



Smart Contract Security Audit Report



Table Of Contents

| | |
|-------------------------------|-------|
| 1 Executive Summary | _____ |
| 2 Audit Methodology | _____ |
| 3 Project Overview | _____ |
| 3.1 Project Introduction | _____ |
| 3.2 Vulnerability Information | _____ |
| 4 Code Overview | _____ |
| 4.1 Contracts Description | _____ |
| 4.2 Visibility Description | _____ |
| 4.3 Vulnerability Summary | _____ |
| 5 Audit Result | _____ |
| 6 Statement | _____ |

1 Executive Summary

On 2024.10.08, the SlowMist security team received the Plume Network team's security audit application for Plume Network Staking, 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

This is the staking contract component of Plume Network, and the audit mainly covers the RWASTaking and SBTCTaking contracts. Users can stake stablecoins in the RWASTaking contract or stake SBTC in the SBTCTaking contract to earn rewards. This is a pre-staking contract. When the main staking contract is deployed (possibly on other chains), the admin role of the pre-staking contract will migrate users' staked tokens to the main staking contract in a centralized manner.

3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
|----|--|---------------------------------------|------------|--------------|
| N1 | Potential Token Compatibility Issues | Design Logic Audit | Suggestion | Fixed |
| N2 | Risks of excessive privilege | Authority Control Vulnerability Audit | Medium | Acknowledged |
| N3 | Potential risks of staking stablecoins of different decimals | Design Logic Audit | Critical | Fixed |

4 Code Overview

4.1 Contracts Description

Audit Version:

<https://github.com/plumenetwork/contracts>

commit: 7acc75ae043648293e01da309eb23e6f5cae34dc

Fixed Version:

<https://github.com/plumenetwork/contracts>

commit: 65a08362968b2bcf669bfbb8efd8d1eca51e5aa2

Audit scope:

- staking/src/RWASTaking.sol
- staking/src/SBTCStaking.sol

The main network address of the contract is as follows:

| Contract Name | Contract Address | Chain |
|-------------------------|--|----------|
| PlumePreStaking (Proxy) | 0xdbd03D676e1cf3c3b656972F88eD21784372AcAB | Ethereum |
| RWASTaking (Impl) | 0x2eA61aA006B79606baFc0aE6870D7FFb22411289 | Ethereum |

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

| RWASTaking | | | |
|-----------------------|------------|------------------|-------------|
| Function Name | Visibility | Mutability | Modifiers |
| _getRWASTakingStorage | Private | - | - |
| <Constructor> | Public | Can Modify State | - |
| initialize | Public | Can Modify State | initializer |
| _authorizeUpgrade | Internal | Can Modify State | onlyRole |
| allowStablecoin | External | Can Modify State | onlyRole |
| withdraw | External | Can Modify State | onlyRole |
| stake | External | Can Modify State | - |
| getTotalAmountStaked | External | - | - |
| getUsers | External | - | - |
| getUserState | External | - | - |
| getAllowedStablecoins | External | - | - |
| isAllowedStablecoin | External | - | - |
| getEndTime | External | - | - |

| SBTCStaking | | | |
|------------------------|------------|------------------|-------------|
| Function Name | Visibility | Mutability | Modifiers |
| _getSBTCStakingStorage | Private | - | - |
| <Constructor> | Public | Can Modify State | - |
| initialize | Public | Can Modify State | initializer |
| _authorizeUpgrade | Internal | Can Modify State | onlyRole |
| withdraw | External | Can Modify State | onlyRole |

| SBTCStaking | | | |
|----------------------|----------|------------------|---|
| stake | External | Can Modify State | - |
| getSBTC | External | - | - |
| getTotalAmountStaked | External | - | - |
| getUsers | External | - | - |
| getUserState | External | - | - |
| getEndTime | External | - | - |

4.3 Vulnerability Summary

[N1] [Suggestion] Potential Token Compatibility Issues

Category: Design Logic Audit

Content

In the RWASTaking contract, the admin can set the whitelist stablecoins that can be staked through the allowStablecoin function. Users can stake whitelist stablecoins through the stake function to obtain rewards. The contract will transfer the user-specified staking amount into the Staking contract through the safeTransferFrom function and update the user's staking status. However, it does not check whether the actual amount of tokens received by the contract matches the expected amount. It should be noted that some stablecoins (such as USDT) support charging fees during transfers, which means that the actual amount of tokens received by the contract during a stablecoin transfer may be less than expected.

