Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.03.10, the SlowMist security team received the Parasail team's security audit application for pFIL incremental 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.
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

Filecoin perpetual pledge swap (pFIL) smart contracts.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Unlimited triggering of OnRequestCalculate events	Design Logic Audit	Medium	Fixed

NO	Title	Category	Level	Status
N2	Inconsistent event logging	Design Logic Audit	Low	Fixed
N3	Redundant code	Others	Suggestion	Fixed
N4	Missing zero address check	Others	Suggestion	Fixed
N5	Missing event records	Others	Suggestion	Fixed
N6	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/parasail-network/pFIL-contracts

Initial audit version: a958f2508680610a765ab84cc274f4b616424260

Final audit version: 2662612d8fad492491b1731d4f0975c7320805a6

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:

AgentImplContract			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
<Fallback>	External	Payable	-

AgentImplContract			
initialize	External	Can Modify State	initializer
agentWithdrawFromMiner	External	Can Modify State	-
getReservedBalance	Public	-	-
calculateSafePledge	External	Can Modify State	onlyOwner
updateControlAddress	External	Can Modify State	onlyProtocol
reclaimOwnerAddress	External	Can Modify State	onlyOwner
delegateOwnerAddress	External	Can Modify State	onlyOwner
changeBeneficiary	External	Can Modify State	onlyOwner
paybackFIL	External	Payable	-
getNodeBalance	External	-	-
getAvailableBalance	Public	-	-
getOwnerAddress	Public	-	-
isActive	Public	-	-
_getOwnerReturn	Internal	-	-
_getBeneficiary	Internal	-	-
_getBeneficiaryRaw	Internal	-	-
_changeOwnerAddressWrapper	Internal	Can Modify State	-
_sendRequestSafePledge	Internal	Can Modify State	-
getTotalMinted	Public	-	-
_resetAgent	Internal	Can Modify State	-
_getIDAddress	Internal	-	-
_validateOriginOwner	Internal	-	-

AgentImplContract			
_validateAddress	Internal	-	-
version	External	-	-

Repl			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
updateAgentImpl	External	Can Modify State	onlyOwner
setPendingSwapTime	External	Can Modify State	onlyOwner
setAddress	External	Can Modify State	onlyOwner
setFee	External	Can Modify State	onlyOwner
controlProtocol	External	Can Modify State	onlyOwner
setInterestRateModel	External	Can Modify State	onlyOwner
createAgent	External	Can Modify State	whenNotPaused
borrowPFIL	External	Can Modify State	nonReentrant whenNotPaused
borrowFIL	External	Can Modify State	nonReentrant whenNotPaused
repayPFIL	External	Can Modify State	nonReentrant whenNotPaused
repayFIL	External	Payable	nonReentrant whenNotPaused
depositFIL	External	Payable	nonReentrant whenNotPaused
withdrawFIL	External	Can Modify State	nonReentrant whenNotPaused
debtOf	Public	-	-
covertDebtSharesToDebt	Public	-	-
convertDebtToDebtShares	Public	-	-

Repl			
addDebt	Internal	Can Modify State	-
removeDebt	Internal	Can Modify State	-
accureInterest	Public	Can Modify State	-
requestCalculate	External	Can Modify State	isAgentCall
receiveWithdraw	External	Payable	isAgentCall
updateAgentSafePledge	External	Can Modify State	onlySteward
updateAgentControlAddress	External	Can Modify State	onlySteward
resetAgent	External	Can Modify State	isAgentCall
onAgentDelegated	External	Can Modify State	isAgentCall
onAgentWithdraw	External	Can Modify State	isAgentCall
getPFILAddress	External	-	-
getAgents	External	-	-
getAgentsCount	External	-	-
getAgent	External	-	-
isAgent	Public	-	-
_validateBorrowAddDebt	Internal	Can Modify State	-
_securityCheck	Internal	-	-
version	External	-	-
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
getImplementation	External	-	-
_checkValidMiner	Internal	-	-
burnFromWhenPaused	External	Can Modify State	onlyOwner

Repl			
migrateToV3	Public	Can Modify State	reinitializer onlyOwner
migrateAgents	Public	Can Modify State	onlyOwner
_migrateAgents	Internal	Can Modify State	-

InterestRateModel			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getBorrowRate	External	-	-

4.3 Vulnerability Summary

[N1] [Medium] Unlimited triggering of OnRequestCalculate events

Category: Design Logic Audit

Content

In the AgentImplContract contract, once the owner authority of the miner is transferred to the `agentID` address, the `owner` role of the contract can trigger the `OnRequestCalculate` event of the Repl contract indefinitely by repeatedly calling the `calculateSafePledge` function, which may lead to an increase in events, interfere with the normal operation of the monitoring service, and may affect the system that relies on this event for business processing.

- contracts/AgentImplementation.sol#L122-L124

```
function calculateSafePledge() external onlyOwner {
    _sendRequestSafePledge(true);
}
```

Solution

It is recommended to add a call interval limit in the calculateSafePledge function to ensure that each agent can only trigger a calculation request once within a reasonable time interval.

Status

Fixed

[N2] [Low] Inconsistent event logging

Category: Design Logic Audit

Content

In the `repayFIL` function of the Repl contract, the actual amount repaid may be `amount` rather than `msg.value`, which may lead to off-chain analysis errors.

