Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.11.16, the SlowMist security team received the team's security audit application for pFIL, 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

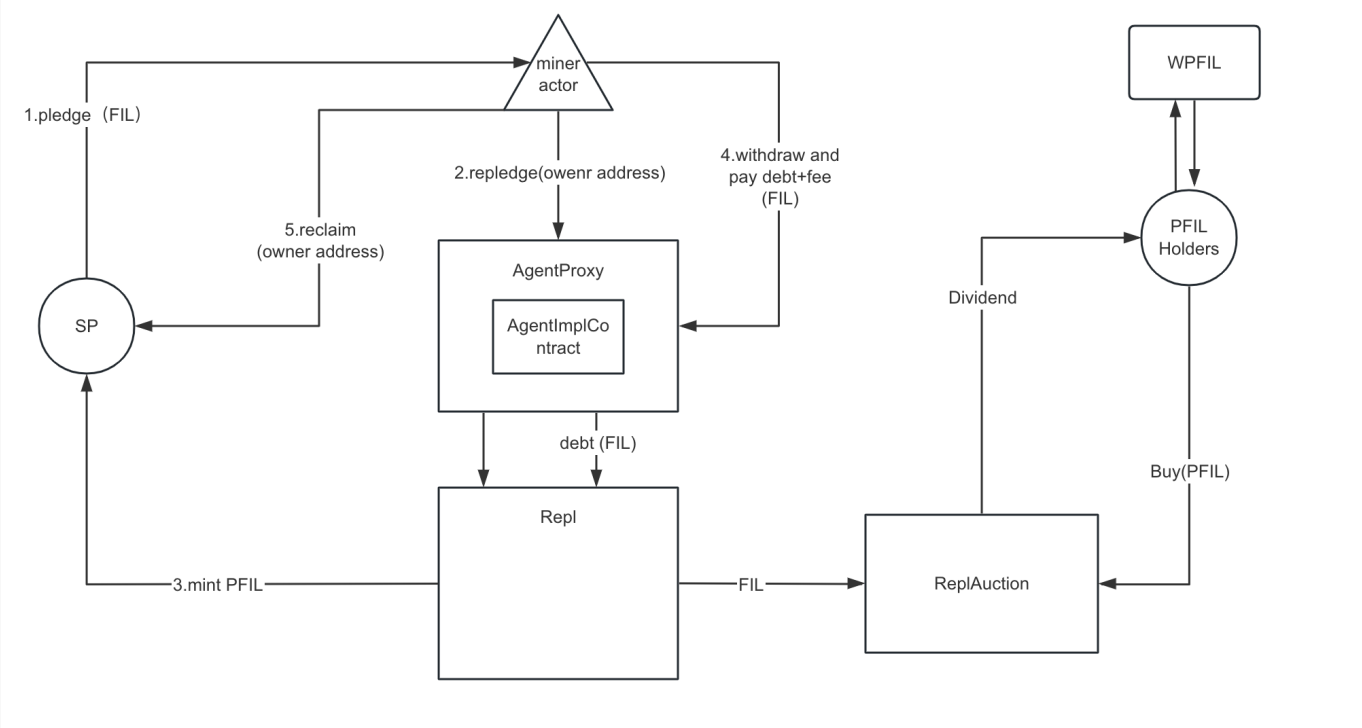
Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
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

An protocol designed to stake the Miner actor's owner address and mint pFIL for the staker.

Fund flow diagram：



3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	lastPledgeSwapTime value not checked	Design Logic Audit	Suggestion	Fixed
N2	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N3	Variable naming error	Others	Low	Fixed
N4	Return value unchecked	Others	Suggestion	Fixed
N5	Avoid using transfer()	Others	Suggestion	Fixed
N6	Missing event record	Others	Suggestion	Fixed
N7	Redundant code	Others	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/Project-pFIL/pFIL-contracts

Initial audit commit: fdfb84c62978e9b4d11fce04479817b2d75601fd

Final audit commit: fcec236f84fc2016cbbe3702202e2c9853ef7852

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	-
initialize	External	Can Modify State	initializer
pledgeSwap	External	Can Modify State	onlyOwner nonReentrant
agentWithdrawFromMiner	External	Can Modify State	-
calculateSafePledge	Public	Can Modify State	onlyOwner
updateSafePledge	External	Can Modify State	onlyProtocol
updateControlAddress	External	Can Modify State	onlyProtocol
reclaimOwnerAddress	External	Can Modify State	onlyOwner
delegateOwnerAddress	External	Can Modify State	onlyOwner
getAvailableBalance	Public	-	-
getOwnerAddress	Public	-	-

AgentImplContract			
_getOwnerReturn	Internal	-	-
_getBeneficiary	Internal	-	-
_changeOwnerAddressWrapper	Internal	Can Modify State	-
getOutstandingTargetPledge	Public	-	-
_resetAgent	Internal	Can Modify State	-
_getIDAddress	Internal	-	-
_validateOriginOwner	Internal	-	-
_validateIsSameAddress	Internal	-	-