Despite this, the possibility of these stablecoins enabling the transfer fee function in a short period of time is extremely low. However, when performing the allowStablecoin operation, it is still necessary to carefully identify fee-on-transfer stablecoins.

Code location: staking/src/RWASTaking.sol#L188-L198

```
function stake(uint256 amount, IERC20 stablecoin) external {
    ...

    stablecoin.safeTransferFrom(msg.sender, address(this), amount);
}
```



```

uint256 timestamp = block.timestamp;
UserState storage userState = $.userStates[msg.sender];
if (userState.lastUpdate == 0) {
    $.users.push(msg.sender);
}
userState.amountSeconds += userState.amountStaked * (timestamp -
userState.lastUpdate);
userState.amountStaked += amount;
userState.lastUpdate = timestamp;
$.totalAmountStaked += amount;

emit Staked(msg.sender, stablecoin, amount, timestamp);
}

```

Solution

If possible, it is recommended to use the difference in the contract's token balance before and after the transfer as the actual amount of tokens received by the contract when the user stakes.

Status

Fixed; Fixed in commit 70e2c223a57b5ba421b21b8bfee5bfd627626a15

[N2] [Medium] Risks of excessive privilege

Category: Authority Control Vulnerability Audit

Content

Users can stake their Stablecoins and SBTC tokens through the pre-staking whitelist in the staking contract to earn rewards. However, when the main staking contract is deployed (possibly on other chains), the admin role will withdraw all tokens from the staking contract using the withdraw function and migrate users' staked tokens to the main staking contract in a centralized manner. This poses the risk of excessive privileges.

Code location:

staking/src/RWAStaking.sol#L155

```

function withdraw() external onlyRole(ADMIN_ROLE) {
    ...
}

```

staking/src/SBTCStaking.sol#L128

```
function withdraw() external onlyRole(ADMIN_ROLE) {
    ...
}
```

Solution

It is recommended to use a multi-signature wallet to manage the admin role to avoid single-point-of-failure risk, but this cannot mitigate the risk of excessive privileges. In the long run, management through community governance can effectively alleviate the risk of excessive privileges and increase user trust.

Status

Acknowledged; The project team has transferred the withdrawal permissions of the RWASTaking contract to the Timelock contract (0x0f5f5515b768C14b55934924ED322c43773308F9), with a multisig wallet (0xa472f6bDf1E676C7B773591d5D820aDC27a2D51c) managing the execution rights of the Timelock contract.

[N3] [Critical] Potential risks of staking stablecoins of different decimals

Category: Design Logic Audit

Content

In the RWASTaking contract, if multiple supported stablecoins have different decimals, then amountStaked will record total staked amounts with inconsistent decimals. This will lead to incorrect calculations of amountSeconds.

Code location: staking/src/RWASTaking.sol#L308

```
function stake(uint256 amount, IERC20 stablecoin) external nonReentrant {
    ...

    uint256 previousBalance = stablecoin.balanceOf(address(this));

    stablecoin.safeTransferFrom(msg.sender, address(this), amount);

    uint256 newBalance = stablecoin.balanceOf(address(this));
    uint256 actualAmount = newBalance - previousBalance;
    ...
    userState.amountSeconds += userState.amountStaked * (timestamp -
userState.lastUpdate);
    ...
}
```

Solution

If support for multiple stablecoins with different decimals is required, their decimals should be normalized before updating amountSeconds, amountStaked, and totalAmountStaked.

Status

Fixed

5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
|----------------|------------------------|-------------------------|--------------|
| 0X002410090001 | SlowMist Security Team | 2024.10.08 - 2024.10.09 | Low Risk |

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical risk, 1 medium risk, and 1 suggestion. All the findings were acknowledged or fixed. The code was not deployed to the mainnet. Currently, the withdrawal permissions in the RWASTaking contract are managed by the timelock contract, with multisig contract controlling the execution rights of the timelock. Therefore, the overall contract risk is low.

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](https://twitter.com/SlowMist_Team)



Github
<https://github.com/slowmist>