Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.09.16, the SlowMist security team received the Wombex Finance team's security audit application for Wombex Finance, 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:

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

Serial Number	Audit Class	Audit Subclass
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
		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

Project: Wombex Finance

Type: DeFi

Module: Reward Module + Locker + Drop + Staking + Booster

Audit Version

https://github.com/wombex-finance/wombex-contracts

commit:eb94985302434f19f465c4f1f8ad25587b495c45

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Compatibility issue	Others	Suggestion	Fixed
N2	Donate Logic Issue	Design Logic Audit	Low	Ignored
N3	Missing error checking	Others	Suggestion	Fixed
N4	Incident records are not accurate	Malicious Event Log Audit	Suggestion	Fixed
N5	Risk of excessive authority	Authority Control Vulnerability	Low	Ignored
N6	Variable without assignment	Uninitialized Storage Pointers Vulnerability	Medium	Fixed
N7	Token transfer reminder	Others	Suggestion	Fixed
N8	Event log missing	Malicious Event Log Audit	Suggestion	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:

BaseRewardPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
totalSupply	Public	-	-
balanceOf	Public	-	-
lastTimeRewardApplicable	Public	-	-
rewardPerToken	Public	-	-
earned	Public	-	-
stake	Public	Can Modify State	-
stakeAll	External	Can Modify State	-
stakeFor	Public	Can Modify State	-
_processStake	Internal	Can Modify State	updateReward
withdraw	Public	Can Modify State	updateReward

BaseRewardPool			
withdrawAll	External	Can Modify State	-
withdrawAndUnwrap	Public	Can Modify State	-
_withdrawAndUnwrapTo	Internal	Can Modify State	updateReward
withdrawAllAndUnwrap	External	Can Modify State	-
getReward	Public	Can Modify State	updateReward
getReward	External	Can Modify State	-
donate	External	Can Modify State	-
processIdleRewards	External	Can Modify State	-
queueNewRewards	External	Can Modify State	-
_notifyRewardAmount	Internal	Can Modify State	updateReward
_lastTimeRewardApplicable	Internal	-	-
_earned	Internal	-	-
_rewardPerToken	Internal	-	-

BaseRewardPool4626			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseRewardPool
totalAssets	External	-	-
deposit	Public	Can Modify State	nonReentrant
mint	External	Can Modify State	-

BaseRewardPool4626			
withdraw	Public	Can Modify State	nonReentrant
redeem	External	Can Modify State	-
convertToShares	Public	-	-
convertToAssets	Public	-	-
maxDeposit	Public	-	-
previewDeposit	External	-	-
maxMint	External	-	-
previewMint	External	-	-
maxWithdraw	Public	-	-
previewWithdraw	Public	-	-
maxRedeem	External	-	-
previewRedeem	External	-	-
name	External	-	-
symbol	External	-	-
decimals	External	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	External	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-

BaseRewardPool4626			
_approve	Internal	Can Modify State	-
transferFrom	External	Can Modify State	-

Booster			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setOwner	External	Can Modify State	-
setFeeManager	External	Can Modify State	-
setPoolManager	External	Can Modify State	-
setFactories	External	Can Modify State	-
setExtraRewardsDistributor	External	Can Modify State	-
setRewardClaimedPenalty	External	Can Modify State	-
setVoteDelegate	External	Can Modify State	-
setVotingValid	External	Can Modify State	-
setLockRewardContracts	External	Can Modify State	-
setMintRatio	External	Can Modify State	-
updateDistributionByTokens	External	Can Modify State	-
setEarmarkIncentive	External	Can Modify State	-
addPool	External	Can Modify State	-
shutdownPool	External	Can Modify State	-

Booster			
shutdownSystem	External	Can Modify State	-
deposit	Public	Can Modify State	-
depositAll	External	Can Modify State	-
_withdraw	Internal	Can Modify State	-
withdraw	Public	Can Modify State	-
withdrawAll	Public	Can Modify State	-
withdrawTo	External	Can Modify State	-
setVote	External	Can Modify State	-
voteExecute	External	Payable	-
_earmarkRewards	Internal	Can Modify State	-
earmarkRewards	External	Can Modify State	-
rewardClaimed	External	Can Modify State	-
poolLength	External	-	-
distributionByTokenLength	External	-	-
distributionTokenList	External	-	-

CvxCrvToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
setOperator	External	Can Modify State	-

CvxCrvToken			
mint	External	Can Modify State	-
burn	External	Can Modify State	-

DepositToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	-
burn	External	Can Modify State	-

RewardFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
CreateCrvRewards	External	Can Modify State	-

TokenFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
CreateDepositToken	External	Can Modify State	-

VoterProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

VoterProxy			
<Receive Ether>	External	Payable	-
getName	External	-	-
setOwner	External	Can Modify State	-
setLpTokensPid	External	Can Modify State	-
setRewardDeposit	External	Can Modify State	-
setOperator	External	Can Modify State	-
setDepositor	External	Can Modify State	-
setVote	External	Can Modify State	-
isValidSignature	Public	-	-
deposit	External	Can Modify State	-
lock	External	Can Modify State	-
releaseLock	External	Can Modify State	-
withdraw	External	Can Modify State	-
withdrawLp	Public	Can Modify State	-
withdrawAllLp	External	Can Modify State	-
_withdrawSomeLp	Internal	Can Modify State	-
claimCrv	External	Can Modify State	-
balanceOfPool	Public	-	-
execute	External	Payable	-

ExtraRewardsDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
modifyWhitelist	External	Can Modify State	onlyOwner
addReward	External	Can Modify State	-
addRewardToEpoch	External	Can Modify State	-
_addReward	Internal	Can Modify State	nonReentrant
getReward	Public	Can Modify State	-
getReward	Public	Can Modify State	-
_getReward	Internal	Can Modify State	nonReentrant
forfeitRewards	External	Can Modify State	-
claimableRewards	External	-	-
claimableRewardsAtEpoch	External	-	-
_allClaimableRewards	Internal	-	-
_claimableRewards	Internal	-	-
rewardEpochsCount	External	-	-

Wmx			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
init	External	Can Modify State	-

Wmx			
updateOperator	Public	Can Modify State	-
mint	External	Can Modify State	-
minterMint	External	Can Modify State	-

WmxClaimZap			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getName	External	-	-
setApprovals	External	Can Modify State	-
_checkOption	Internal	-	-
claimRewards	External	Can Modify State	-
_claimExtras	Internal	Can Modify State	-

WmxLocker			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
modifyBlacklist	External	Can Modify State	onlyOwner
addReward	External	Can Modify State	onlyOwner
approveRewardDistributor	External	Can Modify State	onlyOwner
setKickIncentive	External	Can Modify State	onlyOwner
shutdown	External	Can Modify State	onlyOwner

WmxLocker			
recoverERC20	External	Can Modify State	onlyOwner
setApprovals	External	Can Modify State	-
lock	External	Can Modify State	nonReentrant updateReward
_lock	Internal	Can Modify State	notBlacklisted
getReward	External	Can Modify State	-
getReward	Public	Can Modify State	nonReentrant updateReward
getReward	External	Can Modify State	nonReentrant updateReward
checkpointEpoch	External	Can Modify State	-
_checkpointEpoch	Internal	Can Modify State	-
processExpiredLocks	External	Can Modify State	nonReentrant
kickExpiredLocks	External	Can Modify State	nonReentrant
emergencyWithdraw	External	Can Modify State	nonReentrant
_processExpiredLocks	Internal	Can Modify State	updateReward
delegate	External	Can Modify State	nonReentrant
_checkpointDelegate	Internal	Can Modify State	-
delegates	Public	-	-
getVotes	External	-	-
checkpoints	External	-	-
numCheckpoints	External	-	-
getPastVotes	Public	-	-

WmxLocker			
getPastTotalSupply	External	-	-
_checkpointsLookup	Private	-	-
balanceOf	External	-	-
balanceAtEpochOf	Public	-	-
lockedBalances	External	-	-
totalSupply	External	-	-
totalSupplyAtEpoch	Public	-	-
findEpochId	Public	-	-
epochCount	External	-	-
decimals	External	-	-
name	External	-	-
symbol	External	-	-
claimableRewards	External	-	-
lastTimeRewardApplicable	External	-	-
rewardPerToken	External	-	-
_earned	Internal	-	-
_lastTimeRewardApplicable	Internal	-	-
_rewardPerToken	Internal	-	-
queueNewRewards	External	Can Modify State	nonReentrant
_notifyReward	Internal	Can Modify State	updateReward

WmxLocker			
rewardTokensLen	External	-	-
rewardTokensList	External	-	-

WmxMerkleDrop			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setDao	External	Can Modify State	-
setRoot	External	Can Modify State	-
startEarly	External	Can Modify State	-
withdrawExpired	External	Can Modify State	-
setLocker	External	Can Modify State	-
rescueReward	Public	Can Modify State	-
claim	Public	Can Modify State	-

WmxMinter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
mint	External	Can Modify State	

WmxPenaltyForwarder			
Function Name	Visibility	Mutability	Modifiers

WmxPenaltyForwarder			
<Constructor>	Public	Can Modify State	Ownable
forward	Public	Can Modify State	-
setDistributor	Public	Can Modify State	onlyOwner

WmxRewardPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
totalSupply	Public	-	-
balanceOf	Public	-	-
lastTimeRewardApplicable	Public	-	-
rewardPerToken	Public	-	-
earned	Public	-	-
stake	Public	Can Modify State	updateReward
stakeAll	External	Can Modify State	-
stakeFor	Public	Can Modify State	updateReward
withdraw	Public	Can Modify State	updateReward
getReward	Public	Can Modify State	updateReward
forwardPenalty	Public	Can Modify State	-
rescueReward	Public	Can Modify State	-
setLocker	External	Can Modify State	-

WmxRewardPool			
initialiseRewards	External	Can Modify State	-

WmxVestedEscrow			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setAdmin	External	Can Modify State	-
setLocker	External	Can Modify State	-
fund	External	Can Modify State	nonReentrant
cancel	External	Can Modify State	nonReentrant
available	Public	-	-
remaining	Public	-	-
_totalVestedOf	Internal	-	-
claim	External	Can Modify State	nonReentrant
_claim	Internal	Can Modify State	-

WomDepositor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setLockConfig	External	Can Modify State	onlyOwner
setCustomLock	External	Can Modify State	onlyOwner
deposit	External	Can Modify State	-

WomDepositor			
deposit	External	Can Modify State	-
deposit	Public	Can Modify State	-
_smartLock	Internal	Can Modify State	-
depositCustomLock	Public	Can Modify State	-
releaseCustomLock	Public	Can Modify State	-
getCustomLockSlotsLength	Public	-	-

WomStakingProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setConfig	External	Can Modify State	onlyOwner
setApprovals	External	Can Modify State	-
rescueToken	External	Can Modify State	onlyOwner
queueNewRewards	External	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Suggestion] Compatibility issue

Category: Others

Content

- contracts/vendor/BaseRewardPool.sol

In the `BaseRewardPool` contract, the `stake`, `stakeFor`, `donate`, `queueNewRewards` function is used to transfer the tokens into this contract, but the `BaseRewardPool` contract does not check how many tokens the contract actually received. If `BaseRewardPool` lists deflationary tokens, the tokens transferred through the the `stake`, `stakeFor`, `donate`, `queueNewRewards` function do not match the tokens actually received by the contract. This will lead to a bias in the calculation of the number of tokens.

```
function stake(uint256 _amount)
    public
    returns(bool)
{
    _processStake(_amount, msg.sender);

    stakingToken.safeTransferFrom(msg.sender, address(this), _amount);
    emit Staked(msg.sender, _amount);

    return true;
}
function stakeFor(address _for, uint256 _amount)
    public
    returns(bool)
{
    _processStake(_amount, _for);

    //take away from sender
    stakingToken.safeTransferFrom(msg.sender, address(this), _amount);
    emit Staked(_for, _amount);

    return true;
}

function donate(address _token, uint256 _amount) external returns(bool){
        IERC20(_token).safeTransferFrom(msg.sender, address(this), _amount);
        tokenRewards[_token].queuedRewards =
tokenRewards[_token].queuedRewards.add(_amount);
    }
    function queueNewRewards(address _token, uint256 _rewards) external returns(bool)
{
        require(msg.sender == operator, "!authorized");

        IERC20(_token).safeTransferFrom(msg.sender, address(this), _rewards);
```

```
        RewardState storage rState = tokenRewards[_token];
        if (rState.lastUpdateTime == 0) {
            rState.token = _token;
            allRewardTokens.push(_token);
            require(allRewardTokens.length <= MAX_TOKENS, "!`max_tokens`");
        }
        _rewards = _rewards.add(rState.queuedRewards);

        if (block.timestamp >= rState.periodFinish) {
            _notifyRewardAmount(rState, _rewards);
            rState.queuedRewards = 0;
            return true;
        }

        //et = now - (finish-duration)
        uint256 elapsedTime = block.timestamp.sub(rState.periodFinish.sub(DURATION));
        //current at now: rewardRate * elapsedTime
        uint256 currentAtNow = rState.rewardRate * elapsedTime;
        uint256 queuedRatio = currentAtNow.mul(1000).div(_rewards);
        //uint256 queuedRatio = currentRewards.mul(1000).div(_rewards);
        if(queuedRatio < NEW_REWARD_RATIO){
            _notifyRewardAmount(rState, _rewards);
            rState.queuedRewards = 0;
        }else{
            rState.queuedRewards = _rewards;
        }
        return true;
    }
```

