Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.12.20, the SlowMist security team received the Filda team's security audit application for Filda2.0 iterative audit, 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:

https://github.com/fildaio/compound-protocol/commit/785b2a9252227af62257449315366cd3dc1fce8a

https://github.com/fildaio/compound-protocol/commit/54d7926ad210f8af03617d997286bad9e8a7fc69

https://github.com/fildaio/compound-protocol/commit/f5936b5e611479c55d368b5e278fd42f30e1ed76

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability	Medium	Confirming

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:

QsQuickDualLPDelegate			
Function Name	Visibility	Mutability	Modifiers
_becomeImplementation	Public	Can Modify State	-
claimRewards	Public	Can Modify State	-
setRewardFToken	External	Can Modify State	-
borrow	External	Can Modify State	-
repayBorrow	External	Can Modify State	-
repayBorrowBehalf	External	Can Modify State	-
liquidateBorrow	External	Can Modify State	-
flashLoan	External	Can Modify State	-
_addReserves	External	Can Modify State	-
transferTokens	Internal	Can Modify State	-
getCashPrior	Internal	-	-

QsQuickDualLPDelegate			
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-
seizeInternal	Internal	Can Modify State	-
redeem	External	Can Modify State	-
redeemUnderlying	External	Can Modify State	-
claimFromQuick	Internal	Can Modify State	-
mintToFilda	Internal	Can Modify State	-
updateLPSupplyIndex	Internal	Can Modify State	-
updateSupplierIndex	Internal	Can Modify State	-
tokenBalance	Internal	-	-

QsQuickLPDelegate			
Function Name	Visibility	Mutability	Modifiers
_becomeImplementation	Public	Can Modify State	-
claimRewards	Public	Can Modify State	-
setRewardFToken	External	Can Modify State	-
borrow	External	Can Modify State	-
repayBorrow	External	Can Modify State	-
repayBorrowBehalf	External	Can Modify State	-
liquidateBorrow	External	Can Modify State	-

QsQuickLPDelegate			
flashLoan	External	Can Modify State	-
_addReserves	External	Can Modify State	-
transferTokens	Internal	Can Modify State	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-
seizeInternal	Internal	Can Modify State	-
redeem	External	Can Modify State	-
redeemUnderlying	External	Can Modify State	-
claimFromQuick	Internal	Can Modify State	-
mintToFilda	Internal	Can Modify State	-
updateLPSupplyIndex	Internal	Can Modify State	-
updateSupplierIndex	Internal	Can Modify State	-
tokenBalance	Internal	-	-

QsSushiLPDelegate			
Function Name	Visibility	Mutability	Modifiers
_becomeImplementation	Public	Can Modify State	-
claimRewards	Public	Can Modify State	-
setRewardFToken	External	Can Modify State	-

QsSushiLPDelegate			
borrow	External	Can Modify State	-
repayBorrow	External	Can Modify State	-
repayBorrowBehalf	External	Can Modify State	-
liquidateBorrow	External	Can Modify State	-
flashLoan	External	Can Modify State	-
_addReserves	External	Can Modify State	-
transferTokens	Internal	Can Modify State	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-
seizeInternal	Internal	Can Modify State	-
redeem	External	Can Modify State	-
redeemUnderlying	External	Can Modify State	-
claimRewardsFromSushi	Internal	Can Modify State	-
mintToFilda	Internal	Can Modify State	-
updateLPSupplyIndex	Internal	Can Modify State	-
updateSupplierIndex	Internal	Can Modify State	-
tokenBalance	Internal	-	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the QsQuickDualLPDelegate, QsQuickLPDelegate, and QsSushiLPDelegate contracts, the admin role can modify the stakingRewards contract address through the _becomeImplementation function. Since the contract will approve the user's collateral to the stakingRewards contract, this will lead to excessive risk of the admin's authority.

Code location:

contracts/QsQuickDualLPDelegate.sol

contracts/QsQuickLPDelegate.sol

contracts/QsSushiLPDelegate.sol

```
    function _becomeImplementation(bytes memory data) public {
        super._becomeImplementation(data);

        (address stakingRewardsAddr, address ftokenStorageAddr) = abi.decode(data,
(address, address));
        stakingRewards = IStakingDualRewards(stakingRewardsAddr);
        require(address(stakingRewards.stakingToken()) == underlying, "mismatch
underlying");

        bool pushComp = false;
        if (rewardsTokens.length == 0) {
            pushComp = true;
            rewardsTokens.push(address(stakingRewards.rewardsTokenA()));
            rewardsTokens.push(address(stakingRewards.rewardsTokenB()));
        }


        FTokenStorage ftokenStorage = FTokenStorage(ftokenStorageAddr);
        for (uint8 i = 0; i < rewardsTokens.length; i++) {
            // ignore native token, delegator cannot send value
            if(rewardsTokens[i] == ftokenStorage.WETH()) continue;

            address ftoken = ftokenStorage.ftoken(rewardsTokens[i]);
            if (ftoken == address(0)) continue;
```

```
            harvestComp = true;
            rewardsFToken[rewardsTokens[i]] = ftoken;

            // Approve moving mdx rewards into the fMdx contract.
            EIP20Interface(rewardsTokens[i]).approve(ftoken, uint(-1));
        }

        if (harvestComp && pushComp) {
            rewardsTokens.push(Qstroller(address(comptroller)).getCompAddress());
        }

        // Approve moving our LP into the heco pool contract.
        EIP20Interface(underlying).approve(stakingRewardsAddr, uint(-1));
    }
```

Solution

It is recommended to transfer the ownership of admin to community governance

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002112220001	SlowMist Security Team	2021.12.20 - 2021.12.22	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 vulnerabilities. All the 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