# Smart Contract Security Audit Report

[2021]

# Table Of Contents

# 1 Executive Summary

On 2021.09.13, the SlowMist security team received the Waterfall Protocol team's security audit application for Waterfall Protocol, 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

**(1) 2021.09.13 - 2021.09.22 audit scope**

Audit Version Code:

https://github.com/WaterfallDefi/Waterfall-contracts

commit: c9900ecec8737c6a0630b25c26c6a5860b23e9ec

Fixed Version Code:

https://github.com/WaterfallDefi/Waterfall-contracts

commit: c2feb30c39d4e8b0bfc31f6a41123c4db62ff134

**(2) 2021.11.13 - 2021.11.14 audit scope**

Audit Version Code:

https://github.com/WaterfallDefi/Waterfall-contracts

commit: 216f5b1e78cacf8e6df3682c84793657d2f6760e

Fixed Version Code:

https://github.com/WaterfallDefi/Waterfall-contracts

commit: 0a14f6b3443de91ac217483a574b3d97ea526293

## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | ProducedFee management security reminder | Others | Suggestion | Fixed |
| N2 | Excessive authority issues | Authority Control Vulnerability | Medium | Fixed |
| N3 | DoS issue | Denial of Service Vulnerability | Low | Confirmed |
| N4 | Underflow issue | Integer Overflow and Underflow Vulnerability | High | Fixed |
| N5 | Slippage check is missing | Design Logic Audit | Critical | Fixed |
| N6 | Missing permission check | Authority Control Vulnerability | Critical | Fixed |
| N7 | Compatibility security reminder | Others | Suggestion | Confirmed |
| N8 | Modifier name optimization | Others | Suggestion | Fixed |

# 4 Code Overview

## 4.1 Contracts Description

The main network address of the contract is as follows:

`Nov-16-2021 08:47:18 AM +UTC's contract status`

**BSC mainnet address:**

core: 0xE76e7b361C9fd6178b75079A30b8043C2bF95b11

masterwtf: 0xEe0C679c7e1D676ED919F97290D62dcCD4f4F853

mBUSD: 0x8e0695128FC3Fe13dA0Af009b179642e5611057a

alpacaStrat: 0x40707E5711ce08d77b1D970fB28D7b224F184125

venusStrat: 0x3B265276F2Cf95D200E7E1a138914e1C43Fdd359

creamStrat: 0xDa4C1e82B78C799bC7980583183390681A72Aff1

timelockcontroller: 0x870Ca87C64182cc315705194C0ffa8023EA741Ca

tranchemaster: 0xFDDA6514a13161a5a1724F8DD787d9d3faeC9557

WTF Token: 0x2fA0cac2c75Efb50382B5091C6494194eAcF65B0

**multi-sgin contract owners:**

https://gnosis-safe.io/app/bnb:0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B/settings/owners

**Core contract configuration:**

EXECUTOR_ROLE: 0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B(multi-sgin contract)

GOVERN_ROLE_0: 0xE76e7b361C9fd6178b75079A30b8043C2bF95b11(Core contract)

GOVERN_ROLE_1: 0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B(multi-sgin contract)

GUARDIAN_ROLE: None

MASTER_ROLE: 0xFDDA6514a13161a5a1724F8DD787d9d3faeC9557(TrancheMaster contract)

MULTISTRATEGY_ROLE: 0x8e0695128FC3Fe13dA0Af009b179642e5611057a(MultiStrategyToken contract)

PROPOSER_ROLE: 0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B(multi-sgin contract)

TIMELOCK_ROLE: 0x870Ca87C64182cc315705194C0ffa8023EA741Ca(timelockcontroller contract)