- contracts/Repl.sol#L297-L308

```
    function repayFIL(address agent) external payable nonReentrant whenNotPaused {
        // accureInterest();
        uint256 amount = debtOf(agent) > msg.value ? msg.value : debtOf(agent);
        removeDebt(amount, agent);
        if (msg.value > amount) {
            (bool success, ) = msg.sender.call{value: msg.value - amount}("");
            if (!success) revert TransferFailed();
        }
        _securityCheck();

        emit Repay(agent, msg.value);
    }
```

Solution

It is recommended to replace the msg.value parameter of the Repay event with the amount parameter.

Status

Fixed

[N3] [Suggestion] Redundant code

Category: Others

Content

In the Repl contract, the `onAgentWithdraw` function can be called by AgentImplContract contract and trigger the `OnAgentWithdraw` event, but the `onAgentWithdraw` function is not called in the AgentImplContract contract.

- contracts/Repl.sol#L473-L475

```
    function onAgentWithdraw(uint64 _minerID, uint256 amount) external isAgentCall {
        emit OnAgentWithdraw(msg.sender, _minerID, amount);
    }
```

Solution

It is recommended to delete the redundant code.

Status

Fixed

[N4] [Suggestion] Missing zero address check

Category: Others

Content

In the Repl contract, the `initialize` function `updateAgentImpl` function, `setInterestRateModel` function, `borrowFIL` function, and `migrateToV3` function lack zero address checks for address type parameters.

- contracts/Repl.sol#L136-L150, L156-L159, L213-L215, L263-L279, L572-L590

```
    function initialize(address _pFIL, address _steward, address _feeTo) public
initializer {
        __ReentrancyGuard_init();
        __Ownable_init();
        __Pausable_init();
        __UUPSUpgradeable_init();
        if (_pFIL == address(0)) revert InvalidAddress();
        pFIL = IPFIL(_pFIL);
        steward = _steward;
        feeTo = _feeTo;
        pendingSwap = 24 hours;
        // fee = amount * days * feePerDay / 10000
        feePerDay = 50;
        // ratio = dividendTakeRate / 10000
        dividendTakeRate = 1000;
    }

    function updateAgentImpl(address _agentImpl) external onlyOwner {
        agentImplementation = _agentImpl;
        emit AgentImplUpdated(_agentImpl);
    }
```

```
    function setInterestRateModel(address _interestRateModel) external onlyOwner {
        interestRateModel = IInterestRateModel(_interestRateModel);
    }

    function borrowFIL(
        uint256 amount,
        address agent,
        address receiver
    ) external nonReentrant whenNotPaused {
        if (msg.sender != IAgentContract(agent).owner()) revert CallerNotAgentOwner();
        if (!IAgentContract(agent).isActive()) revert AgentNotActive();
        accureInterest();
        if (agentPledgeInfo[agent].lastSafePledgeUpdateTime + pendingSwap >
block.timestamp)
            revert PendingCalculate();
        _validateBorrowAddDebt(agent, amount);
        (bool success, ) = receiver.call{value: amount}("");
        if (!success) revert TransferFailed();
        _securityCheck();

        emit Borrow(agent, amount);
    }

    function migrateToV3(
        address interestRateModel_,
        address agentImplementation_,
        uint256 conversionRate,
        address[] calldata activeAgents
    ) public reinitializer(3) onlyOwner {
        lastAccureInterestTime = block.timestamp;
        interestRateModel = IInterestRateModel(interestRateModel_);
        agentImplementation = agentImplementation_;

        // migrate agents data
        _migrateAgents(activeAgents, conversionRate);

        // rebase pFIL
        uint256 pFILTotalSupply = pFIL.totalSupply();
        uint256 pFILToSlash = pFILTotalSupply - (pFILTotalSupply * conversionRate) /
1e18;
        pFIL.burnFrom(address(this), pFILToSlash, 0);
        _securityCheck();
    }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N5] [Suggestion] Missing event records

Category: Others

Content

In the Repl contract, the `setInterestRateModel` function modifies important contract variables but lacks event records.

- contracts/Repl.sol#L213-L215

```
function setInterestRateModel(address _interestRateModel) external onlyOwner {
    interestRateModel = IInterestRateModel(_interestRateModel);
}
```

Solution

It is recommended to add event logging.

Status

Fixed

[N6] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the Repl contract, the `Owner` role can modify important variables in the contract through the following functions.

- contracts/Repl.sol#L156-L159, L165-L168, L175-L172, L189-L196, L202-L211, L213-L215, L557-L563, L572-L590, L597-L599

```
function updateAgentImpl
function setPendingSwapTime
function setAddress
function setFee
function controlProtocol
function setInterestRateModel
```

```
    function burnFromWhenPaused
    function migrateToV3
```

2.In the Repl contract, the `Steward` role can modify the user's `safePledge` variable through the `updateAgentSafePledge` function and modify the `worker` and `control` addresses of the miner node through the `updateAgentControlAddress` function.

- contracts/Repl.sol#L438-L448

```
    function updateAgentSafePledge
    function updateAgentControlAddress
```

3.The Repl contract uses OpenZeppelin's UUPSUpgradeable upgrade mechanism, which allows the `Owner` role to upgrade the smart contract.

- contracts/Repl.sol#L7, L536

```
    import {UUPSUpgradeable} from "@openzeppelin/contracts-
upgradeable/proxy/utils/UUPSUpgradeable.sol";

    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}
```

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. And 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

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002503110001	SlowMist Security Team	2025.03.10 - 2025.03.11	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 2 medium risk, 1 low risk, 3 suggestion.

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