Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.02.19, the SlowMist security team received the 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	Invalid condition check	Design Logic Audit	Low	Fixed
N2	Redundant code	Others	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N3	Mint error of retainedPFIL	Design Logic Audit	Medium	Fixed
N4	Bypass passive minting pFIL	Design Logic Audit	High	Fixed
N5	Function reclaimOwnerAddress cannot be used normally	Design Logic Audit	High	Fixed
N6	Preemptive initialization	Design Logic Audit	Suggestion	Acknowledged
N7	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N8	Missing event record	Others	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

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

Initial audit commit:39df36c02cedbef29c0818cc92b76d1f7fcca961

(Focus on code changes from version: fcec236f84fc2016cbbe3702202e2c9853ef7852 to version:

39df36c02cedbef29c0818cc92b76d1f7fcca961)

Final audit commit:8f0b5ecb58315132df11d62422c45c8ab05b88a8

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	-
getReservedBalance	Public	-	-
calculateSafePledge	External	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
changeBeneficiary	External	Can Modify State	onlyOwner
onAuctionEnd	External	Can Modify State	onlyProtocol
paybackPFIL	External	Can Modify State	needClearing
paybackFIL	External	Payable	-
withdrawFIL	External	Can Modify State	onlyOwner nonReentrant
passiveMint	Public	Can Modify State	nonReentrant
getAvailableBalance	Public	-	-
getOwnerAddress	Public	-	-
_getOwnerReturn	Internal	-	-

AgentImplContract			
_getBeneficiary	Internal	-	-
_getBeneficiaryRaw	Internal	-	-
_changeOwnerAddressWrapper	Internal	Can Modify State	-
_sendRequestSafePledge	Internal	Can Modify State	-
getOutstandingTargetPledge	Public	-	-
getTotalMinted	Public	-	-
getMultiplier	Internal	-	-
_resetAgent	Internal	Can Modify State	-
_getIDAddress	Internal	-	-
_validateOriginOwner	Internal	-	-
_validateAddress	Internal	-	-
version	External	-	-

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	-	-

PFIL			
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
mint	External	Can Modify State	onlyMinter
burnFrom	Public	Can Modify State	onlyMinter
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	-

PFIL			
_emitTransferAfterMintingShares	Internal	Can Modify State	-

ReplOracle			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
getAggregatedPrice	External	-	-
appendV3Pair	Public	Can Modify State	onlyOwner
calculateV3Price	Public	-	-
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
getImplementation	External	-	-
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
setAuction	External	Can Modify State	onlyOwner
controlProtocol	External	Can Modify State	onlyOwner

Repl			
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
onAgentDelegated	External	Can Modify State	isAgentCall
onAgentWithdraw	External	Can Modify State	isAgentCall
getRewardPerSecond	External	-	-
passiveMintPFIL	External	Can Modify State	isAgentCall
onPaybackPFIL	External	Can Modify State	isAgentCall
getPFILAddress	External	-	-
getAgents	External	-	-
getAgent	External	-	-
isAgent	Public	-	-
_securityCheck	Internal	-	-
_calculateAgentFee	Internal	-	-
v2Init	External	Can Modify State	-
version	External	-	-
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
getImplementation	External	-	-

Repl			
_checkValidMiner	Internal	-	-
burnFromWhenPaused	External	Can Modify State	onlyOwner

ReplAuction			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
setProtocol	External	Can Modify State	onlyOwner
setDuration	External	Can Modify State	onlyOwner
setStartPrice	External	Can Modify State	onlyOwner
setPriceStep	External	Can Modify State	onlyOwner
controlAuction	External	Can Modify State	onlyOwner
receiveFIL	External	Can Modify State	onlyProtocol
buy	External	Can Modify State	nonReentrant
getPriceByAgent	Public	-	-
getRemainingFILForAuction	External	-	-
auctionIsExpired	External	-	-
_startAuction	Internal	Can Modify State	-
version	External	-	-
_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	-
getWPFILByPFIL	External	-	-
getPFILByWPFIL	External	-	-
PFILPerToken	External	-	-
tokensPerPFIL	External	-	-

4.3 Vulnerability Summary

[N1] [Low] Invalid condition check

Category: Design Logic Audit

Content

In the `getMultiplier` function of the AgentImplContract contract, the situation of `lastPassiveMintingTime` = 0 doesn‘t exist. If `lastPassiveMintingTime`=0, `lastPassiveMintingTime` will be set to block.timestamp in the `passiveMint` function.

- AgentImplementation.sol#L390-L401

```
    function getMultiplier() internal view returns (uint256) {
        // 0.2/180/24/3600 = 12860082304 base 1e18
        uint256 multiperPer = 12860082304;
        uint cur = 1.2e18 - ((block.timestamp - delegateTime) % 31104000) *
multiperPer;
        uint pre;
        if (lastPassiveMintingTime == 0) {
            pre = 1.2e18;
        } else {
            pre = 1.2e18 - ((lastPassiveMintingTime - delegateTime) % 31104000) *
multiperPer;
        }
        return (cur + pre) / 2;
    }
```

Solution

It is recommended to delete the code in the getMultiplier function that determines whether lastPassiveMintingTime is 0.