**masterwtf contract configuration：**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

rewardToken: 0x2fa0cac2c75efb50382b5091c6494194eacf65b0

**MultiStrategyToken contract configuration:**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

token: 0xe9e7cea3dedca5984780bafc599bd69add087d56

**StrategyVenus contract configuration:**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

distributionAddress: 0xfd36e2c2a6789db23113685031d7f16329158384

earnedAddress: 0xcf6bb5389c92bdda8a3747ddb454cb7a64626c63

owner: 0x8e0695128fc3fe13da0af009b179642e5611057a

uniRouterAddress: 0x10ed43c718714eb63d5aa57b78b54704e256024e

vTokenAddress: 0x95c78222b3d6e262426483d42cfa53685a67ab9d

wantAddress: 0xe9e7cea3dedca5984780bafc599bd69add087d56

wbnbAddress: 0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c

**StrategyCream contract configuration:**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

distributionAddress: 0x589de0f0ccf905477646599bb3e5c622c84cc0ba

earnedAddress: 0xd4cb328a82bdf5f03eb737f37fa6b370aef3e888

owner: 0x8e0695128fc3fe13da0af009b179642e5611057a

uniRouterAddress: 0x10ed43c718714eb63d5aa57b78b54704e256024e

vTokenAddress: 0x2bc4eb013ddee29d37920938b96d353171289b7c

wantAddress: 0xe9e7cea3dedca5984780bafc599bd69add087d56

wbnbAddress: 0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c

**StrategyAlpaca contract configuration：**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

alpacaAddress: 0x8f0528ce5ef7b51152a59745befdd91d97091d2f

fairLaunchAddress: 0xa625ab01b08ce023b2a342dbb12a16f2c8489a8f

uniRouterAddress: 0x10ed43c718714eb63d5aa57b78b54704e256024e

vaultAddress: 0x7c9e73d4c71dae564d41f78d56439bb4ba87592f

wantAddress: 0xe9e7cea3dedca5984780bafc599bd69add087d56

wbnbAddress: 0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c

owner: 0x8e0695128fc3fe13da0af009b179642e5611057a

**timelockcontroller contract configuration：**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

minDelay: 86400 (24 hour)

**tranchemaster contract configuration：**

core: 0xe76e7b361c9fd6178b75079a30b8043c2bf95b11

currency: 0xe9e7cea3dedca5984780bafc599bd69add087d56

devAddress: 0x46d0c754463e3bd07c1451cf4a683fecd507d36b

staker: 0xee0c679c7e1d676ed919f97290d62dccd4f4f853

strategy: 0x8e0695128fc3fe13da0af009b179642e5611057a

## 4.2 Visibility Description

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

| MasterChef | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |

| MasterChef | | | |
|---|---|---|---|
| <Constructor> | Public | Can Modify State | CoreRef |
| updateMultiplier | Public | Can Modify State | onlyGovernor |
| poolLength | Public | - | - |
| add | Public | Can Modify State | onlyGovernor |
| set | Public | Can Modify State | onlyGovernor validatePid |
| getMultiplier | Public | - | - |
| pendingReward | Public | - | validatePid |
| massUpdatePools | Public | Can Modify State | - |
| updatePool | Public | Can Modify State | validatePid |
| updateStake | Public | Can Modify State | onlyGuardian validatePid nonReentrant |
| start | Public | Can Modify State | onlyGuardian nonReentrant |
| next | Public | Can Modify State | onlyGuardian nonReentrant |
| claim | Public | Can Modify State | nonReentrant |
| claimAll | Public | Can Modify State | nonReentrant |
| safeRewardTransfer | Internal | Can Modify State | - |
| updaterewardPerBlock | Public | Can Modify State | onlyGuardianOrGovernor |

| Strategy | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | CoreRef |

| Strategy | | | |
|---|---|---|---|
| setFarms | Public | Can Modify State | onlyGovernor |
| setAlpacaToken | Public | Can Modify State | onlyGovernor |
| setVenusToken | Public | Can Modify State | onlyGovernor |
| setPancakeRouter | Public | Can Modify State | onlyGovernor |
| setVenusOracle | Public | Can Modify State | onlyGovernor |
| setAlpacaOracle | Public | Can Modify State | onlyGovernor |
| setSlipPageFactor | Public | Can Modify State | onlyRole |
| farm | Public | Can Modify State | onlyRole nonReentrant |
| redeem | Public | Can Modify State | onlyRole nonReentrant |
| _farm | Internal | Can Modify State | - |
| _redeem | Internal | Can Modify State | - |
| _safeSwap | Private | Can Modify State | - |

| TrancheMaster | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | CoreRef |
| setStrategy | Public | Can Modify State | onlyGovernor |
| setStaker | Public | Can Modify State | onlyGovernor |
| setDuration | Public | Can Modify State | onlyGovernor |

