(DEFAULT_ADMIN_ROLE);
    IDCard = IDCard_;
}
```

Code location:MultiDAO-contracts/contracts/VEPowerOracleReceiver.sol #L69-77

```
    function setSenders(uint256[] memory chainIDs, address[] memory  senders) public
onlyAdmin {
        for (uint i = 0; i < chainIDs.length; i++) {
            sender[chainIDs[i]] = senders[i];
        }
    }

    function setAnyCallProxy(address proxy) public onlyAdmin {
        anyCallProxy = proxy;
    }
```

Code location:MultiDAO-contracts/contracts/IDNFT.sol #L31-37

```
    function setHonor(address honor_) external onlyOwner {
        honor = honor_;
    }

    function allowTransfer(uint256 tokenId) external onlyOwner {
        isAllowTransfer[tokenId] = true;
    }
```

Solution

It is recommended to add the corresponding event record.

Status

Fixed

[N2] [Suggestion] Design logic issue

Category: Design Logic Audit

Content

The __initVEEpoch function is not called in the initialization function.

Code location:MultiDAO-contracts/contracts/MultiHonor.sol #L60-62

```
function __initVEEpoch() public {
    veEpochLength = 7257600; // 12 weeks
}
```

Solution

It is recommended to add the __initVEEpoch function to the initialization function.

Status

Fixed

[N3] [Suggestion] Arithmetic accuracy deviation vulnerability

Category: Arithmetic Accuracy Deviation Vulnerability

Content

The vePower2vePoint function uses a calculation method that calculates the division first and then the multiplication, and there may be errors here.

Code location:MultiDAO-contracts/contracts/MultiHonor.sol #L211-213

```
function vePower2vePoint(uint256 v) public pure returns (uint256) {
    return 125 * log_2((v / 1 ether +1) ** 2) + 514 * v / 1 ether / 1000;
}
```

Solution

It is recommended to optimize the operation logic.

Status

Confirmed

[N4] [Suggestion] Possible spillover risk

Category: Integer Overflow and Underflow Vulnerability

Content

The contract does not specify a version and does not use SafeMath, there is a risk of overflow.

Solution

It is recommended to use solidity version 0.8 or higher.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002209150001	SlowMist Security Team	2022.09.13 - 2022.09.15	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 4 suggestions. And 1 suggestion were confirmed; 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