- contracts/vendor/Booster.sol

In the `Booster` contract, the `deposit`, function is used to transfer the tokens into this contract, but the `Booster` contract does not check how many tokens the contract actually received. If `Booster` lists deflationary tokens, the tokens transferred through the the `deposit`, function do not match the tokens actually received by the contract. This will lead to a bias in the calculation of the number of tokens.

```
    function deposit(uint256 _pid, uint256 _amount, bool _stake) public returns(bool)
{
```

```
        require(!isShutdown,"shutdown");
        PoolInfo storage pool = poolInfo[_pid];
        require(pool.shutdown == false, "pool is closed");

        //send to proxy to stake
        address lptoken = pool.lptoken;
        IERC20(lptoken).safeTransferFrom(msg.sender, voterProxy, _amount);

        //stake
        address gauge = pool.gauge;
        require(gauge != address(0),"!gauge setting");
        IStaker(voterProxy).deposit(lptoken, gauge);

        address token = pool.token;
        if(_stake){
            //mint here and send to rewards on user behalf
            ITokenMinter(token).mint(address(this), _amount);
            IRewards(pool.crvRewards).stakeFor(msg.sender, _amount);
        }else{
            //add user balance directly
            ITokenMinter(token).mint(msg.sender, _amount);
        }

        emit Deposited(msg.sender, _pid, _amount);
        return true;
    }
```

Solution

Check the token balance difference before and after the transfer when performing the token transfer operation to this contract.

Status

Fixed; fixed in commit 6b59eba385959d1a0067471acd28f79e5e097d41.

[N2] [Low] Donate Logic Issue

Category: Design Logic Audit

Content

- contracts/vendor/BaseRewardPool.sol

If the `donate` function is added in the `allRewardTokens` list, it should have the same verification execution logic as `queueNewRewards`, otherwise there will be a problem of inaccurate rewards.

```
function donate(address _token, uint256 _amount) external returns(bool){
        IERC20(_token).safeTransferFrom(msg.sender, address(this), _amount);
        tokenRewards[_token].queuedRewards =
tokenRewards[_token].queuedRewards.add(_amount);
    }
```

Solution

It should have the same verification execution logic as `queueNewRewards`.

Status

Ignored; Donations will wait in queue until Booster will call queueNewRewards().

[N3] [Suggestion] Missing error checking

Category: Others

Content

- contracts/vendor/Booster.sol

catch unhandled exception.

```
function shutdownPool(uint256 _pid) external returns(bool){
    require(msg.sender==poolManager, "!auth");
    PoolInfo storage pool = poolInfo[_pid];

    //withdraw from gauge
    try IStaker(voterProxy).withdrawAllLp(pool.lptoken,pool.gauge){
    }catch{}

    pool.shutdown = true;

    emit PoolShutdown(_pid);
    return true;
```

```
}

function shutdownSystem() external{
    require(msg.sender == owner, "!auth");
    isShutdown = true;

    for(uint i=0; i < poolInfo.length; i++){
        PoolInfo storage pool = poolInfo[i];
        if (pool.shutdown) continue;

        address token = pool.lptoken;
        address gauge = pool.gauge;

        //withdraw from gauge
        try IStaker(voterProxy).withdrawAllLp(token,gauge){
            pool.shutdown = true;
        }catch{}
    }
}
```

Solution

Handle exceptions.

Status

Fixed

[N4] [Suggestion] Incident records are not accurate

Category: Malicious Event Log Audit

Content

- contracts/vendor/Booster.sol

The event logged by `rewardFactory` when not `address(0)` is wrong.

```
function setFactories(address _rfactory, address _tfactory) external {
    require(msg.sender == owner, "!auth");

    //reward factory only allow this to be called once even if owner
```

```
    //removes ability to inject malicious staking contracts
    //token factory can also be immutable
    if(rewardFactory == address(0)){
        rewardFactory = _rfactory;
        tokenFactory = _tfactory;

        emit FactoriesUpdated(_rfactory, _tfactory);
    } else {
        emit FactoriesUpdated(address(0), address(0));
    }
}
```

Solution

Log the correct event.

Status

Fixed; fixed in commit 5c958930e434c10ca2039d8adec3e67fa7320ea0.

[N5] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

- contracts/vendor/Booster.sol
- contracts/vendor/VoterProxy.sol

The `owner` role can control each role that has transfer permission in the contract. And it takes effect immediately after setting. If the owner permission is lost, it will cause capital loss to the project party.

Solution

It is recommended to transfer the authority of roles with excessive authorization risk to the governance contract, at least using multi-signature wallets.

Status

Ignored; Project party: At the beginning of the project, multi-signature will be used and then it will be converted to community governance.

