Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.11.01, the SlowMist security team received the FantomStarter.io team's security audit application for FantomStarter, 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.

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/fantomstarterio/contracts/blob/main/contracts/StakerTierFantomStarter.sol

commit: d0c05b4966065f9eacdda76035503427376bf308.

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	Confirming
N2	Dev address setting enhancement suggestions	Others	Suggestion	Confirming

NO	Title	Category	Level	Status
N3	Decimals loss issue	Others	Suggestion	Confirming

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:

StakerTierFantomStarter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
supportsInterface	Public	-	-
setTierFactoryAddress	Public	Can Modify State	isAdmin
setLockupDays	Public	Can Modify State	isAdmin
setCoolDownDays	Public	Can Modify State	isAdmin
setDevWalletAddress	Public	Can Modify State	isAdmin
setWithholdUnstakingPercentage	Public	Can Modify State	isAdmin

StakerTierFantomStarter			
getNFTIdEligibleForBenefits	Public	-	-
getTokenAmountStakedForBenefits	Public	-	-
getLockupTimeLeftStakedNFT	Public	-	-
getLockupTimeLeftStakedFSTokens	Public	-	-
hasWalletCompletedStakingNFT	Public	-	-
hasWalletCompletedStakingFSTokens	Public	-	-
hasWalletCurrentlyStakedNFT	Public	-	-
hasWalletCurrentlyStakedFSTokens	Public	-	-
hasCancelCoolDownDaysExpiredForStakedNFT	Public	-	-
stakeTierNFT	Public	Can Modify State	-
stakeTierFSTokens	Public	Can Modify State	-
updateCurrentStakedFSTokens	Public	Can Modify State	-
unstakeTierNFT	Public	Can Modify State	-
unstakeTierFSTokensAfterLockupPeriod	Public	Can Modify State	-
unstakeTierFSTokensBeforeLockupPeriod	Public	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event records

Category: Others

Content

In the contract, the Admin role can set tierFactory lockupDays, coolDownDays, devWallet, and withholdUnstakingPercentage through the setTierFactoryAddress, setLockupDays, setCoolDownDays, setDevWalletAddress and setWithholdUnstakingPercentage function, but no event logging is performed.

Code location：

/fantomstarterio/contracts/contracts/StakerTierFantomStarter.sol#L1273-1306

```
    function setTierFactoryAddress(ITierFactoryFantomStarter _tierFactory) public
isAdmin {
        tierFactory = _tierFactory;
    }

    /**
    * @dev Replace the lockup_days period, only admins.
    **/
    function setLockupDays(uint256 _days) public isAdmin {
        lockupDays = _days;
    }

    /**
    * @dev Replace the cooldowndays period, only admins.
    **/
    function setCoolDownDays(uint256 _days) public isAdmin {
        coolDownDays = _days;
    }

    /**
    * @dev Replace the dev wallet
    **/
    function setDevWalletAddress(address _devWallet) public isAdmin {
        devWallet = _devWallet;
    }

    /**
    * @dev Replace the withhold unstaking percentage
    **/
    function setWithholdUnstakingPercentage(uint256 _withholdUnstakingPercentage)
public isAdmin {
        require(_withholdUnstakingPercentage < 100, "You cannot withhold more than 99
percent");
        require(_withholdUnstakingPercentage >= 1, "You need to withhold at least 1
percent");

        withholdUnstakingPercentage = _withholdUnstakingPercentage;
    }
```

Solution

It is recommended to record events when modifying sensitive parameters.

Status

Confirming

[N2] [Suggestion] Dev address setting enhancement suggestions

Category: Others

Content

If the dev address is an EOA address, in a scenario where the private key is leaked, the team's revenue will be stolen.

And no event logging is performed.

Code location：

/fantomstarterio/contracts/contracts/StakerTierFantomStarter.sol#L1294-1296

```
    function setDevWalletAddress(address _devWallet) public isAdmin {
        devWallet = _devWallet;
    }
```

Solution

It is recommended to set the development address as a multi-signature contract to avoid the leakage of private keys and the theft of team rewards. And it is recommended to record events when modifying sensitive parameters.

Status

Confirming

[N3] [Suggestion] Decimals loss issue

Category: Others

Content

In the unstakeTierFSTokensBeforeLockupPeriodfuntion, when calculating _fsTokensToReturnToStaker and _fsTokensPenaltyToDevWallet, it will be divided by 100 at the end, resulting in a loss of decimals to the _fsTokensToReturnToStaker from users.

Code location:

/fantomstarterio/contracts/contracts/StakerTierFantomStarter.sol#L1555-1579

```
    function unstakeTierFSTokensBeforeLockupPeriod() public {
        require(hasWalletCurrentlyStakedFSTokens(msg.sender) == true, "You have no
FSTokens staked");
        require(hasWalletCompletedStakingFSTokens(msg.sender) == false, "Lockup
period has expired");

        // 1. get the staked NFT object
        StakedTierFSToken memory _stakedTierFSToken =
walletStakedTierFSTokenList[msg.sender];

        uint256 _fsTokensToReturnToStaker = 0;
        uint256 _fsTokensPenaltyToDevWallet = 0;

        // 2. Set the tokens that need to be returned
        uint256 _percentageToStaker = 100 -
_stakedTierFSToken.withholdUnstakingPercentage;
        _fsTokensToReturnToStaker = _stakedTierFSToken.tokenAmount *
_percentageToStaker / 100;
        _fsTokensPenaltyToDevWallet = _stakedTierFSToken.tokenAmount *
_stakedTierFSToken.withholdUnstakingPercentage / 100;

        // 3. Safe transfer the tokens to the dev wallet and the staker
        fsToken.safeTransfer(msg.sender, _fsTokensToReturnToStaker);
        fsToken.safeTransfer(devWallet, _fsTokensPenaltyToDevWallet);

        // 4. Remove the StakedTierFSToken Object from the mapping
        delete walletStakedTierFSTokenList[msg.sender];

        // Emit event
        emit UnstakedBeforeLockupPeriodFSToken(msg.sender, _fsTokensToReturnToStaker,
_fsTokensPenaltyToDevWallet);
    }
```

Solution

It’s recommended to use (_stakedTierFSToken.tokenAmount - _fsTokensPenaltyToDevWallet) to calculate the _fsTokensToReturnToStaker to reduce the decimals loss.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002111050002	SlowMist Security Team	2021.11.01 - 2021.11.05	Passed

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 suggestion vulnerabilities. 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.