慢雾科技
slow mist

智能合约安全审计报告

1. 概要

慢雾安全团队于 2021 年 02 月 18 日，收到 earning.farm 团队对 earning.farm 系统安全审计的申请，根据项目特点慢雾安全团队制定如下审计方案。

慢雾安全团队将采用“白盒为主，黑灰为辅”的策略，以最贴近真实攻击的方式，对项目进行安全审计。

慢雾科技 DeFi 项目测试方法：

黑盒测试	站在外部从攻击者角度进行安全测试。
灰盒测试	通过脚本工具对代码模块进行安全测试，观察内部运行状态，挖掘弱点。
白盒测试	基于项目的源代码，进行脆弱性分析和漏洞挖掘。

慢雾科技 DeFi 漏洞风险等级：

严重漏洞	严重漏洞会对项目的安全造成重大影响，强烈建议修复严重漏洞。
高危漏洞	高危漏洞会影响项目的正常运行，强烈建议修复高危漏洞。
中危漏洞	中危漏洞会影响项目的运行，建议修复中危漏洞。
低危漏洞	低危漏洞可能在特定场景中会影响项目的业务操作，建议项目方自行评估和考虑这些问题是否需要修复。
弱点	理论上存在安全隐患，但工程上极难复现。
增强建议	编码或架构存在更好的实践方法。

2. 审计方法

慢雾安全团队智能合约安全审计流程包含两个步骤:

- 使用开源或内部自动化分析的工具对合约代码中常见的安全漏洞进行扫描和测试。
- 人工审计代码的安全问题，通过人工分析合约代码，发现代码中潜在的安全问题。

如下是合约代码审计过程中我们会重点审查的漏洞列表:

（其他未知安全漏洞不包含在本次审计责任范围）

- 重入攻击
- 重放攻击
- 重排攻击
- 短地址攻击
- 拒绝服务攻击
- 交易顺序依赖
- 条件竞争攻击
- 权限控制攻击
- 整数上溢/下溢攻击
- 时间戳依赖攻击
- Gas 使用，Gas 限制和循环
- 冗余的回调函数
- 不安全的接口使用
- 函数状态变量的显式可见性
- 逻辑缺陷
- 未声明的存储指针
- 算术精度误差
- tx.origin 身份验证
- 假充值漏洞
- 变量覆盖

3. 项目背景

3.1 项目介绍

earning.farm 协议无缝对接市场上经过时间和资金验证的，安全且回报较高的 Defi 协议，为用户提供极低风险但收益较高的数字资产回报。本金的安全性是 earning.farm 第一考虑的要素。除了本身是使用智能合约开发且经过多家安全公司的审计，对接的其他协议也都选择了同质资产，比如稳定币，btc，eth 等，以防止币价的高波动带来的资产损益。在不远的将来，协议的管理将进化成 Dao 的形式，由社区对协议管理和进化，排除初创团队的中心化风险。

审计合约文件：

项目源代码

审计初始版本：

文件名：

cff.zip:

SHA256:

6638ef7940660f01597da84c4748ba8845a4d6acb80311fe79b343d56e7f7ab1

审计最终版本：

文件名：

cff.zip:

SHA256:

25c635af28f0fbfa17215f8cf2159b7b2e39db241f7910435f9053dda8e51008

3.2 审计合约结构

```
.
├── AddressList.sol
├── Migrations.sol
├── TrustList.sol
├── TrustListTools.sol
├── assets
│   └── TokenBank.sol
├── core
│   ├── CFController.sol
│   ├── CFETHController.sol
│   ├── CFETHVault.sol
│   ├── CFVault.sol
│   ├── CRVExchange.sol
│   ├── IPool.sol
│   ├── btcpool
│   │   ├── BbtcPool.sol
│   │   ├── HbtcPool.sol
│   │   ├── IWbtcPoolBase.sol
│   │   └── TbtcPool.sol
│   ├── ethpool
│   │   ├── IethPoolBase.sol
│   │   └── SethPool.sol
│   └── pool
│       ├── AavePool.sol
│       ├── BusdPool.sol
│       ├── CompoundPool.sol
│       ├── GusdPool.sol
│       ├── IPoolBase.sol
│       ├── TriPool.sol
│       └── YPool.sol
├── erc20
│   ├── ERC20DepositApprover.sol
│   ├── ERC20Impl.sol
│   ├── ERC20Token.sol
│   ├── IERC20.sol
│   ├── SafeERC20.sol
│   └── TokenInterface.sol
├── test
│   ├── CRV.sol
│   ├── DummyDex.sol
```

```
|    ├── STDERC20.sol
|    ├── TestAddressList.sol
|    ├── TestERC20.sol
|    ├── TestTokenClaimer.sol
|    ├── TestUSDCPool.sol
|    ├── TestUSDCPool2.sol
|    ├── USDC.sol
|    ├── USDT.sol
|    ├── WBTC.sol
|    ├── hack.sol
|    └── hacketh.sol
└── utils
     ├── Address.sol
     ├── AddressArray.sol
     ├── Ownable.sol
     ├── ReentrancyGuard.sol
     ├── SafeMath.sol
     └── TokenClaimer.sol
```

3.3 项目合约地址

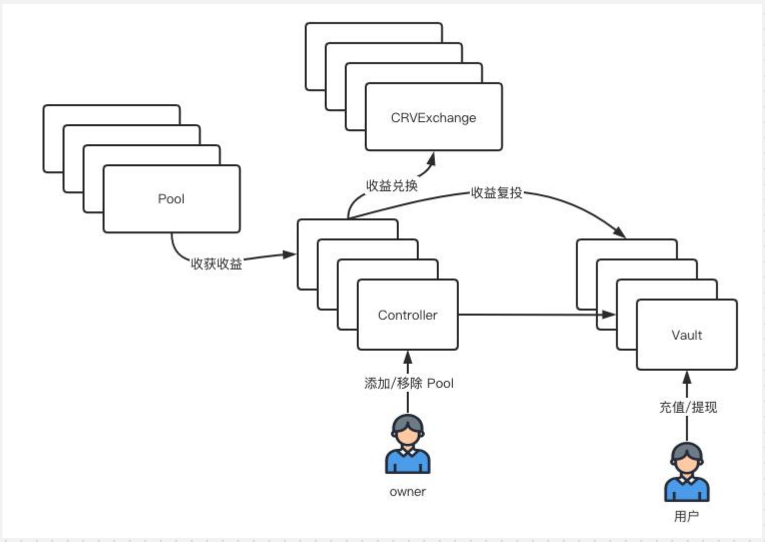
合约名	合约地址
CRVExchange	0x65ff67d6a61ae6bab8821c584f5f394d35e9b50e
TrustList（USDC）	0x246e50a0161b005AC9DD0AE80661C5ae843B64c3
CFToken（USDC）	0x44B439f6624e84c29f2ee5fd3C2CCf29C8DC7572
ERC20DepositApprover（USDC）	0xe5afC078684683dc232E053c2c9D86015Aa00Ec6
CFVault（USDC）	0xa1E263225E24333CA4d26083C94092D6bfEe1DDd
CFController（USDC）	0x6c57eA7e3B81b8a166ED5b8E396a401C7a9c890b
CompoundPool	0x4484F01080e8F596d82407926e99895A947EC87b