[N6] [Medium] Variable without assignment

Category: Uninitialized Storage Pointers Vulnerability

Content

- contracts/vendor/Booster.sol

The variable `feeTokens` is not assigned in the contract, which will cause `require(feeTokens[_gauge].distro == address(0), "!gauge");` this judgment will be invalid

```
function addPool(address _lptoken, address _gauge) external returns(bool){
        require(msg.sender==poolManager && !isShutdown, "!add");
        require(_gauge != address(0) && _lptoken != address(0),"!param");
        require(feeTokens[_gauge].distro == address(0), "!gauge");

        //the next pool's pid
        uint256 pid = poolInfo.length;

        //create a tokenized deposit
        address token = ITokenFactory(tokenFactory).CreateDepositToken(_lptoken);
        //create a reward contract for crv rewards
        address newRewardPool =
IRewardFactory(rewardFactory).CreateCrvRewards(pid,token,_lptoken);

        IERC20(token).safeApprove(newRewardPool, type(uint256).max);

        //add the new pool
        poolInfo.push(
            PoolInfo({
                lptoken: _lptoken,
                token: token,
                gauge: _gauge,
                crvRewards: newRewardPool,
                shutdown: false
            })
        );

        uint256 distTokensLen = distributionTokens.length;
        for (uint256 i = 0; i < distTokensLen; i++) {
            IERC20(distributionTokens[i]).safeApprove(newRewardPool,
type(uint256).max);
```

```
        }

        emit PoolAdded(_lptoken, _gauge, token, newRewardPool, pid);
        return true;
    }
```

Solution

Variables must be set before use.

Status

Fixed; fixed in commit 5d24d41dd29759cedf7481d800140852b471ee48.

[N7] [Suggestion] Token transfer reminder

Category: Others

Content

- contracts/vendor/Booster.sol

If the token has a false top-up problem, or the return value does not meet the EIP20 specification, it may cause unexpected the result of. `token.transfer(tDistro.distro, amount);` The return value needs to be checked.

```
function _earmarkRewards(uint256 _pid) internal {
        PoolInfo storage pool = poolInfo[_pid];
        require(pool.shutdown == false, "pool is closed");

        address gauge = pool.gauge;
        //claim crv/wom and bonus tokens
        (address[] memory tokens, uint256[] memory balances) =
IStaker(voterProxy).claimCrv(gauge, _pid);

        uint256 tLen = tokens.length;
        for (uint256 i = 0; i < tLen; i++) {
            IERC20 token = IERC20(tokens[i]);
            uint256 balance = balances[i];

            emit EarmarkRewards(address(token), balance);

            if (balance == 0) {
```

```
                continue;
            }
            uint256 dLen = distributionByTokens[address(token)].length;
            require(dLen > 0, "!dLen");

            uint256 earmarkIncentiveAmount =
balance.mul(earmarkIncentive).div(DENOMINATOR);
            uint256 sentSum = earmarkIncentiveAmount;

            for (uint256 j = 0; j < dLen; j++) {
                TokenDistro memory tDistro = distributionByTokens[address(token)][j];
                uint256 amount = balance.mul(tDistro.share).div(DENOMINATOR);
                if (tDistro.callQueue) {
                    IRewards(tDistro.distro).queueNewRewards(address(token), amount);
                } else {
                    token.transfer(tDistro.distro, amount);
                }
                sentSum = sentSum.add(amount);
            }
            if (earmarkIncentiveAmount > 0) {
                token.safeTransfer(msg.sender, earmarkIncentiveAmount);
            }
            //send crv to lp provider reward contract
            IRewards(pool.crvRewards).queueNewRewards(address(token),
balance.sub(sentSum));
        }
    }
```

Solution

Check the return value.

Status

Fixed; fixed in commit 2739343e24449f7a19e7d330c4d1baa07594f1d9.

[N8] [Suggestion] Event log missing

Category: Malicious Event Log Audit

Content

- contracts/vendor/VoterProxy.sol

`setDepositor`, `setOperator`, `setRewardDeposit`, `setOwner` key function calls do not record events.

- contracts/vendor/CvxCrvToken.sol

`setOperator` key function calls do not record events.

- contracts/WmxVestedEscrow.sol

`setAdmin`, `setLocker` key function calls do not record events.

- contracts/WmxRewardPool.sol

`setLocker` key function calls do not record events.

- contracts/WmxLocker.sol

`addReward` key function calls do not record events.

Solution

Record key events.

Status

Fixed; fixed in commit 82500a60e4864a49f10a564a90299a20a56317ad.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002210100002	SlowMist Security Team	2022.09.16 - 2022.10.10	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 1 medium risk, 2 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