智能合约安全审计报告

目录

1 前言

慢雾安全团队于 2021 年 03 月 08 日，收到 autofarm 团队对 autofarm 项目的代码进行安全审计的申请，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慢雾安全团队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

审计版本的代码：

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/56e3bde53c532a0faf114dceb51a23b526b33f8c/AutoFarmV2_CrossChain.sol

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/e8f460aad1dda0e840f5421818b5dbcaf68dfc33/StratX2_MDEX.sol

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/a22112979cb67aa61b5c2fd9da81705922e63d61/StratVLEV2.sol

修复版本的代码:

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/b27f2cafcf51667726bd70220420707c89b75975/AutoFarmV2_CrossChain.sol

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/b27f2cafcf51667726bd70220420707c89b75975/StratX2_MDEX.sol

https://github.com/autofarm-network/AutofarmV2_CrossChain/blob/b27f2cafcf51667726bd70220420707c89b75975/StratVLEV2.sol

4. 代码概述

4.1 合约可见性分析

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:

StratX2_MDEX			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
noTimeLockFunc1	Public	Can Modify State	-
_convertMDXToEarned	Internal	Can Modify State	-

StratX2			
Function Name	Visibility	Mutability	Modifiers
deposit	Public	Can Modify State	onlyOwner nonReentrant whenNotPaused
farm	Public	Can Modify State	nonReentrant
_farm	Internal	Can Modify State	-
_unfarm	Internal	Can Modify State	-
withdraw	Public	Can Modify State	onlyOwner nonReentrant
earn	Public	Can Modify State	whenNotPaused
buyBack	Internal	Can Modify State	-
distributeFees	Internal	Can Modify State	-
convertDustToEarned	Public	Can Modify State	whenNotPaused
pause	Public	Can Modify State	-

unpause	Public	Can Modify State	-
setSettings	Public	Can Modify State	-
setGov	Public	Can Modify State	-
setOnlyGov	Public	Can Modify State	-
setUniRouterAddress	Public	Can Modify State	-
setBuyBackAddress	Public	Can Modify State	-
setRewardsAddress	Public	Can Modify State	-
inCaseTokensGetStuck	Public	Can Modify State	-
_wrapBNB	Internal	Can Modify State	-
wrapBNB	Public	Can Modify State	-
_safeSwap	Internal	Can Modify State	-

AutoFarmV2_CrossChain			
Function Name	Visibility	Mutability	Modifiers
poolLength	External	-	-
add	Public	Can Modify State	onlyOwner
stakedWantTokens	External	-	-
deposit	Public	Can Modify State	nonReentrant
withdraw	Public	Can Modify State	nonReentrant
inCaseTokensGetStuck	Public	Can Modify State	onlyOwner

StratVLEV2			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_supply	Internal	Can Modify State	-
_removeSupply	Internal	Can Modify State	-
_borrow	Internal	Can Modify State	-
_repayBorrow	Internal	Can Modify State	-
deposit	Public	Can Modify State	onlyOwner nonReentrant whenNotPaused
farm	Public	Can Modify State	nonReentrant
_farm	Internal	Can Modify State	-
_leverage	Internal	Can Modify State	-
deleverageOnce	Public	Can Modify State	-
_deleverageOnce	Internal	Can Modify State	-
deleverageUntilNotOverLevered	Public	Can Modify State	-
_deleverage	Internal	Can Modify State	-
rebalance	External	Can Modify State	-
earn	External	Can Modify State	whenNotPaused

buyBack	Internal	Can Modify State	-
distributeFees	Internal	Can Modify State	-
withdraw	External	Can Modify State	onlyOwner nonReentrant
pause	Public	Can Modify State	-
unpause	External	Can Modify State	-
_resetAllowances	Internal	Can Modify State	-
resetAllowances	Public	Can Modify State	-
updateBalance	Public	Can Modify State	-
wantLockedTotal	Public	-	-
wantLockedInHere	Public	-	-
setSettings	Public	Can Modify State	-
setGov	Public	Can Modify State	-
setOnlyGov	Public	Can Modify State	-
setUniRouterAddress	Public	Can Modify State	-
setBuyBackAddress	Public	Can Modify State	-
setRewardsAddress	Public	Can Modify State	-
inCaseTokensGetStuck	Public	Can Modify State	-
_wrapBNB	Internal	Can Modify State	-
_unwrapBNB	Internal	Can Modify State	-
wrapBNB	Public	Can Modify State	-
_safeSwap	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

