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 Mdex 团队对 Mdex 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

MDEX.COM，火币生态链 Heco 全生态币种交易最大的平台，一款基于资金池理念自动做市的去中心化交易产品，作为一款功能完备的 DEX 的同时还提出并实现了基于火币生态链和以太坊公链的双链 DEX 模型。融合了火币生态链交易费低廉和以太坊生态圈繁荣的优势，支持流动性挖矿和交易挖矿的“双重挖矿机制”。MDEX.COM 致力于打造火币生态链 Heco 上的集 DEX、IMO、DAO 为一体的 DeFi 平台，为用户提供更加安全可信、资产选择和配置更多样化、预期投资回报率更高的去中心化代币兑换服务。MDEX.COM 是 Heco 生态中价值交换的核心枢纽，是打通 CeFi 与 DeFi 市场的一次重要尝试。

审计合约文件：

项目源代码

审计初始版本：

https://github.com/mdexSwap/contracts

commit: f9650a130c67c4b7804e8f92355ad7a7d2d50722

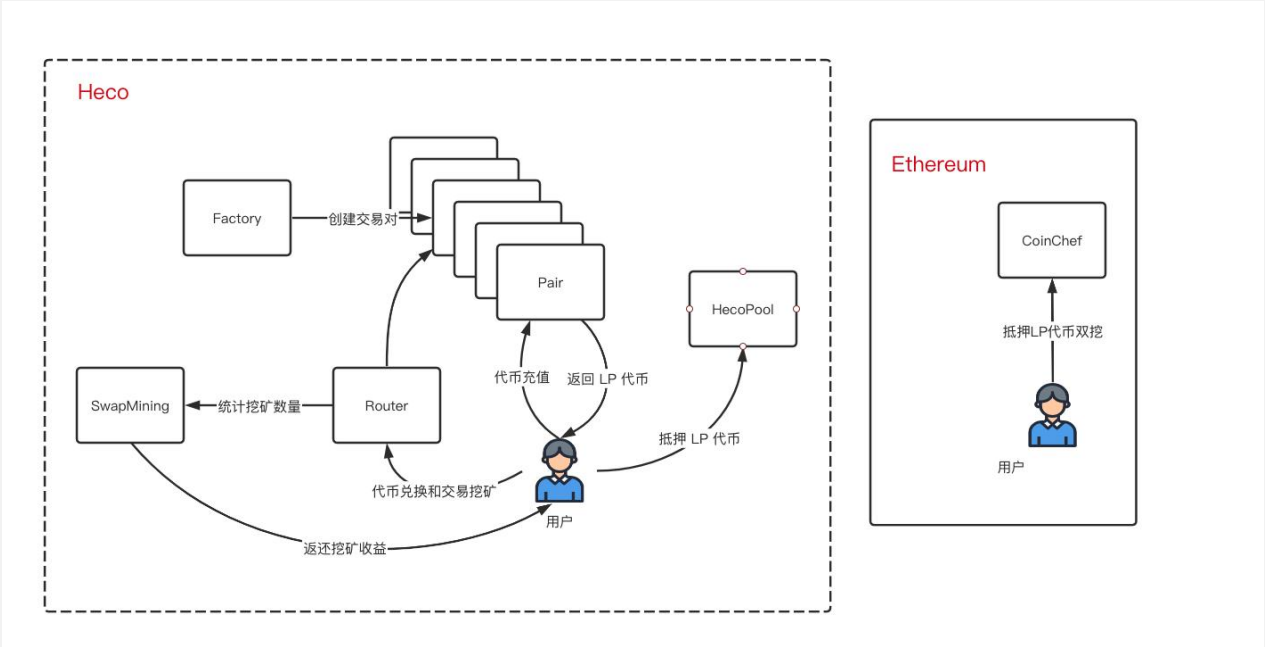
3.2 项目结构

```
contracts
├── Migrations.sol
├── assets
│   ├── Airdrop.sol
│   ├── AirdropMDX.sol
│   ├── BlackHole.sol
│   └── Repurchase.sol
├── governance
│   ├── GovernorAlpha.sol
│   └── Timelock.sol
├── heco
```

```
|   ├── Factory.sol
|   ├── HecoPool.sol
|   ├── MdxTokenHeco.sol
|   ├── Router.sol
|   └── SwapMining.sol
├── interface
|   ├── IERC20.sol
|   ├── IMdexFactory.sol
|   ├── IMdexPair.sol
|   └── IMdx.sol
├── library
|   └── SafeMath.sol
├── mainnet
|   ├── CoinChef.sol
|   └── MdxToken.sol
├── oracle
|   └── Oracle.sol
└── timeLock
    └── TeamTimeLock.sol
```