BUSDPool	0x01964ca263624b105DDE4486E6d1130A207D9117
TriPool	0x35fD3579956808e68D4DF7b3efD63575230Df26F
YPool	0xE673562C20bC3898b91d7700e81Fcb30abDb398C
AavePool	0x6E2549D909ED2D461b0594CC755eE6dD927e0640
GUSDPool	0xb7768f0672a64953e30C4F0c0a2Bd901172fB020
TrustList（WBTC）	0xb9a5263108F14EF5021047a82852f473c2Dc400f
CFToken（WBTC）	0x0319180cA78edB186fBFC7786E0E5f51FDF21263
CFVault（WBTC）	0x1C62D47Aae452877D4E4ff6D7ECDaE5A50ef7326
CFController（WBTC）	0xeb792a00F482DD2f75702dE9eFbC1C2C72e54a2E
HbtcPool	0x8C224F75433EaF4e1B07E0FCa9Ba79148498384D
BbtcPool	0x8E8482f207AeEF576bfe4D5526a53894eE9aAc2b
TbtcPool	0xbab88A555c865744Ef3d207649A215D56576D663
TrustList (ETH)	0x745d97Eb8c714ac839d3dD505C996A3dFA27B476
CFETHToken	0xA709eCF2253B18A757214D64F42026Be8F008bD8
CFETHVault	0xa5eafc384f22d743EDfFa1aE5375bac95495fE2e
CFETHController	0x56Ac11ac8801c1929D19bb84B5f06a9D7c551B1e
SethPool	0x7478064432745612B7944C3229ff7ece51bf3e3E

4. 代码概述

4.1 合约架构

earning.farm 系统主要由 5 个部分组成，分别是 Controller 合约、Pool 合约、ERC20 Token 合约、CRVExchange 合约、Vault 合约。其中 Pool 合约用于将对应的资金充值到 Curve 中，Controller 合于负责添加/移除对应的 Pool 池以及收益复投，ERC20 Token 合约用于生成用户对应的充值份额，CRVExchange 合约用于将收益转换成其他代币进行复投，Vault 合约用于处理用户充值和提现。整体架构图如下：



4.2 主要合约函数可见性分析

在审计过程中，慢雾安全团队对核心合约的可见性进行分析，结果如下：

BUSDPool			
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	-	-

TokenBank			
Function Name	Visibility	Mutability	Modifiers
fallback	External	payable	-
constructor	Public	can modify state	TrustListTools
claimStdTokens	Public	can modify state	onlyOwner
balance	Public	can modify state	-
token	Public	-	-
transfer	Public	can modify state	onlyOwner
issue	Public	can modify state	is_trusted

TokenBankFactory			
Function Name	Visibility	Mutability	Modifiers
newTokenBank	Public	can modify state	-

TrustListInterface			
Function Name	Visibility	Mutability	Modifiers
is_trusted	Public	can modify state	-

TrustListTools			
Function Name	Visibility	Mutability	Modifiers

constructor	Public	can modify state	-

TokenClaimer			
Function Name	Visibility	Mutability	Modifiers
_claimStdTokens	Internal	can modify state	-

CurveInterface			
Function Name	Visibility	Mutability	Modifiers
add_liquidity	Public	can modify state	-
remove_liquidity_one_coin	Public	can modify state	-

HbtcPool			
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	can modify state	-

PriceInterface			
Function Name	Visibility	Mutability	Modifiers
get_virtual_price	Public	-	-

CRVGaugeInterface			
Function Name	Visibility	Mutability	Modifiers

deposit	Public	can modify state	-
withdraw	Public	can modify state	-
claim_rewards	Public	can modify state	-

MinterInterface			
Function Name	Visibility	Mutability	Modifiers
mint	Public	can modify state	-

IWbtcPoolBase			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
deposit_wbtc	Internal	can modify state	-
deposit	Public	can modify state	onlyController
deposit_to_gauge	Internal	can modify state	-
withdraw_from_curve	Internal	can modify state	-
withdraw	Public	can modify state	onlyController
withdraw_from_gauge	Internal	can modify state	-
setController	Public	can modify state	onlyOwner
claimStdToken	Public	can modify state	onlyOwner
earn_crv	Public	can modify state	onlyController
get_lp_token_balance	Public	-	-
get_lp_token_addr	Public	-	-

IethPoolBase

Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
deposit_eth	Internal	can modify state	-
deposit	Public	payable	onlyController
deposit_to_gauge	Internal	can modify state	-
withdraw_from_curve	Internal	can modify state	-
withdraw	Public	can modify state	onlyController
withdraw_from_gauge	Internal	can modify state	-
setController	Public	can modify state	onlyOwner
claimStdToken	Public	can modify state	onlyOwner
earn_crv	Public	can modify state	onlyController
get_lp_token_balance	Public	-	-
get_lp_token_addr	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	-	-