| TrancheMaster | | | |
|---|---|---|---|
| add | Public | Can Modify State | onlyGovernor |
| set | Public | Can Modify State | onlyGovernor checkTrancheID |
| _updateInvest | Internal | Can Modify State | - |
| balanceOf | Public | - | - |
| getInvest | Public | - | checkTrancheID |
| _tryStart | Internal | Can Modify State | - |
| investDirect | Public | Can Modify State | checkTrancheID checkNotActive updateInvest nonReentrant |
| deposit | Public | Can Modify State | updateInvest nonReentrant |
| invest | Public | Can Modify State | checkTrancheID checkNotActive updateInvest nonReentrant |
| _invest | Private | Can Modify State | - |
| redeem | Public | Can Modify State | checkTrancheID checkNotActive updateInvest nonReentrant |
| _redeem | Private | Can Modify State | - |
| redeemDirect | Public | Can Modify State | checkTrancheID checkNotActive updateInvest nonReentrant |
| withdraw | Public | Can Modify State | updateInvest nonReentrant |
| _startCycle | Internal | Can Modify State | checkNotActive |
| _stopCycle | Internal | Can Modify State | - |
| _calculateExchangeRate | Internal | - | - |

| TrancheMaster | | | |
|---|---|---|---|
| _processExit | Internal | Can Modify State | - |
| stop | Public | Can Modify State | checkActive nonReentrant |
| withdrawFee | Public | Can Modify State | onlyGovernor |

| WTF | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ERC20 ERC20Capped |

## 4.3 Vulnerability Summary

**[N1] [Suggestion] ProducedFee management security reminder**

**Category: Others**

**Content**

Governor can extract the handling fee in the contract. If the Governor is an EOA address, there may be a situation where the private key is stolen, and the handling fee will be maliciously transferred.

- contracts/main/TrancheMaster.sol#L561-L566

```
function withdrawFee(uint256 amount) public onlyGovernor {
        require(amount <= producedFee, "not enough balance for fee");
        producedFee = producedFee.sub(amount);
        IERC20(currency).safeTransfer(msg.sender, amount);
        emit WithdrawFee(msg.sender, amount);
    }
```

**Solution**

It is recommended to transfer the ProducedFee to the designated multi-signature address.

**Status**

Fixed; The project team using multi-sgin contract for management: https://gnosis-safe.io/app/bnb:0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B/settings/owners

### [N2] [Medium] Excessive authority issues

**Category: Authority Control Vulnerability**

**Content**

The Governor role can modify the configuration of the contract and modify the address of the external contract to be called. External contracts that have not undergone security audits may be at risk, which will affect the user's funds. In addition, there is no event record for the function to modify contract parameters, which is not conducive to the review of community users.

- contracts/main/Strategy.sol#L83-L113

```
function setFarms(
        address cream,
        uint256 s1,
        address venus,
        uint256 s2,
        address alpaca,
        uint256 s3
    ) public onlyGovernor {
        CreamFarm = cream;
        VenusFarm = venus;
        AlpacaFarm = alpaca;
        farms[0].addr = cream;
        farms[1].addr = venus;
        farms[2].addr = alpaca;
        farms[0].shares = s1.mul(PercentageScale).div(PercentageParamScale);
        farms[1].shares = s2.mul(PercentageScale).div(PercentageParamScale);
        farms[2].shares = s3.mul(PercentageScale).div(PercentageParamScale);
    }

function setAlpacaToken(address a) public onlyGovernor {
        AlpacaToken = a;
    }
```

```
function setVenusToken(address a) public onlyGovernor {
        VenusToken = a;
    }

function setPancakeRouter(address a) public onlyGovernor {
        PancakeRouter = a;
    }
```

- contracts/main/MasterChef.sol#L67-L70

```
function updateMultiplier(uint256 multiplierNumber) public onlyGovernor {
        BONUS_MULTIPLIER = multiplierNumber;
    }
```

- contracts/main/MasterChef.sol#L75-L95

```
function add(uint256 _allocPoint) public onlyGovernor {
        totalAllocPoint = totalAllocPoint.add(_allocPoint);
        poolInfo.push(PoolInfo({
            allocPoint: _allocPoint
        }));
    }

function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate)
        public
        onlyGovernor
        validatePid(_pid)
    {
        if (_withUpdate) {
            massUpdatePools();
        }
        uint256 prevAllocPoint = poolInfo[_pid].allocPoint;
        poolInfo[_pid].allocPoint = _allocPoint;
        if (prevAllocPoint != _allocPoint) {
            totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint);
        }
    }
```