3.3 项目架构

Mdex 项目根据链分成 2 个部分，其中 Ethereum 链主要提供代币双挖功能，用户将 LP 代币存入 CoinChef 合约中可联动 SushiSwap 代币池进行代币双挖。Heco 链提供代币交易对创建、LP 抵押挖矿及交易挖矿功能。整体架构图如下：



4. 代码概述

4.1 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:

CoinChef			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
poolLength	Public	-	-
addSushiLP	Public	can modify state	onlyOwner
isSushiLP	Public	-	-
getSushiLPLength	Public	-	-
getSushiLPAddress	Public	-	-

add	Public	can modify state	-
set	Public	can modify state	onlyOwner
setPoolCorr	Public	can modify state	onlyOwner
massUpdatePools	Public	can modify state	-
updatePool	Public	can modify state	-
pending	External	-	-
pendingMdxAndSushi	Private	-	-
pendingMdx	Private	-	-
deposit	Public	can modify state	-
depositMdxAndSushi	Private	can modify state	-
depositMdx	Private	can modify state	-
withdraw	Public	can modify state	-
withdrawMdxAndSushi	Private	can modify state	-
withdrawMdx	Private	can modify state	-
emergencyWithdraw	Public	can modify state	-
emergencyWithdrawMdxAndSushi	Private	can modify state	-
emergencyWithdrawMdx	Private	can modify state	-
safeMdxTransfer	Internal	can modify state	-

MdexOracleLibrary			
Function Name	Visibility	Mutability	Modifiers
currentBlockTimestamp	Internal	-	-
currentCumulativePrices	Internal	-	-

SwapMining			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
poolLength	Public	-	-
addPair	Public	can modify state	onlyOwner
setPair	Public	can modify state	onlyOwner
setMdxPerBlock	Public	can modify state	onlyOwner
addWhitelist	Public	can modify state	onlyOwner
delWhitelist	Public	can modify state	onlyOwner
getWhitelistLength	Public	-	-
isWhitelist	Public	-	-
getWhitelist	Public	-	-
setHalvingPeriod	Public	can modify state	onlyOwner
setRouter	Public	can modify state	onlyOwner
setOracle	Public	can modify state	onlyOwner
phase	Public	can modify state	-
phase	Public	-	-
reward	Public	-	-
reward	Public	-	-
getMdxReward	Public	-	-
massMintPools	Public	can modify state	-
mint	Public	can modify state	-
swap	Public	can modify state	onlyRouter
takerWithdraw	Public	can modify state	-

getUserReward	Public	-	-
getPoolInfo	Public	-	-
getQuantity	Public	-	-

Oracle			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
update	External	can modify state	-
computeAmountOut	Private	-	-
consult	External	-	-

TransferHelper			
Function Name	Visibility	Mutability	Modifiers
safeApprove	Internal	can modify state	-
safeTransfer	Internal	can modify state	-
safeTransferFrom	Internal	can modify state	-

Airdrop			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
poolLength	External	-	-
newAirdrop	Public	can modify state	onlyOwner
updatePoolLastRewardBlock	Private	can modify state	-

setCycle	Public	can modify state	onlyOwner
add	Public	can modify state	onlyOwner
set	Public	can modify state	onlyOwner
massUpdatePools	Public	can modify state	-
updatePool	Public	can modify state	-
pending	External	-	-
deposit	Public	can modify state	-
withdraw	Public	can modify state	-
emergencyWithdraw	Public	can modify state	-
safeWhtTransfer	Internal	can modify state	-

MdexERC20			
Function Name	**Visibility**	**Mutability**	**Modifiers**
constructor	Public	can modify state	-
_mint	Internal	can modify state	-
_burn	Internal	can modify state	-
_approve	Private	can modify state	-
_transfer	Private	can modify state	-
approve	External	can modify state	-
transfer	External	can modify state	-
transferFrom	External	can modify state	-
permit	External	can modify state	-

MdexPair			
Function Name	Visibility	Mutability	Modifiers
getReserves	Public	-	-
_safeTransfer	Private	can modify state	-
constructor	Public	can modify state	-
initialize	External	can modify state	-
_update	Private	can modify state	-
_mintFee	Private	can modify state	-
mint	External	can modify state	lock
burn	External	can modify state	lock
swap	External	can modify state	lock
skim	External	can modify state	lock
sync	External	can modify state	lock
price	Public	-	-

MdexFactory			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
allPairsLength	External	-	-
createPair	External	can modify state	-
setFeeTo	External	can modify state	-
setFeeToSetter	External	can modify state	-
setFeeToRate	External	can modify state	-

sortTokens	Public	-	-
pairFor	Public	-	-
getReserves	Public	-	-
quote	Public	-	-
getAmountOut	Public	-	-
getAmountIn	Public	-	-
getAmountsOut	Public	-	-
getAmountsIn	Public	-	-