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	-	-

CFController			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
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
earnCRV	Public	can modify state	-
refundTarget	Public	can modify state	-
pauseAndTransferTo	Public	can modify state	onlyOwner
changeExtraToken	Public	can modify state	onlyOwner
changeCRVHandler	Public	can modify state	onlyOwner
changeFeePool	Public	can modify state	onlyOwner

changeHarvestFee	Public	can modify state	onlyOwner

CFControllerFactory			
Function Name	Visibility	Mutability	Modifiers
createCFController	Public	can modify state	-

CFETHController			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
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
earnCRV	Public	can modify state	-
refundTarget	Public	payable	-
pauseAndTransferTo	Public	can modify state	onlyOwner
changeExtraToken	Public	can modify state	onlyOwner
changeCRVHandler	Public	can modify state	onlyOwner
changeFeePool	Public	can modify state	onlyOwner
changeHarvestFee	Public	can modify state	onlyOwner

CFETHControllerFactory			
Function Name	Visibility	Mutability	Modifiers

createCFETHController	Public	can modify state	-

CFETHVault			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
deposit	Public	payable	nonReentrant
withdraw	Public	can modify state	nonReentrant
changeWithdrawFee	Public	can modify state	onlyOwner
changeDepositFee	Public	can modify state	onlyOwner
changeController	Public	can modify state	onlyOwner
changeFeePool	Public	can modify state	onlyOwner
get_virtual_price	Public	-	-

CFETHVaultFactory			
Function Name	Visibility	Mutability	Modifiers
createCFETHVault	Public	can modify state	-

ERC20Base			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
transfer	Public	can modify state	-
transferFrom	Public	can modify state	-
doTransfer	Internal	can modify state	-
balanceOf	Public	-	-

approve	Public	can modify state	-
allowance	Public	-	-
approveAndCall	Public	can modify state	-
totalSupply	Public	-	-
balanceOfAt	Public	-	-
totalSupplyAt	Public	-	-
_generateTokens	Internal	can modify state	-
_destroyTokens	Internal	can modify state	-
_enableTransfers	Internal	can modify state	-
getValueAt	Internal	-	-
updateValueAtNow	Internal	can modify state	-
onTransferDone	Internal	can modify state	-
_addTransferListener	Internal	can modify state	-
_removeTransferListener	Internal	can modify state	-
min	Internal	-	-

CFVault			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
deposit	Public	can modify state	-
withdraw	Public	can modify state	-
changeWithdrawFee	Public	can modify state	onlyOwner
changeDepositFee	Public	can modify state	onlyOwner
changeController	Public	can modify state	onlyOwner
changeFeePool	Public	-	onlyOwner

get_virtual_price	Public		-

CRVExchange			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	ERC20Base TrustListTools
claimStdTokens	Public	can modify state	onlyOwner
createCloneToken	Public	can modify state	-
addTransferListener	Public	can modify state	onlyOwner
removeTransferListener	Public	can modify state	onlyOwner
generateTokens	Public	can modify state	is_trusted
destroyTokens	Public	can modify state	is_trusted
enableTransfers	Public	can modify state	onlyOwner

4.3 代码审计详情

4.3.1 高危漏洞

4.3.1.1 功能不可用

a. 各 remove_liquidity_one_coin 接口参数列表中的 i 变量类型错误，导致无法正常提现，应修改相应的类型为 int256 类型。

```
contract CurveInterface{
  function add_liquidity(uint256[2] memory uamounts, uint256 min_mint_amount) public;
  function remove_liquidity_one_coin(uint256 _token_amount, uint128 i, uint256 min_mint_amount) public;

  address public curve;}
```

b. Aave Pool 中的 PriceInterface 在获取价格的时候使用了 CurveInterface 的 curve 变量，但是实际的 Curve Aave 池并没有这个变量。导致接口无法正常使用，导致无法对 Aave Pool 进行充提。

```
function get_virtual_price() public view returns(uint256){
    return PriceInterface(pool_deposit.curve()).get_virtual_price();
}
```