contracts/main/TrancheMaster.sol#L170-L222

```
function setStrategy(address _strategy) public onlyGovernor {
        strategy = _strategy;
    }

    function setStaker(address _staker) public onlyGovernor {
        staker = _staker;
    }

    function setDuration(uint256 _duration) public onlyGovernor {
        duration = _duration;
    }

function add(
        uint256 target,
        uint256 apy,
        uint256 fee
    )
        public
        onlyGovernor
    {
        require(target > 0, "invalid target");
        require(apy <= MaxAPY, "invalid APY");
        require(fee <= MaxFee, "invalid fee");
        tranches.push(
            Tranche({
                target: target,
                apy: apy.mul(PercentageScale).div(PercentageParamScale),
                fee: fee,
                principal: 0
            })
        );
        emit TrancheAdd(tranches.length - 1, target, apy, fee);
    }

function set(
        uint256 tid,
        uint256 target,
        uint256 apy,
        uint256 fee
    )
        public
        onlyGovernor
        checkTrancheID(tid)
    {
        require(target >= tranches[tid].principal, "invalid target");
```

```
        require(apy <= MaxAPY, "invalid APY");
        require(fee <= MaxFee, "invalid fee");
        tranches[tid].target = target;
        tranches[tid].apy = apy.mul(PercentageScale).div(PercentageParamScale);
        tranches[tid].fee = fee;
        emit TrancheUpdated(tid, target, apy, fee);
    }
```

- contracts/refs/CoreRef.sol#L43-L46

```
function setCore(address core_) external onlyGovernor {
        _core = ICore(core_);
        emit CoreUpdate(core_);
    }
```

**Solution**

It is recommended that the authority of the Governor role be transferred to a timelock contract or governance contract, and the function of modifying contract parameters adopts event recording.

**Status**

Fixed; The project team using multi-sgin contract for management: https://gnosis-safe.io/app/bnb:0x46D0C754463E3Bd07c1451CF4a683fEcD507d36B/settings/owners

### [N3] [Low] DoS issue

**Category: Denial of Service Vulnerability**

**Content**

When poolInfo.length is lager, it will cause out of gas due to too many for loops.

- contracts/main/MasterChef.sol#L119-L124

```
function massUpdatePools() public {
        uint256 length = poolInfo.length;
        for (uint256 pid = 0; pid < length; ++pid) {
            updatePool(pid);
        }
    }
```

contracts/main/TrancheMaster.sol#L277-L288

```
function _tryStart() internal returns (bool) {
        for (uint256 i = 0; i < tranches.length; i++) {
            Tranche memory t = tranches[i];
            if (t.principal < t.target) {
                return false;
            }
        }

        _startCycle();

        return true;
    }
```

**Solution**

It is recommended not to add too many pools, and a manual update mechanism in batches is required.

**Status**

Confirmed; After communicating with the project team, the project will not add too many pools.

### [N4] [High] Underflow issue

**Category: Integer Overflow and Underflow Vulnerability**

**Content**

There is no underflow check when performing arithmetic operations, there has a underflow issue.

- contracts/main/Strategy.sol#L125-L150

```
function _farm(uint256 total) internal {
        if (total == 0) {
            return;
        }
        uint256 used = 0;
        uint256 amount = 0;

        // cream mint
        amount = total.mul(farms[0].shares).div(PercentageScale);
        IERC20(currency).approve(farms[0].addr, amount);
```

```
        ICompoundLike(farms[0].addr).mint(amount);
        used += amount;

        // venus mint
        amount = total.mul(farms[1].shares).div(PercentageScale);
        IERC20(currency).approve(farms[1].addr, amount);
        ICompoundLike(farms[1].addr).mint(amount);
        used += amount;

        // alpaca deposit
        amount = total - used;
        IERC20(currency).approve(farms[2].addr, amount);
        IAlpacaLike(farms[2].addr).deposit(amount);

        emit Farm(total);
    }
```

**Solution**

It is recommended to use SafeMath for arithmetic operations.

**Status**

Fixed; The issues has been fixed in commit: https://github.com/WaterfallDefi/Waterfall-contracts/commit/bf4dcee5ce2389f80902a16d6aff766fdb7c3e4f

### [N5] [Critical] Slippage check is missing

**Category: Design Logic Audit**

**Content**

During the swap operation, the amountOutmin was not checked, and there were sandwich attacks and malicious arbitrage issues.