Status

Fixed

[N2] [Suggestion] Redundant code

Category: Others

Content

The `lastPledgeSwapTime` parameter in struct `AgentLocalVars` is not used.

- Repl.sol#L61

```
uint256 lastPledgeSwapTime;
```

Solution

It is recommended to remove redundant code.

Status

Acknowledged; The project team stated that this parameter is to prevent the impact of old data when the contract is upgraded.

[N3] [Medium] Mint error of retainedPFIL

Category: Design Logic Audit

Content

In the `auctionBidded` function of the Repl contract, `retainedPFIL` was incorrectly minted to `msg.sender` and should be minted to the `_agent` address.

- Repl.sol#L335-L361

```
  function auctionBidded(
        uint256 _FILamount,
        uint256 _pFILAmount,
        address _agent,
        address _winner
    ) external isAuctionCall nonReentrant {
        if (!(_FILamount > 0 && _pFILAmount > 0 && _pFILAmount >= _FILamount))
            revert InvalidValue();
        _securityCheck();
        uint _debt = IAgentContract(_agent).getOutstandingTargetPledge();
        uint retainedPFIL;
        if (_pFILAmount > _debt) {
            retainedPFIL = _pFILAmount - _debt;
        }
        pFIL.burnFrom(_winner, _pFILAmount, _pFILAmount); // burn all pFIL buy back
        if (retainedPFIL > 0) {
            pFIL.mint(msg.sender, retainedPFIL);
        }
        totalFILAuctioned += _FILamount;
        totalBurnedAmount += _pFILAmount - retainedPFIL;
        // Transfer FIL to winner
        (bool success, ) = payable(_winner).call{value: _FILamount}("");
        if (!success) revert WinnerTransferFailed();
        //update recoveredPledgge
        IAgentContract(_agent).onAuctionEnd(_pFILAmount -retainedPFIL );
        emit OnAuctionBidded(_winner, _FILamount, _pFILAmount, _agent);
    }
```

Solution

It is recommended to change msg.sender in pFIL.mint(msg.sender, retainedPFIL); to _agent.

Status

Fixed

[N4] [High] Bypass passive minting pFIL

Category: Design Logic Audit

Content

In the AgentImplContract contract, after the user pledges using the `pledgeSwap` function, as long as the user does not use the `agentWithdrawFromMiner` function and `passiveMint` function during the pledge period, and then repays the debt through the `paybackPFIL` function and `paybackFIL` function, and then triggers the `passiveMint` function, mint passive pFIL is not required, and can Successfully changed the ownership of Miner Actor back to itself.

- contracts/AgentImplementation.sol

  Solution

  It is recommended to modify the way of triggering the passiveMint function to prevent users from bypassing it.

  Status

  Fixed

[N5] [High] Function reclaimOwnerAddress cannot be used normally

Category: Design Logic Audit

Content

In the AgentImplContract contract, there are two situations that prevent users from using the reclaimOwnerAddress function to change ownership of Miner Actor back to itself.

1. When the user uses the `passiveMint` function to initialize the `delegateTime` parameter and `lastPassiveMintingTime` parameter, if the user does not pledge, it will cause `getOutstandingTargetPledge()` == 0 and `lastPassiveMintingTime` cannot be updated. This makes it impossible for users to change ownership of Miner Actor back to itself without staking.
2. When the user's `TotalMinted` amount is greater than `safePledge` and cannot be pledged anymore, and `block.timestamp` - `lastPassiveMintingTime` > 1 days, the user first uses the `paybackPFIL` function or `paybackFIL` function to repay the debt, because at this time `getOutstandingTargetPledge()` == 0, resulting in the inability to Use the `passiveMint` function to update the `lastPassiveMintingTime` parameter. As a result, users can no longer pledge to update `lastPassiveMintingTime` , and cannot change ownership of Miner Actor back to itself.

- AgentImplementation.sol

  Solution

  It is recommended to add in the passiveMint function to update lastPassiveMintingTime when getOutstandingTargetPledge() == 0.

  Status

  Fixed

[N6] [Suggestion] Preemptive initialization

Category: Design Logic Audit

Content

In the Repl contract, the `v2Init` function can be called by any user, which may cause an incorrect `replOracleAddress` to be passed in.

- Repl.sol#L463-L467

```
  function v2Init(address replOracleAddress) external {
        require(totalBurnedAmount == 0 || address(replOracle) == address(0),
"Inited");
        totalBurnedAmount = totalPledgeSwapAmount - pFIL.totalSupply();
        replOracle = IReplOracle(replOracleAddress);
    }
```

Solution

It is recommended to add the onlyOwner modifier.

Status

Acknowledged

[N7] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

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.

- PFIL.sol

```
addMinter
mint
burnFrom
```

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.

- Repl.sol

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

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

[N8] [Suggestion] Missing event record

Category: Others

Content

Missing events for state changes in the contract.

- ReplAuction.sol

```
setProtocol
```

- Repl.sol

```
setAuction
```

- PFIL.sol

```
mint
burnFrom
```

Solution

Recording events.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002402200002	SlowMist Security Team	2024.02.19 - 2024.02.20	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 high risk, 2 medium risk, 1 low risk, 3 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.