UQ112x112			
Function Name	Visibility	Mutability	Modifiers
encode	Internal	-	-
uqdiv	Internal	-	-

HecoPool			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
setHalvingPeriod	Public	can modify state	onlyOwner
setMdxPerBlock	Public	can modify state	onlyOwner
poolLength	Public	-	NO
addMultLP	Public	can modify state	onlyOwner
isMultLP	Public	-	NO
getMultLPLength	Public	-	-

getMultLPAddress	Public	-	-
setPause	Public	can modify state	onlyOwner
setMultLP	Public	can modify state	onlyOwner
replaceMultLP	Public	can modify state	onlyOwner
add	Public	can modify state	onlyOwner
set	Public	can modify state	onlyOwner
setPoolCorr	Public	can modify state	onlyOwner
phase	Public	-	-
reward	Public	-	-
getMdxBlockReward	Public	-	-
massUpdatePools	Public	can modify state	-
updatePool	Public	can modify state	-
pending	External	-	-
pendingMdxAndToken	Private	-	-
pendingMdx	Private	-	-
deposit	Public	can modify state	notPause
depositMdxAndToken	Private	can modify state	-
depositMdx	Private	can modify state	-
withdraw	Public	can modify state	notPause
withdrawMdxAndToken	Private	can modify state	-
withdrawMdx	Private	can modify state	-
emergencyWithdraw	Public	can modify state	notPause
emergencyWithdrawMdxAndToken	Private	can modify state	-
emergencyWithdrawMdx	Private	can modify state	-
safeMdxTransfer	Internal	can modify state	-

DelegateERC20			
Function Name	Visibility	Mutability	Modifiers
_mint	Internal	can modify state	-
_transfer	Internal	can modify state	-
delegate	External	can modify state	-
delegateBySig	External	can modify state	-
getCurrentVotes	External	-	-
getPriorVotes	External	-	-
_delegate	Internal	-	-
_moveDelegates	Internal	can modify state	-
_writeCheckpoint	Internal	can modify state	-
safe32	Internal	-	-
getChainId	Internal	-	-

MdxToken			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	ERC20
mint	Public	can modify state	onlyMinter
addMinter	Public	can modify state	onlyOwner
delMinter	Public	can modify state	onlyOwner
getMinterLength	Public	-	-
isMinter	Public	-	-

getMinter	Public	-	onlyOwner

MdexRouter			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
fallback	External	payable	-
pairFor	Public	-	-
setSwapMining	Public	can modify state	onlyOwner
_addLiquidity	Internal	can modify state	-
addLiquidity	External	can modify state	ensure
addLiquidityETH	External	payable	ensure
removeLiquidity	Public	can modify state	ensure
removeLiquidityETH	Public	can modify state	ensure
removeLiquidityWithPermit	External	can modify state	-
removeLiquidityETHWithPermit	External	can modify state	-
removeLiquidityETHSupportingFeeOnTransferTokens	Public	can modify state	-
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	can modify state	-
_swap	Internal	can modify state	-
swapExactTokensForTokens	External	can modify state	ensure
swapTokensForExactTokens	External	can modify state	ensure
swapExactETHForTokens	External	payable	ensure
swapTokensForExactETH	External	can modify state	ensure
swapExactTokensForETH	External	can modify state	ensure

swapETHForExactTokens	External	payable	ensure
_swapSupportingFeeOnTransferTokens	Internal	can modify state	-
swapExactTokensForTokensSupporting FeeOnTransferTokens	External	can modify state	ensure
swapExactETHForTokensSupportingFee OnTransferTokens	External	payable	ensure
swapExactTokensForETHSupportingFee OnTransferTokens	External	can modify state	ensure
quote	Public	-	-
getAmountOut	Public	-	-
getAmountIn	Public	-	-
getAmountsOut	Public	-	-
getAmountsIn	Public	-	-

TeamTimeLock			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	can modify state	-
getBalance	Public	-	-
getReward	Public	-	-
withDraw	External	can modify state	-
setBeneficiary	Public	can modify state	-

4.2 代码审计详情

4.2.1 中危漏洞

4.2.1.1 交易量伪造风险

SwapMining 合约中，合约在计算用户交易挖矿金额时未考虑用户使用闪电贷进行兑换的问题。当用户使用闪电贷进行交易挖矿时，用户只需支付交易兑换中损耗的手续费以及闪电贷本身的费用，即可伪造闪电贷金额数量的交易。放大并伪造了交易的数量。