4.2 代码审计

4.2.1 高危漏洞

4.2.1.1 三明治攻击问题

如下函数 earn(), buyBack(), convertDustToEarned(), _convertMDXToEarned 没有对滑点进行限制存在三明治攻击的问题，建议添加滑点限制并且滑点的参数仅能 Owner 进行修改。

- AutofarmV2_CrossChain/StratVLEV2.sol

```
function earn() external whenNotPaused {
        if (onlyGov) {
            require(msg.sender == govAddress, "!gov");
        }
```

```
        IVenusDistribution(venusDistributionAddress).claimVenus(address(this));

        uint256 earnedAmt = IERC20(venusAddress).balanceOf(address(this));

        earnedAmt = distributeFees(earnedAmt);
        earnedAmt = buyBack(earnedAmt);

        if (venusAddress != wantAddress) {
            IPancakeRouter02(uniRouterAddress).swapExactTokensForTokens(
                earnedAmt,
                0,
                venusToWantPath,
                address(this),
                now.add(600)
            );
        }
        lastEarnBlock = block.number;

        _farm(false); // Supply wantToken without leverage, to cater for net -ve interest rates.
    }
```

- AutofarmV2_CrossChain/StratVLEV2.sol

```
function buyBack(uint256 _earnedAmt) internal returns (uint256) {
        if (buyBackRate <= 0) {
            return _earnedAmt;
        }

        uint256 buyBackAmt = _earnedAmt.mul(buyBackRate).div(buyBackRateMax);

        IPancakeRouter02(uniRouterAddress).swapExactTokensForTokens(
            buyBackAmt,
            0,
            earnedToAUTOPath,
            buyBackAddress,
            now + 600
        );

        return _earnedAmt.sub(buyBackAmt);
    }
```

- AutofarmV2_CrossChain/StratX2_MDEX.sol

```
function convertDustToEarned() public whenNotPaused {
        require(isAutoComp, "!isAutoComp");
        require(!isCAKEStaking, "isCAKEStaking");

        // Converts dust tokens into earned tokens, which will be reinvested on the next earn().

        // Converts token0 dust (if any) to earned tokens
        uint256 token0Amt = IERC20(token0Address).balanceOf(address(this));
        if (token0Address != earnedAddress && token0Amt > 0) {
            IERC20(token0Address).safeIncreaseAllowance(
                uniRouterAddress,
                token0Amt
            );

            // Swap all dust tokens to earned tokens
            IPancakeRouter02(uniRouterAddress)
                .swapExactTokensForTokensSupportingFeeOnTransferTokens(
                token0Amt,
                0,
                token0ToEarnedPath,
                address(this),
                now + 600
            );
        }

        // Converts token1 dust (if any) to earned tokens
        uint256 token1Amt = IERC20(token1Address).balanceOf(address(this));
        if (token1Address != earnedAddress && token1Amt > 0) {
            IERC20(token1Address).safeIncreaseAllowance(
                uniRouterAddress,
                token1Amt
            );

            // Swap all dust tokens to earned tokens
            IPancakeRouter02(uniRouterAddress)
                .swapExactTokensForTokensSupportingFeeOnTransferTokens(
                token1Amt,
                0,
                token1ToEarnedPath,
                address(this),
                now + 600
```

```
            );
        }
    }
```

- AutofarmV2_CrossChain/StratX2_MDEX.sol

```
function _convertMDXToEarned() internal {
        // Converts MDX (if any) to earned tokens
        uint256 MDXAmt = IERC20(MDXAddress).balanceOf(address(this));
        if (MDXAddress != earnedAddress && MDXAmt > 0) {
            IERC20(MDXAddress).safeIncreaseAllowance(uniRouterAddress, MDXAmt);

            // Swap all dust tokens to earned tokens
            IPancakeRouter02(uniRouterAddress)
                .swapExactTokensForTokensSupportingFeeOnTransferTokens(
                MDXAmt,
                0,
                MDXToEarnedPath,
                address(this),
                now + 60
            );
        }
    }
```