PFIL			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit
addMinter	External	Can Modify State	onlyOwner
pause	Public	Can Modify State	onlyMinter
unpause	Public	Can Modify State	onlyMinter
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
mint	External	Can Modify State	onlyMinter
burnFrom	Public	Can Modify State	onlyMinter

PFIL			
increaseAllowance	Public	Can Modify State	-
decreaseAllowance	Public	Can Modify State	-
getTotalShares	External	-	-
sharesOf	External	-	-
getSharesByFIL	Public	-	-
getFILByShares	Public	-	-
transferShares	External	Can Modify State	-
transferSharesFrom	External	Can Modify State	-
_totalPooledPledge	Internal	-	-
_transfer	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_spendAllowance	Internal	Can Modify State	-
_getTotalShares	Internal	-	-
_sharesOf	Internal	-	-
_transferShares	Internal	Can Modify State	whenNotPaused
_mintShares	Internal	Can Modify State	-
_burnShares	Internal	Can Modify State	-
_emitTransferEvents	Internal	Can Modify State	-
_emitTransferAfterMintingShares	Internal	Can Modify State	-

Repl			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer

Repl			
updateAgentImpl	External	Can Modify State	onlyOwner
setPendingSwapTime	External	Can Modify State	onlyOwner
setAddress	External	Can Modify State	onlyOwner
setFee	External	Can Modify State	onlyOwner
setAuction	External	Can Modify State	onlyOwner
controlProtocol	External	Can Modify State	onlyOwner
createAgent	External	Can Modify State	-
replContractMintPFIL	External	Can Modify State	isAgentCall nonReentrant
requestCalculate	External	Can Modify State	isAgentCall
receiveWithdraw	External	Payable	isAgentCall
updateAgentSafePledge	External	Can Modify State	onlySteward
updateAgentControlAddress	External	Can Modify State	onlySteward
auctionBidded	External	Can Modify State	isAuctionCall nonReentrant
getAgents	External	-	-
getAgent	External	-	-
isAgent	Public	-	-
_securityCheck	Internal	-	-
_calculateAgentFee	Internal	-	-
version	External	-	-
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
getImplementation	External	-	-
_checkValidMiner	Internal	-	-

Repl			
burnFromWhenPaused	External	Can Modify State	onlyOwner

AgentProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_beforeFallback	Internal	Can Modify State	-
_implementation	Internal	-	-
getAgentImplemention	Public	-	-

ReplAuction			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setProtocol	External	Can Modify State	onlyOwner
setDuration	External	Can Modify State	onlyOwner
setStartPrice	External	Can Modify State	onlyOwner
controlAuction	External	Can Modify State	onlyOwner
receiveFIL	External	Can Modify State	onlyProtocol
buy	External	Can Modify State	nonReentrant
getPrice	Public	-	-
getRemainingFILForAuction	Public	-	-
auctionIsExpired	Public	-	-
_startAuction	Internal	Can Modify State	-
version	External	-	-

ReplAuction			
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
getImplementation	External	-	-

wPFIL			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20Permit ERC20
wrap	External	Can Modify State	-
unwrap	External	Can Modify State	-
getWFILByPFIL	External	-	-
getPFILByWPFIL	External	-	-
PFILPerToken	External	-	-
tokensPerPFIL	External	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] lastPledgeSwapTime value not checked

Category: Design Logic Audit

Content

In the _calculateAgentFee function of the Repl contract, when calculating fees for the first pledge, the value of the `agentVars[_agent].lastPledgeSwapTime` parameter is 0, making the value of the `_during` parameter equal to block.timestamp, which is too large.

```
    function _calculateAgentFee(
        address _agent,
        uint256 _baseAmount
    ) internal view returns (uint256 _fee) {
        uint256 _during = block.timestamp - agentVars[_agent].lastPledgeSwapTime;
        // basement = 1000000
```

```
            _fee = (_during * _baseAmount * feePerDay) / (1000000 * 24 * 3600);
        }
```

Solution

It is recommended to check whether the agentVars[_agent].lastPledgeSwapTime parameter is 0. If so, the value of the _during parameter is return 0.

Status

Fixed

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

Protocol roles can update control addresses through the `updateControlAddress` function.