代码位置：SwapMining.sol swap 函数

```
function swap(address account, address input, address output, uint256 amount) public onlyRouter returns (bool) {
        require(account != address(0), "SwapMining: taker swap account is the zero address");
        require(input != address(0), "SwapMining: taker swap input is the zero address");
        require(output != address(0), "SwapMining: taker swap output is the zero address");

        if (poolLength() <= 0) {
            return false;
        }

        if (!isWhitelist(input) || !isWhitelist(output)) {
            return false;
        }

        address pair = IMdexFactory(factory).pairFor(input, output);

        PoolInfo storage pool = poolInfo[pairOfPid[pair]];
        // If it does not exist or the allocPoint is 0 then return
        if (pool.pair != pair || pool.allocPoint <= 0) {
            return false;
        }

        uint256 quantity = getQuantity(output, amount, targetToken);
        if (quantity <= 0) {
            return false;
        }
```

```
        mint(pairOfPid[pair]);
        //SlowMist// 未对用户是否使用闪电贷做检查，导致用户可通过闪电贷伪造交易量
        pool.quantity = pool.quantity.add(quantity);
        pool.totalQuantity = pool.totalQuantity.add(quantity);
        UserInfo storage user = userInfo[pairOfPid[pair]][account];
        user.quantity = user.quantity.add(quantity);
        user.blockNumber = block.number;
        return true;
    }
```

修复状态：忽略，经与项目方确认后，项目方确认此类攻击成本较高。目前存在较多用户同时参与交易挖矿活动。攻击者难以通过闪电贷放大的交易量获利。

4.2.2 低危漏洞

4.2.2.1 权限过大问题

HecoPool 合约的 Owner 权限可对敏感参数进行修改、添加新的代币池和双挖池等。存在权限过大问题。该功能属于业务设计上的需要且无法直接影响到用户的资产，建议添加事件记录方便社区用户对参数设置改动的审查。

```
function setMultLP(address _multLpToken, address _multLpChef) public onlyOwner {
        require(_multLpToken != address(0) && _multLpChef != address(0), "is the zero address");
        multLpToken = _multLpToken;
        multLpChef = _multLpChef;
    }

    function replaceMultLP(address _multLpToken, address _multLpChef) public onlyOwner {
        require(_multLpToken != address(0) && _multLpChef != address(0), "is the zero address");
        require(paused == true, "No mining suspension");
        multLpToken = _multLpToken;
        multLpChef = _multLpChef;
        uint256 length = getMultLPLength();
        while (length > 0) {
            address dAddress = EnumerableSet.at(_multLP, 0);
            uint256 pid = LpOfPid[dAddress];
            IMasterChefHeco(multLpChef).emergencyWithdraw(poolCorrespond[pid]);
```

```
            EnumerableSet.remove(_multLP, dAddress);
            length--;
        }
    }

    // Add a new lp to the pool. Can only be called by the owner.
    // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
    function add(uint256 _allocPoint, IERC20 _lpToken, bool _withUpdate) public onlyOwner {
        require(address(_lpToken) != address(0), "_lpToken is the zero address");
        if (_withUpdate) {
            massUpdatePools();
        }
        uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
        totalAllocPoint = totalAllocPoint.add(_allocPoint);
        poolInfo.push(PoolInfo({
        lpToken : _lpToken,
        allocPoint : _allocPoint,
        lastRewardBlock : lastRewardBlock,
        accMdxPerShare : 0,
        accMultLpPerShare : 0,
        totalAmount : 0
        }));
        LpOfPid[address(_lpToken)] = poolLength() - 1;
    }

    // Update the given pool's MDX allocation point. Can only be called by the owner.
    function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public onlyOwner {
        if (_withUpdate) {
            massUpdatePools();
        }
        totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
        poolInfo[_pid].allocPoint = _allocPoint;
    }

    // The current pool corresponds to the pid of the multLP pool
    function setPoolCorr(uint256 _pid, uint256 _sid) public onlyOwner {
        require(_pid <= poolLength() - 1, "not find this pool");
        poolCorrespond[_pid] = _sid;
    }

    function phase(uint256 blockNumber) public view returns (uint256) {
```

```
        if (halvingPeriod == 0) {
            return 0;
        }
        if (blockNumber > startBlock) {
            return (blockNumber.sub(startBlock).sub(1)).div(halvingPeriod);
        }
        return 0;
    }
```

修复状态：忽略，经过与项目方的沟通讨论，目前 multLpChef 和 multLpToken 参数的均是 public。社区用户可以直接获取到参数的值对参数设置状态进行审查。

5. 审计结果

5.1 总结

审计结论 ：通过

审计编号 ：0X002103010001

审计时间 ：2021 年 03 月 01 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。审计期间发现了 2 个问题。其中包含 1 个中危漏洞、1 个低危漏洞，经过与项目方沟通反馈确认审计过程中发现的中危风险中的交易量伪造风险忽略，低危风险中的权限过大问题忽略。综合评估无风险

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。

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号