Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.12.06, the SlowMist security team received the Earning.farm team's security audit application for CFFv2, 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 Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit Version:

https://gitlab.com/asresearch/cff-contract-v2/-/tree/master/contracts/core

commit: c86bef3f13c7585f547f9cd0ca900f94664e96b7

Fixed Version:

https://gitlab.com/asresearch/cff-contract-v2/-/tree/master/contracts/core

commit: 02b78791f0d950a462be1a4a1c7605463d1b9d34

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status

NO	Title	Category	Level	Status
N1	Low-level call issue	Others	Low	Confirmed
N2	Contract variable usage issue	Design Logic Audit	Suggestion	Confirmed
N3	`addExtraToken` issue	Design Logic Audit	Suggestion	Confirmed
N4	Missing event record	Others	Suggestion	Confirmed
N5	Logical redundancy issue	Design Logic Audit	Low	Confirmed
N6	Reentrancy risk	Reentrancy Vulnerability	Critical	Fixed
N7	Risk of excessive authority	Authority Control Vulnerability	Medium	Confirmed

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:

CFControllerV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

CFControllerV2			
setVault	Public	Can Modify State	onlyOwner
get_current_pool	Public	-	-
add_pool	Public	Can Modify State	onlyOwner
remove_pool	Public	Can Modify State	onlyOwner
change_current_pool	Public	Can Modify State	onlyOwner
_deposit	Internal	Can Modify State	-
deposit	Public	Can Modify State	onlyVault
get_pid	Internal	Can Modify State	-
withdraw	Public	Can Modify State	onlyVault
earnReward	Public	Can Modify State	onlyOwner
_refundTarget	Internal	Can Modify State	-
pause	Public	Can Modify State	onlyOwner
addExtraToken	Public	Can Modify State	onlyOwner
removeExtraToken	Public	Can Modify State	onlyOwner
changeYieldHandler	Public	Can Modify State	onlyOwner
changeFeePool	Public	Can Modify State	onlyOwner
changeHarvestFee	Public	Can Modify State	onlyOwner
clearCachedPID	Public	Can Modify State	onlyOwner
<Fallback>	External	Payable	-

CFVaultV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
set_max_amount	Public	Can Modify State	onlyOwner
set_slippage	Public	Can Modify State	onlyOwner
deposit	Public	Payable	nonReentrant
withdraw	Public	Can Modify State	-
changeWithdrawFee	Public	Can Modify State	onlyOwner
changeController	Public	Can Modify State	onlyOwner
changeFeePool	Public	Can Modify State	onlyOwner
get_virtual_price	Public	-	-
get_asset	Public	-	-
<Fallback>	External	Payable	-

CRVExchangeV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
path_from_addr	Public	-	-
path_to_addr	Public	-	-
handleCRV	Public	Can Modify State	-
handleExtraToken	Public	Can Modify State	-

CRVExchangeV2			
get_out_for_dex_path	Internal	-	-
addPath	Public	Can Modify State	onlyOwner
removePath	Public	Can Modify State	onlyOwner
removePathWithHash	Public	Can Modify State	onlyOwner
<Fallback>	External	Payable	-

UpgradeV1ToV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
upgrade	Public	Can Modify State	-
<Fallback>	External	Payable	-

BbtcPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

HbtcPool			
Function Name	Visibility	Mutability	Modifiers

HbtcPool			
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

SbtcPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

IWbtcPoolBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
deposit	Public	Can Modify State	onlyController
withdraw_from_curve	Internal	Can Modify State	-
withdraw	Public	Can Modify State	onlyController
setController	Public	Can Modify State	onlyOwner
get_lp_token_balance	Public	-	-

IWbtcPoolBase			
get_lp_token_addr	Public	-	-
callWithData	Public	Payable	onlyOwner

RenbtcPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

TbtcPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_wbtc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

IETHPoolBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_eth	Internal	Can Modify State	-