- contracts/AgentImplementation.sol#L158-L172

```
    function updateControlAddress(
        uint64 _worker,
        uint64[] calldata _controlAddresses
    ) external onlyProtocol {
        CommonTypes.FilAddress[] memory controllers = new CommonTypes.FilAddress[](
            _controlAddresses.length
        );
        for (uint64 i = 0; i < _controlAddresses.length; i++) {
            controllers[i] = FilAddresses.fromActorID(_controlAddresses[i]);
        }
        MinerAPI.changeWorkerAddress(
            CommonTypes.FilActorId.wrap(actorID),
            MinerTypes.ChangeWorkerAddressParams(FilAddresses.fromActorID(_worker),
controllers)
        );
    }
```

In the PFIL contract, the Owner role can add the minter role. The minter role can mint pFIL tokens at will without an upper limit, and the malicious minter role can participate in the auction and bid to purchase FIL.The minter role can burn the user's pFIL and shares at will, affecting the balance of pFIL holders.The minter role can lock the contract, making it impossible to transfer pFIL.

- contracts/PFIL.sol

```
addMinter
mint
burnFrom
pause
unpause
```

In the Repl contract and the ReplAuction contract, the Owner role can modify the key variables of the contract and upgrade the contract.It is important to note that the Steward role can modify the controller address of the Miner actor through the `updateAgentControlAddress()` function, affecting the node's operation and maintenance permissions. And the Steward role can set the value of the parameter safePledge, which affects the number of user mint pFILs.

- contracts/Repl.sol

```
updateAgentImpl
setPendingSwapTime
setAddress
setFee
setAuction
controlProtocol
updateAgentControlAddress
_authorizeUpgrade
burnFromWhenPaused
```

- contracts/ReplAuction.sol

```
setProtocol
setDuration
setStartPrice
_authorizeUpgrade
```

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 authority involving emergency contract suspension can be managed by the

EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N3] [Low] Variable naming error

Category: Others

Content

In the `struct AuctionInfo` of the ReplAuction contract, the `pFILBoughtBack` parameter records the number of FIL sold in the specified round of auction, not the number of pFIL.

contracts/ReplAuction.sol#L41-L47

```
    struct AuctionInfo {
        uint filForAuction;
        uint pFILBoughtBack;
        uint startAtTime;
        uint expiresAtTime;
        uint id;
    }
```

Solution

It is recommended to modify the name of the pFILBoughtBack parameter

Status

Fixed

[N4] [Suggestion] Return value unchecked

Category: Others

Content

In the `wrap()` function and `unwrap()` function of the wPFIL contract, the `transfer()` function and `transferFrom()` function are used to transfer ERC20 tokens, and the return value is not checked.

- contracts/wPFIL.sol#L47-53

```
    function wrap(uint256 _pFILAmount) external returns (uint256) {
        require(_pFILAmount > 0, "wpFIL: can't wrap zero pFIL");
        uint256 wpFILAmount = pFIL.getSharesByFIL(_pFILAmount);
```

```
        _mint(msg.sender, wpFILAmount);
        pFIL.transferFrom(msg.sender, address(this), _pFILAmount);
        return wpFILAmount;
    }
```

- contracts/wPFIL.sol#L63-L69

```
    function unwrap(uint256 _wpFILAmount) external returns (uint256) {
        require(_wpFILAmount > 0, "wpFIL: zero amount unwrap not allowed");
        uint256 pFILAmount = pFIL.getFILByShares(_wpFILAmount);
        _burn(msg.sender, _wpFILAmount);
        pFIL.transfer(msg.sender, pFILAmount);
        return pFILAmount;
    }
```

Solution

It is recommended to check the return value of transfer() and transferFrom() or use safetransfer() and safetransferFrom().

Status

Fixed

[N5] [Suggestion] Avoid using transfer()

Category: Others

Content

In the AgentImplContract contract,, it is not recommended to use `transfer()` because the gas cost changes introduced by EIP 1884 may cause the contract to no longer be secure.

- contracts/Repl.sol#L235-L250,L288-L306

```
    receiveWithdraw
    auctionBidded
```

- contracts/AgentImplementation.sol

```
    reclaimOwnerAddress
```

Solution

It is recommended to use call() function instead of transfer() function.

Status

Fixed

[N6] [Suggestion] Missing event record

Category: Others

Content

Missing events for state changes in the contract.

- contracts/Repl.sol

```
updateAgentImpl
setAddress
setFee
setAuction
controlProtocol
```

- contracts/ReplAuction.sol

```
setProtocol
setDuration
setStartPrice
```

- contracts/PFIL.sol

```
addMinter
mint
burnFrom
```

Solution

Recording events.

Status

Fixed

[N7] [Suggestion] Redundant code

Category: Others

Content

The defined variables and events are not used.

- contracts/AgentImplementation.sol
- contracts/Repl.sol
- contracts/ReplAuction.sol

```
    uint256[49] private __gap;
```

- contracts/PFIL.sol

```
    event MinterAdded(address indexed account);
    event MinterRemoved(address indexed account);
```

Solution

Can remove useless code and add event logging in the corresponding function.

Status

Fixed; The project party stated that the __gap parameter will be reserved for contract upgrades.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002311210001	SlowMist Security Team	2023.11.16 - 2023.11.21	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, 1 low risk, 5 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