Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.08.30, the SlowMist security team received the Pontoon team's security audit application for Pontoon, 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 party 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 Aduit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit version:

https://github.com/pontoonfi/pontoon-staking

commit: b530503db7fcbddb56fd47c2a0f888b3fc5a77b2

https://github.com/pontoonfi/pontoon

commit: 6a8df4f59599e1568fda97a77e1468a9224f91fb

https://github.com/pontoonfi/pontoon-token

commit: 9e4c916544728584466a1950bd425e49789da824

Fix version:

https://github.com/pontoonfi/pontoon-staking

commit: c08dbe7955e154a0789bab8ede7c25f4228d5a30 https://github.com/pontoonfi/pontoon

commit: 8146dcfd1436aeae9c231b30eff1de8d0439f2ba

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing event records	Others	Suggestion	Fixed
N2	Check time issue	Design Logic Audit	Low	Confirmed
N3	Risk of excessive authority	Authority Control Vulnerability	Low	Confirmed
N4	Compatibility issues	Design Logic Audit	Suggestion	Fixed
N5	Does not comply with the Checks-Effects-Interactions principle	Others	Suggestion	Fixed
N6	Access control issues	Authority Control Vulnerability	Low	Fixed

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:

PontoonToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
setGovernance	Public	Can Modify State	onlyGovernance
recoverToken	External	Can Modify State	onlyGovernance

PontoonTokenVesting			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
updateStartTime	External	Can Modify State	onlyOwner
addOrUpdateInvestor	External	Can Modify State	onlyOwner
addOrUpdateInvestors	External	Can Modify State	onlyOwner
recoverToken	External	Can Modify State	onlyOwner
claimTeamUnlockedTokens	External	Can Modify State	started
claimInvestorUnlockedTokens	External	Can Modify State	onlyInvestor started
_addTeam	Private	Can Modify State	-
_addRound	Private	Can Modify State	-
_massUpdateCliffEndTime	Private	Can Modify State	-

PontoonTokenVesting			
_addInvestor	Private	Can Modify State	-
_getInvestorUnlockedTokensAndVestingLeft	Private	-	-
_getTeamTokensAndVestingLeft	Private	-	-
getInvestorClaimableTokens	External	-	-
getTeamClaimableTokens	External	-	-
getInvestors	External	-	-

PontoonFarm			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
calcReward	Private	-	-
calcAccRewardPerToken	Private	-	-
userData	Public	-	-
updatePool	Private	Can Modify State	-
releaseRewards	Private	Can Modify State	-
pendingRewards	External	-	-
stake	External	Can Modify State	nonReentrant
unstake	External	Can Modify State	nonReentrant
emergencyUnstake	Public	Can Modify State	nonReentrant
withdrawRemainingRewards	External	Can Modify State	onlyOwner

PontoonFarmFactory			
Function Name	Visibility	Mutability	Modifiers
deploy	External	Can Modify State	onlyOwner
pontoonFarmsCount	Public	-	-

Utils			
Function Name	Visibility	Mutability	Modifiers
isContract	Public	-	-

Vault			
Function Name	Visibility	Mutability	Modifiers
safeRewardTransfer	Public	Can Modify State	onlyOwner

PontoonBridge			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
swap	External	Can Modify State	nonReentrant
redeem	External	Can Modify State	nonReentrant
addPool	External	Can Modify State	onlyOwner
updatePool	External	Can Modify State	onlyOwner
getAllTokens	External	-	-
getLiquidityPool	External	-	-

PontoonBridge			
getSwapState	External	-	-
updateValidator	External	Can Modify State	onlyOwner
updateLockPeriod	Public	Can Modify State	onlyOwner
updateLockPeriodForPool	Public	Can Modify State	onlyOwner
updateFee	External	Can Modify State	onlyOwner

PontoonPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
transferLiquidity	External	Can Modify State	onlyBridge
addLiquidity	External	Can Modify State	-
removeLiquidity	External	Can Modify State	nonReentrant
updateAccruedFee	External	Can Modify State	onlyBridge
getUnlockTime	External	-	-
updateLockPeriod	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event records

Category: Others

Content

- In the PontoonTokenVesting contract, the owner role can update the start time through the updateStartTime function, but the event is not recorded.
- In the PontoonPool contract, the owner can modify the lockPeriod parameter of the contract through the updateLockPeriod function, and the Bridge can modify the accruedFee parameter of the contract through the updateAccruedFee function. But no event recording.
- In the PontoonBridg contract, the owner can modify the validator, lockPeriod and fee parameters through the eupdateValidator function, updateLockPeriod function and updateFee function, respectively. But no event recording.

Code location:

- pontoon-token/contracts/PontoonTokenVesting.sol

```
function updateStartTime(uint256 _startAfter) external override onlyOwner() {
    require(_startAfter > 0, "Invalid startTime");
    require(block.timestamp < startTime, "Already started");

    uint256 _startTime = block.timestamp + _startAfter;

    _massUpdateCliffEndTime(_startTime);

    startTime = _startTime;
}
```

- poontoon/contracts/PontoonPool.sol

```
function updateAccruedFee(uint256 _fee) external onlyBridge {
    accruedFee += _fee;
}
function updateLockPeriod(uint256 _lockPeriod) external onlyOwner {
    lockPeriod = _lockPeriod;
}
```

- poontoon/contracts/PontoonBridge.sol

```
    function updateValidator(address _validator) external onlyOwner {
        require(_validator != address(0), "Bridge: validator address is zero 
address");
        validator = _validator;
    }
    function updateLockPeriod(uint256 _lockPeriod) public onlyOwner {
        lockPeriod = _lockPeriod;
    }
    function updateFee(uint256 _fee) external onlyOwner {
        require(_fee <= (10**feePrecision), "Bridge: fee cannot be larger than fee 
precision");
        fee = _fee;
    }
```

Solution

It is recommended to record events when updating the contract's sensitive parameters for follow-up self-examination or community review.