修复情况：已修复。

4.3.2 中危漏洞

4.3.2.1 未进行滑点检查

a. IPoolBase / IethPoolBase/ IWbtcPoolBase 合约在处理用户资金充值时直接将对应资金转入 Curve 资金池中，并未做相应的滑点控制，导致用户在充值时可能被抢跑，导致资产损失。

```
function deposit(uint256 _amount) public{
    deposit_usdc_amount = deposit_usdc_amount + _amount;
    deposit_usdc(_amount);
    uint256 cur = IERC20(lp_token_addr).balanceOf(address(this));
    lp_balance = lp_balance + cur;
    deposit_to_gauge();
}
```

```
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;
    deposit_to_gauge();
}
```

```
function deposit() public payable onlyController{
    deposit_wbtc_amount = deposit_wbtc_amount + _amount;
    deposit_eth();
    uint256 cur = IERC20(lp_token_addr).balanceOf(address(this));
```

```
        lp_balance = lp_balance + cur;
        deposit_to_gauge();
    }
```

b. 在进行 CRV 兑换操作的时候，合约未对兑换滑点进行检查，导致可能因滑点造成的损失。

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

        for(uint di = 0; di < dexs.length; di ++){
            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(di, pi, amount);
                if( t > maxOut ){
                    fdi = di;
                    fpi = pi;
                    maxOut = t;
                }
            }
        }
        IERC20(from).transferFrom(msg.sender, address(this), amount);
        IERC20(from).approve(dexs[fdi], amount);
        SushiUniInterface(dexs[fdi]).swapExactTokensForTokens(amount, 0, paths[path_indexes[fpi]], address(this),
block.timestamp + 10800);

        uint256 target_amount = IERC20(target_token).balanceOf(address(this));
        IERC20(target_token).approve(address(msg.sender), target_amount);
        CFControllerInterface(msg.sender).refundTarget(target_amount);
    }
```

修复情况：已修复。

4.3.2.2 权限过大风险

controller / Vault / Exchgange / TrustList / Pool 的 owner 权限 目前未移交到对应的 Timelock 合约或社区治理，项目方可通过 owner 权限任意修改敏感参数，或转移用户资金，存在权限过大风险。

修复情况：未修复。

4.3.3 增强建议

4.3.3.1 未对校验充值金额进行校验。

```
function deposit(uint256 _amount) public{
    require(controller != CFControllerInterface(0x0) && controller.get_current_pool() != ICurvePool(0x0), "paused");
    require(IERC20(target_token).allowance(msg.sender, address(this)) >= _amount, "CFVault: not enough allowance");

    //SlowMist// 未对 _amount 大小进行限制，导致 gas 浪费

    uint tt_before = IERC20(target_token).balanceOf(address(this));
    IERC20(target_token).safeTransferFrom(msg.sender, address(this), _amount);

    if(deposit_fee_ratio != 0 && fee_pool != address(0x0)){
      uint256 f = _amount.safeMul(deposit_fee_ratio).safeDiv(ratio_base);
      emit CFFDepositFee(msg.sender, _amount, f);
      _amount = _amount.safeSub(f);
      if(f != 0){
        IERC20(target_token).safeTransfer(fee_pool, f);
      }
    }

    uint tt_after = IERC20(target_token).balanceOf(address(this));
    _amount = tt_after.safeSub(tt_before);

    IERC20(target_token).safeApprove(address(controller.get_current_pool()), 0);
    IERC20(target_token).safeApprove(address(controller.get_current_pool()),_amount);
```

修复情况：忽略。

4.3.3.2 地址硬编码

合约中多处地方将地址硬编码，一旦相关外部地址改变后，将导致业务逻辑失效。

修复情况：忽略。

4.3.3.3 操作简化导致可能的计算数值误差

CFValut.sol / CFETHVault.sol 合约在处理充值的精度时采用精度合并的方式扩大充值金额，而不是先扩大再缩小的方式，可能导致一定程度上的误差。