IETHPoolBase			
deposit	Public	Can Modify State	onlyController
withdraw_from_curve	Internal	Can Modify State	-
withdraw	Public	Can Modify State	onlyController
setController	Public	Can Modify State	onlyOwner
get_lp_token_balance	Public	-	-
get_lp_token_addr	Public	-	-
<Fallback>	External	Payable	-
callWithData	Public	Payable	onlyOwner

SethPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_eth	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

StethPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_eth	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-

StethPool			
get_virtual_price	Public	-	-

AavePool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

AlusdPoolV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

BUSDPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-

BUSDPool			
get_virtual_price	Public	-	-

CompoundPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

GUSDPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

IUSDCPoolBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
deposit	Public	Can Modify State	onlyController

IUSDCPoolBase			
withdraw_from_curve	Internal	Can Modify State	-
withdraw	Public	Can Modify State	onlyController
setController	Public	Can Modify State	onlyOwner
get_lp_token_balance	Public	-	-
get_lp_token_addr	Public	-	-
callWithData	Public	Payable	onlyOwner

LusdPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

TriPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

YPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit_usdc	Internal	Can Modify State	-
withdraw_from_curve	Internal	Can Modify State	-
get_virtual_price	Public	-	-

4.3 Vulnerability Summary

[N1] [Low] Low-level call issue

Category: Others

Content

In the CRVExchangeV2 contract, the handleExtraToken function is used to perform the token transfer operation after the token swap. Low-level calls are used when transferring native tokens, but the amount of gas usage is not limited, which may lead to unknown security risks.

Code location: contracts/core/CRVExchange.sol

```
  function handleExtraToken(address from, address target_token, uint256 amount, uint
min_amount) public{
    uint256 maxOut = 0;
    uint256 fpi = 0;

    for(uint pi = 0; pi < path_indexes.length; pi ++){
      if(path_from_addr(pi) != from || path_to_addr(pi) != target_token){
        continue;
      }
      uint256 t = get_out_for_dex_path(pi, amount);
      if( t > maxOut ){
        fpi = pi;
        maxOut = t;
```

```
      }
    }

    address dex = paths[path_indexes[fpi]].dex;
    IERC20(from).safeTransferFrom(msg.sender, address(this), amount);
    IERC20(from).safeApprove(dex, amount);
    if(target_token == weth){

SushiUniInterfaceERC20(dex).swapExactTokensForETHSupportingFeeOnTransferTokens(amount
, min_amount, paths[path_indexes[fpi]].path, address(this), block.timestamp + 10800);
      uint256 target_amount = address(this).balance;
      require(target_amount >= min_amount, "slippage screwed you");
      (bool status, ) = msg.sender.call.value(target_amount)("");
      require(status, "CRVExchange transfer eth failed");
    }else{

SushiUniInterfaceERC20(dex).swapExactTokensForTokensSupportingFeeOnTransferTokens(amo
unt, min_amount, paths[path_indexes[fpi]].path, address(this), block.timestamp +
10800);
      uint256 target_amount = IERC20(target_token).balanceOf(address(this));
      require(target_amount >= min_amount, "slippage screwed you");
      IERC20(target_token).safeTransfer(address(msg.sender), target_amount);
    }
  }
```

Solution

It is recommended to limit gas usage when transferring native tokens through low-level calls.

Status

Confirmed

[N2] [Suggestion] Contract variable usage issue

Category: Design Logic Audit

Content

In the IWbtcPoolBase contract, when the Controller role calls the deposit function, the deposit_wbtc_amount parameter will increase, but when the Controller role calls the withdraw function, the deposit_wbtc_amount parameter does not decrease accordingly. And the withdraw_wbtc_amount parameter in the contract is not used.

The deposit_eth_amount and withdraw_eth_amount parameters in the IETHPoolBase contract are the same.

The deposit_usdc_amount and withdraw_usdc_amount parameters in the IUSDCPoolBase contract are the same.