Status

Fixed

[N2] [Low] Check time issue

Category: Design Logic Audit

Content

In the PontoonTokenVesting contract, the owner can add an investor through the addOrUpdateInvestor function and the addOrUpdateInvestors function, but it does not check whether the current time is less than cliffEndTime when adding the investor. If investor is added after cliffEndTime, it may cause the issue of early release of tokens or repeated release of tokens.

Code location: pontoon-token/contracts/PontoonTokenVesting.sol

```
    function addOrUpdateInvestor(
        RoundType _roundType,
        address _investor,
        uint256 _amount
    ) external override onlyOwner() {
        _addInvestor(_roundType, _investor, _amount);

        emit InvestorAdded(_roundType, _investor, _amount);
    }
    function addOrUpdateInvestors(
        RoundType[] memory _roundType,
        address[] memory _investors,
        uint256[] memory _amount
    ) external override onlyOwner() {
        uint256 length = _roundType.length;

        require(_investors.length == length && _amount.length == length, "Arguments
length not match");

        for (uint256 i = 0; i < length; i++) {
            _addInvestor(_roundType[i], _investors[i], _amount[i]);
        }

        emit InvestorsAdded(_roundType, _investors, _amount);
    }
```

Solution

If it is not designed as expected, it is recommended to check whether the current time is less than cliffEndTime when adding investor.

Status

Confirmed

[N3] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the PontoonTokenVesting contract, the owner can withdraw any token in the contract through the recoverToken function. However, the note of this function indicates that the expected design does not allow the withdrawal of toon tokens. Therefore, the current actual design situation does not match expectations.

Code location: pontoon-token/contracts/PontoonTokenVesting.sol

```
    /**
     * @notice recover any erc20 token (ex - toon token)
     */
    function recoverToken(address _token, uint256 amount) external override onlyOwner
{
        IERC20(_token).safeTransfer(_msgSender(), amount);
        emit RecoverToken(_token, amount);
    }
```

Solution

It is recommended to check in the recoverToken function that the passed `_token` parameter is not equal to toonToken.

Status

Confirmed

[N4] [Suggestion] Compatibility issues

Category: Design Logic Audit

Content

In the PontoonPool contract, the user can add liquidity through the addLiquidity function, which will assign the value of the `liquidity[msg.sender].lpTokenBalance` according to the `_amount` parameter value passed by the user, and then transfer the tokens to this contract through the safeTransferFrom function. But if the token is a deflationary token, the actual number of tokens received by the contract will be different from the value recorded by `lpTokenAmount`.

The same is true for the swap function of the PontoonBridge contract.

The same is true for the stake function of the PontoonFarm contract.

Code location: pontoon/contracts/PontoonPool.sol

```
    function addLiquidity(uint256 _amount) external {
        uint256 lpTokenAmount = _amount * (10**factor);

        liquidity[msg.sender].lpTokenBalance += lpTokenAmount;
        liquidity[msg.sender].unlockTime = block.timestamp + lockPeriod;

        IERC20(token).safeTransferFrom(msg.sender, address(this), _amount);
        _mint(msg.sender, lpTokenAmount);
    }
```

Solution

If PontoonPool will receive deflationary tokens in the future, we recommend using the contract balance difference before and after the user transfer to record lpTokenAmount to avoid compatibility risks.

Status

Fixed

[N5] [Suggestion] Does not comply with the Checks-Effects-Interactions principle

Category: Others

Content

In the PontoonPool contract, the user can add liquidity through the addLiquidity function, but it will first change the user's lpTokenBalance state and unlockTime state before performing the transfer operation. This does not comply with the Checks-Effects-Interactions principle.

Code location: pontoon/contracts/PontoonPool.sol

```
    function addLiquidity(uint256 _amount) external {
        uint256 lpTokenAmount = _amount * (10**factor);

        liquidity[msg.sender].lpTokenBalance += lpTokenAmount;
        liquidity[msg.sender].unlockTime = block.timestamp + lockPeriod;

        IERC20(token).safeTransferFrom(msg.sender, address(this), _amount);
        _mint(msg.sender, lpTokenAmount);
    }
```

Solution

It is recommended to transfer funds before changing the status when users add liquidity.

Status

Fixed

[N6] [Low] Access control issues

Category: Authority Control Vulnerability

Content

In the PontoonBridge contract, the owner can call the updateLockPeriod function of the PontoonPool contract through the updateLockPeriodForPool function to modify the lockPeriod parameter. However, the updateLockPeriod function of the PontoonPool contract uses the onlyOwner modifier, which does not match expectations.

Code location: pontoon/contracts/PontoonBridge.sol

```
function updateLockPeriodForPool(address _token, uint256 _lockPeriod)
    public
    onlyOwner
{
    PontoonPool liquidityPool = liquidityPools[_token];
    PontoonPool(liquidityPool).updateLockPeriod(_lockPeriod);
}

function updateLockPeriod(uint256 _lockPeriod) external onlyOwner {
    lockPeriod = _lockPeriod;
}
```

Solution

It is recommended that the PontoonPool contract use the onlyBridge modifier.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002109070001	SlowMist Security Team	2021.08.30 - 2021.09.07	Low 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 3 low risk, 3 suggestion vulnerabilities. And 2 low risk vulnerabilities were confirmed and being fixed; All other 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.