```
   function deposit(uint256 _amount) public{
    require(controller != CFControllerInterface(0x0) && controller.get_current_pool() != ICurvePool(0x0), "paused");
    require(IERC20(target_token).allowance(msg.sender, address(this)) >= _amount, "CFVault: not enough allowance");
    require(_amount <= max_amount, "too large amount");
    require(slip != 0, "Slippage not set");

    uint tt_before = IERC20(target_token).balanceOf(address(this));
    IERC20(target_token).safeTransferFrom(msg.sender, address(this), _amount);

    if(deposit_fee_ratio != 0 && fee_pool != address(0x0)){
      uint256 f = _amount.safeMul(deposit_fee_ratio).safeDiv(ratio_base);
      emit CFFDepositFee(msg.sender, _amount, f);
      if(f != 0){
        IERC20(target_token).safeTransfer(fee_pool, f);
      }
    }

    uint tt_after = IERC20(target_token).balanceOf(address(this));
    _amount = tt_after.safeSub(tt_before);

    uint dec = uint(10)**(ERC20Base(target_token).decimals());
    uint vir = controller.get_current_pool().get_virtual_price();
    uint min_amount = _amount.safeMul(uint(1e32)).safeMul(slip).safeDiv(dec).safeDiv(vir);

    IERC20(target_token).safeApprove(address(controller.get_current_pool()), 0);
    IERC20(target_token).safeApprove(address(controller.get_current_pool()),_amount);
```

```
function deposit() public payable nonReentrant{
    require(controller != CFETHControllerInterface(0x0) && controller.get_current_pool() != ICurvePoolForETH(0x0),
"paused");
    require(msg.value > 0, "CFVault: zero amount");
    require(slip != 0, "Slippage not set");
    uint _amount = msg.value;

    require(_amount <= max_amount, "too large amount");

    if(deposit_fee_ratio != 0 && fee_pool != address(0x0)){
      uint256 f = _amount.safeMul(deposit_fee_ratio).safeDiv(ratio_base);
      emit CFFDepositFee(msg.sender, _amount, f);
      _amount = _amount.safeSub(f);
      if(f != 0){
        fee_pool.transfer(f);
      }
    }
    uint eth_before = address(this).balance;

    uint vir = controller.get_current_pool().get_virtual_price();
    uint min_amount = _amount.safeMul(uint(1e14)).safeMul(slip).safeDiv(vir);

    uint lp_before = controller.get_current_pool().get_lp_token_balance();
```

修复情况：忽略

4.3.3.4 未对 refundTarget 函数进行限制，可能导致用户资金损失

CFController.sol / CFETHController.sol 合约未对 refundTarget 函数限制调用来源，用户可能会误操作

调用该函数导致资金损失。

```
function refundTarget() public payable{
    //IERC20(target_token).safeTransferFrom(msg.sender, address(this), _amount);
    uint _amount = msg.value;
    if(harvest_fee_ratio != 0 && fee_pool != address(0x0)){
      uint256 f = _amount.safeMul(harvest_fee_ratio).safeDiv(ratio_base);
      emit CFFRefund(_amount, f);
      _amount = _amount.safeSub(f);
      if(f != 0){
        fee_pool.transfer(_amount);
      }
    }else{
      emit CFFRefund(_amount, 0);
    }
    ICurvePoolForETH(current_pool).deposit.value(_amount)();
}
```

修复情况： CFETHController.sol 合约已修复，CFController 合约忽略。

5. 审计结果

5.1 总结

审计结论 ： 中风险

审计编号 ：0X002103180003

审计时间 ：2021 年 03 月 18 日

审计团队 ： 慢雾安全团队

审计总结 ： 慢雾安全团队采用人工结合内部工具对代码进行分析。审计期间发现 9 个问题。其中包含 2 个高危漏洞、3 个中危漏、并提出了 4 点增强建议。由于目前项目各合约权限暂未移交给社区治理或 TimeLock 合约，因此项目仍存在权限过大的风险。综合评估为中风险。

6. 声明

慢雾仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，慢雾无法判断其智能合约安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料(简称“已提供资料”)。慢雾假设：已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任。