- contracts/main/Strategy.sol#L194-L211

```
    function _swapForBUSDFromPancake(
        address _fromToken,
        address _toToken,
        uint256 amount
    ) private returns (uint256) {
        address[] memory swapPath2 = new address[](2);
```

```
        swapPath2[0] = _fromToken;
        swapPath2[1] = _toToken;
        uint256[] memory amountOuts = IUniswapV2Router02(PancakeRouter)
            .swapExactTokensForTokens(
                amount,
                1,
                swapPath2,
                address(this),
                block.timestamp + 200
            );
        return amountOuts[amountOuts.length - 1];
    }
```

**Solution**

It is recommended to use an oracle to get the price and check the slippage.

**Status**

Fixed; This issue has been fixed in commit: https://github.com/WaterfallDefi/Waterfall-contracts/commit/

3cbe73c638af377fe9516fa342a2703df1becade

### [N6] [Critical] Missing permission check

**Category: Authority Control Vulnerability**

**Content**

The stop function does not perform authentication checks, anyone can call the stop method to stop the cycle.

- contracts/main/TrancheMaster.sol#L557-L560

```
function stop() public checkActive nonReentrant {
        _stopCycle();
    }
```

**Solution**

It is recommended to add a permission check in the stop function.

**Status**

Fixed

### [N7] [Suggestion] Compatibility security reminder

**Category: Others**

**Content**

The project is not compatible with deflationary and inflationary tokens.

**Solution**

It is recommended to evaluate the compatibility when docking related tokens.

**Status**

Confirmed

### [N8] [Suggestion] Modifier name optimization

**Category: Others**

**Content**

The withdraw function is called by the Multiistrategy contract. Because it will execute

`IERC20(wantAddress).safeTransfer(owner(), balance);`,May cause community users to misunderstand that Owner will transfer assets to EOA addresses.

- Waterfall-contracts/contracts/strategy/StrategyAlpaca.sol#L89

```
    function withdraw(uint256 minReturnWant) public override onlyOwner nonReentrant {
        require(minReturnWant > 0, "Min return is zero");
        (uint256 _amount, , , ) = FairLaunch(fairLaunchAddress).userInfo(poolId,
address(this));
        FairLaunch(fairLaunchAddress).withdraw(address(this), poolId, _amount);
        Vault(vaultAddress).withdraw(Vault(vaultAddress).balanceOf(address(this)));

        uint256 earnedAmt = IERC20(alpacaAddress).balanceOf(address(this));
        if (alpacaAddress != wantAddress && earnedAmt != 0) {
            IPancakeRouter02(uniRouterAddress).swapExactTokensForTokens(
                earnedAmt,
```

```
                minReturnWant,
                earnedToWantPath,
                address(this),
                now.add(600)
            );
        }

        uint256 balance = wantLockedInHere();
        IERC20(wantAddress).safeTransfer(owner(), balance);
    }
```

- Waterfall-contracts/contracts/strategy/StrategyVenus.sol#L162

```
function withdraw(uint256 minReturnWant) public override onlyOwner nonReentrant {
        require(minReturnWant > 0, "Min return is zero");

        _withdraw();

        if (isComp) {
            IVenusDistribution(distributionAddress).claimComp(address(this));
        } else {
            IVenusDistribution(distributionAddress).claimVenus(address(this));
        }

        uint256 earnedAmt = IERC20(earnedAddress).balanceOf(address(this));
        if (earnedAddress != wantAddress && earnedAmt != 0) {
            IPancakeRouter02(uniRouterAddress).swapExactTokensForTokens(
                earnedAmt,
                minReturnWant,
                earnedToWantPath,
                address(this),
                now.add(600)
            );
        }

        uint256 wantBal = wantLockedInHere();
        IERC20(wantAddress).safeTransfer(owner(), wantBal);
    }
```

**Solution**

It is recommended to change onlyOwner to onlyMultistrategy to avoid misunderstanding by community users.

**Status**

Fixed; This issue has been fixed in commit: https://github.com/WaterfallDefi/Waterfall-contracts/commit/0a14f6b3443de91ac217483a574b3d97ea526293

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002111150004 | SlowMist Security Team | 2021.09.13 - 2021.09.22<br>2021.11.13 - 2021.11.14 | 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 2 critical risk, 1 high risk, 2 medium risk, 3 suggestion vulnerabilities. 2 suggestion vulnerabilities were confirmed; All other findings were fixed. The code was 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.

**Official Website**

www.slowmist.com

**E-mail**

team@slowmist.com

**Twitter**

@SlowMist_Team

**Github**

https://github.com/slowmist