Code location:

contracts/core/btcpool/IWbtcPoolBase.sol

```
uint256 public deposit_wbtc_amount;
uint256 public withdraw_wbtc_amount;

function deposit(uint256 _amount) public onlyController{
  deposit_wbtc_amount = deposit_wbtc_amount + _amount;
  deposit_wbtc(_amount);
  uint256 cur = IERC20(lp_token_addr).balanceOf(address(this));
  lp_balance = lp_balance + cur;
  IERC20(lp_token_addr).safeTransfer(msg.sender, cur);
}

//@_amount: lp token amount
function withdraw(uint256 _amount) public onlyController{
  withdraw_from_curve(_amount);
  lp_balance = lp_balance - _amount;

  IERC20(wbtc).transfer(msg.sender, IERC20(wbtc).balanceOf(address(this)));
}
```

contracts/core/ethpool/IETHPoolBase.sol

```
uint256 public deposit_eth_amount;
uint256 public withdraw_eth_amount;

function deposit(uint256 amount) public onlyController{
  require(amount <= address(this).balance, "IETHPoolBase, not enough amount");
  uint _amount = amount;
  deposit_eth_amount = deposit_eth_amount + _amount;
  deposit_eth(_amount);
  uint256 cur = IERC20(lp_token_addr).balanceOf(address(this));
  lp_balance = lp_balance + cur;
  IERC20(lp_token_addr).safeTransfer(msg.sender, cur);
}
```

```
//@_amount: lp token amount
function withdraw(uint256 _amount) public onlyController{
    withdraw_from_curve(_amount);
    lp_balance = lp_balance - _amount;

    (bool status, ) = msg.sender.call.value(address(this).balance)("");
    require(status, "IETHPoolBase transfer eth failed");
}
```

contracts/core/pool/IPoolBase.sol

```
uint256 public deposit_usdc_amount;
uint256 public withdraw_usdc_amount;

function deposit(uint256 _amount) public onlyController{
  deposit_usdc_amount = deposit_usdc_amount + _amount;
  deposit_usdc(_amount);
  uint256 cur = IERC20(lp_token_addr).balanceOf(address(this));
  lp_balance = lp_balance + cur;
  IERC20(lp_token_addr).safeTransfer(msg.sender, cur);
}

//@_amount: lp token amount
function withdraw(uint256 _amount) public onlyController{
  withdraw_from_curve(_amount);
  lp_balance = lp_balance - _amount;
  IERC20(usdc).safeTransfer(msg.sender, IERC20(usdc).balanceOf(address(this)));
}
```

Solution

It is recommended to clarify the design expectations.

Status

Confirmed

[N3] [Suggestion] `addExtraToken` issue

Category: Design Logic Audit

Content

In the CFControllerV2 contract, the owner can add extra_yield_tokens through the addExtraToken function, but it does not check whether the added extra_yield_token already exists.

Code location: contracts/core/CFControllerV2.sol

```
function addExtraToken(address _new) public onlyOwner{
  require(_new != address(0x0), "invalid extra token");
  extra_yield_tokens.push(_new);
  emit AddExtraToken(_new);
}
```

Solution

It is recommended to check whether it already exists when adding a new extra_yield_token.

Status

Confirmed

[N4] [Suggestion] Missing event record

Category: Others

Content

In the IWbtcPoolBase contract, the owner can modify the controller and vault addresses through the setController function, but no event recording is performed.

Code location:

contracts/core/btcpool/IWbtcPoolBase.sol

contracts/core/ethpool/IETHPoolBase.sol

contracts/core/pool/IPoolBase.sol

```
function setController(address _controller, address _vault) public onlyOwner{
  controller = _controller;
  vault = _vault;
}
```

Solution

It is recommended to record incidents when modifying sensitive parameters for follow-up self-examination or community review.

Status

Confirmed

[N5] [Low] Logical redundancy issue

Category: Design Logic Audit

Content

In the AavePool contract, the withdraw_from_curve function will first authorize the lp_token_addr token to the pool_deposit contract and then call the remove_liquidity_one_coin function of the pool_deposit contract to remove liquidity. However, since the minter role of the lp_token_addr contract is the pool_deposit contract, there is no need to perform an approve operation.

