Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.03.16, the SlowMist security team received the SwellNetwork team's security audit application for SwellNetwork, 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 Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit version:

Project address:www.github.com/SwellNetwork/v2-core

commit:48aaf0fa95161262da3500ebc79aacdc1028b69d

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Too much permission issue	Authority Control Vulnerability	Medium	Ignored
N2	Unaudited related contracts	Authority Control Vulnerability	Low	Ignored

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:

SWNFTUpgrade			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
setswETHAddress	External	Can Modify State	onlyOwner
setFeePool	External	Can Modify State	onlyOwner
addStrategy	External	Can Modify State	onlyOwner
removeStrategy	External	Can Modify State	onlyOwner
addWhiteList	External	Can Modify State	onlyOwner
renounceOwnership	Public	-	onlyOwner
deposit	Public	Can Modify State	-
withdraw	Public	Can Modify State	-
enterStrategy	Public	Can Modify State	-
exitStrategy	Public	Can Modify State	-

SWNFTUpgrade			
batchAction	External	Can Modify State	-
stake	External	Payable	-
unstake	External	-	-
validatorsLength	External	-	-
tvl	External	-	-
tokenURI	Public	-	-
getWithdrawalCredentials	Public	-	-
_stake	Private	Can Modify State	-
_pubKeyToString	Private	-	-
_authorizeUpgrade	Internal	-	onlyOwner

Strategy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
enter	External	Can Modify State	onlyswNFT
exit	External	Can Modify State	onlyswNFT

SWDAO			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
burn	External	Can Modify State	onlyOwner

SWDAO			
mint	External	Can Modify State	onlyOwner

SWETH			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	onlyMinter
burn	External	Can Modify State	onlyMinter

4.3 Vulnerability Summary

[N1] [Medium] Too much permission issue

Category: Authority Control Vulnerability

Content

The Owner role can mint coins and burn any user's tokens for any user through the burn and mint functions in the swDAO contract and the swETH contract.

Code location:v2-core/contracts/swDAO.sol #L15-L21

```
function burn(uint256 amount) external onlyOwner{
    _burn(msg.sender, amount);
}

function mint(uint256 amount) external onlyOwner{
    _mint(msg.sender, amount);
}
```

Code location:v2-core/contracts/swETH.sol #L25-L31

```
function mint(uint256 amount) external onlyMinter{
    _mint(minter, amount);
}

function burn(uint256 amount) external onlyMinter{
    _burn(minter, amount);
}
```

Solution

It is recommended to hand over the permissions of the Owner role to the governance contract or use the timeLock contract to manage. At least multisig should be used.

Status

Ignored; After communication, the project team said: We will be deploying using Protocol DAO, a Gnosis multisig wallet.

[N2] [Low] Unaudited related contracts

Category: Authority Control Vulnerability

Content

The Owner role can modify the swETHAddress contract address through the setswETHAddress function, modify the feePool contract address through setFeePool, and add and delete the list of strategies addresses through the addStrategy and removeStrategy functions. If the modified contract address is an unaudited contract address, we cannot guarantee its security.

Code location:v2-core/contracts/swNFTUpgrade.sol #L93-L127

```
function setswETHAddress(address _swETHAddress) onlyOwner external {
    require(_swETHAddress != address(0), "Address cannot be 0");
    swETHAddress = _swETHAddress;
    emit LogSetSWETHAddress(swETHAddress);
}

/// @notice set fee pool address
/// @param _feePool The address of the fee pool
```

```
function setFeePool(address _feePool) onlyOwner external {
    require(_feePool != address(0), "Address cannot be 0");
    feePool = _feePool;
    emit LogSetFeePool(feePool);
}

/// @notice Add a new strategy
/// @param strategy The strategy address to add
function addStrategy(address strategy) onlyOwner external{
    require(strategy != address(0), "address cannot be 0");
    strategies.push(strategy);
    emit LogAddStrategy(strategy);
}

/// @notice Remove a strategy
/// @param strategy The strategy index to remove
function removeStrategy(uint strategy) onlyOwner external{
    uint length = strategies.length;
    require(strategy < length, "Index out of range");
    require(strategies[strategy] != address(0), "strategy does not exist");
    //TODO: Need to check balance before removing
    require(length >= 1, "no strategy to remove");
    address last = strategies[length-1];
    emit LogRemoveStrategy(strategy, strategies[strategy]);
    strategies[strategy] = last;
    strategies.pop();
}
```

Solution

It is recommended to hand over the permissions of the Owner role to the governance contract.

Status

Ignored; The project team will hand over the authority to the Protocol DAO governance.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002203210003	SlowMist Security Team	2022.03.16 - 2022.03.21	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 1 medium risk, 1 low risk ; 1 medium risk, 1 low risk were ignored; 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