Code location: contracts/core/pool/AavePool.sol

```
  function withdraw_from_curve(uint256 _amount) internal{
    require(_amount <= get_lp_token_balance(), "withdraw_from_curve: too large
amount");
    IERC20(lp_token_addr).approve(address(pool_deposit), _amount);
    pool_deposit.remove_liquidity_one_coin(_amount, 1, 0, true);
  }
```

Solution

It is recommended to remove redundant approve logic.

Status

Confirmed

[N6] [Critical] Reentrancy risk

Category: Reentrancy Vulnerability

Content

In the CFVaultV2 contract, the user can withdraw assets through the withdraw function, but in the withdraw function, it will first transfer the assets to the user and then destroy the user's credentials through the destroyTokens function. If the transfer is native tokens, this will lead to a risk of reentrancy.

Code location: contracts/core/CFVault.sol

```
  function withdraw(uint256 _amount) public{
    require(controller != CFControllerInterface(0x0) && controller.get_current_pool()
!= ICurvePool(0x0), "paused");
    require(slip != 0, "Slippage not set");
    uint256 amount = IERC20(lp_token).balanceOf(msg.sender);
    require(amount >= _amount, "no enough LP tokens");

    uint LP_token_amount =
_amount.safeMul(controller.get_current_pool().get_lp_token_balance()).safeDiv(IERC20(
lp_token).totalSupply());

    uint dec = uint(10)**(TransferableToken.decimals(target_token));
    uint vir = controller.get_current_pool().get_virtual_price();
    uint min_amount =
LP_token_amount.safeMul(vir).safeMul(slip).safeMul(dec).safeDiv(uint(1e40));

    uint256 _before = TransferableToken.balanceOfAddr(target_token, address(this));
    controller.withdraw(LP_token_amount);
    uint256 _after = TransferableToken.balanceOfAddr(target_token, address(this));
    uint256 target_amount = _after.safeSub(_before);

    require(target_amount >= min_amount, "Slippage");


    if(withdraw_fee_ratio != 0 && fee_pool != address(0x0)){
      uint256 f = target_amount.safeMul(withdraw_fee_ratio).safeDiv(ratio_base);
      uint256 r = target_amount.safeSub(f);
      TransferableToken.transfer(target_token, msg.sender, r);
      TransferableToken.transfer(target_token, fee_pool, f);
      TokenInterfaceERC20(lp_token).destroyTokens(msg.sender, _amount);
      emit CFFWithdraw(msg.sender, r, _amount, f, get_virtual_price());
```

```
    }else{
      TransferableToken.transfer(target_token, msg.sender, target_amount);
      TokenInterfaceERC20(lp_token).destroyTokens(msg.sender, _amount);
      emit CFFWithdraw(msg.sender, target_amount, _amount, 0, get_virtual_price());
    }
  }
```

Solution

It is recommended to follow the `Checks-Effects-Interactions` specification, or use the `nonReentrant` decorator.

Status

Fixed

[N7] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the IWbtcPoolBase, IETHPoolBase and IUSDCPoolBase contracts, the owner can call any data through the callWithData function. Since these strategies contracts indirectly keep the user's assets, any data call will cause the risk of excessive owner authority.

Code location:

contracts/core/btcpool/IWbtcPoolBase.sol

contracts/core/ethpool/IETHPoolBase.sol

contracts/core/pool/IPoolBase.sol

```
  function callWithData(address payable to, bytes memory data)public payable
onlyOwner{
    (bool status, ) = to.call.value(msg.value)(data);
    require(status, "call failed");
  }
```

Solution

It is recommended to transfer owner ownership to community governance

Status

Confirmed; After communicating with the project team, the project team stated that it will transfer the ownership of the owner to the multisig contract of ENF DAO in the future.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002112130002	SlowMist Security Team	2021.12.06 - 2021.12.13	Medium 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, 2 low risks, 3 suggestion vulnerabilities. All the findings were fixed. The code was not deployed to the mainnet. As the ownership of some of the contract owners has not been transferred to community governance, the project still has a risk of excessive authority.

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