- AutofarmV2_CrossChain/StratX2_MDEX.sol

```
function buyBack(uint256 _earnedAmt) internal returns (uint256) {
        if (buyBackRate <= 0) {
            return _earnedAmt;
        }

        uint256 buyBackAmt = _earnedAmt.mul(buyBackRate).div(buyBackRateMax);

        if (earnedAddress == AUTOAddress) {
            IERC20(earnedAddress).safeTransfer(buyBackAddress, buyBackAmt);
        } else {
            IERC20(earnedAddress).safeIncreaseAllowance(
                uniRouterAddress,
                buyBackAmt
            );

            IPancakeRouter02(uniRouterAddress)
                .swapExactTokensForTokensSupportingFeeOnTransferTokens(
                buyBackAmt,
```

```
                0,
                earnedToAUTOPath,
                buyBackAddress,
                now + 600
            );
        }

        return _earnedAmt.sub(buyBackAmt);
    }
```

- AutofarmV2_CrossChain/StratVLEV2.sol

```
function earn() external whenNotPaused {
        if (onlyGov) {
            require(msg.sender == govAddress, "!gov");
        }

        IVenusDistribution(venusDistributionAddress).claimVenus(address(this));

        uint256 earnedAmt = IERC20(venusAddress).balanceOf(address(this));

        earnedAmt = distributeFees(earnedAmt);
        earnedAmt = buyBack(earnedAmt);

        if (venusAddress != wantAddress) {
            IPancakeRouter02(uniRouterAddress).swapExactTokensForTokens(
                earnedAmt,
                0,
                venusToWantPath,
                address(this),
                now.add(600)
            );
        }

        lastEarnBlock = block.number;

        _farm(false); // Supply wantToken without leverage, to cater for net -ve interest rates.
    }
```

修复状态：该问题在 commit: b27f2cafcf51667726bd70220420707c89b75975 中进行了修复。

4.2.2 中风险漏洞

4.2.2.1 权限过大审计

add 函数存在权限过大问题, Owner 可以任意添加挖矿的池子，存在 Owner 偷挖的风险，并且用户的资产是发送给 _strat 地址的，Owner 可以任意设置 _strat 地址，注意与外部合约的兼容性问题，建议将 Owner 权限移交给 timelock 合约,并且添加事件对 add 函数进行记录,并在设置外部合约之前要对兼容性进行评估。

- AutofarmV2_CrossChain/AutoFarmV2_CrossChain.sol

```
function add(
        uint256 _allocPoint,
        IERC20 _want,
        bool _withUpdate,
        address _strat
    ) public onlyOwner {
        if (_withUpdate) {
            massUpdatePools();
        }

        totalAllocPoint = totalAllocPoint.add(_allocPoint);
        poolInfo.push(
            PoolInfo({
                want: _want,
                allocPoint: _allocPoint,
                lastRewardBlock: 0,
                accAUTOPerShare: 0,
                strat: _strat
            })
        );
    }
```

修复状态：已修复，已经将 Owner 权限移交给 timelock 合约。

参考：

https://hecoinfo.com/tx/0xbe90b5dba8315b30a010ea957e9631154857b93d84cdb344c11b339b5f3e5421

Gov 角色的权限过大，可以任意设置外部合约的地址，恶意和错误的外部合约会导致用户的资金受到损失，

存在权限过大的问题，建议将 Gov 角色的权限转移给 timelock 合约。

- AutofarmV2_CrossChain/StratX2.sol

```
function setEntranceFeeFactor(uint256 _entranceFeeFactor) public {
        require(msg.sender == govAddress, "!gov");
        require(_entranceFeeFactor >= entranceFeeFactorLL, "!safe – too low");
        require(_entranceFeeFactor <= entranceFeeFactorMax, "!safe – too high");
        entranceFeeFactor = _entranceFeeFactor;
    }

    function setWithdrawFeeFactor(uint256 _withdrawFeeFactor) public {
        require(msg.sender == govAddress, "!gov");
        require(_withdrawFeeFactor >= withdrawFeeFactorLL, "!safe – too low");
        require(_withdrawFeeFactor <= withdrawFeeFactorMax, "!safe – too high");
        withdrawFeeFactor = _withdrawFeeFactor;
    }

    function setControllerFee(uint256 _controllerFee) public {
        require(msg.sender == govAddress, "!gov");
        require(_controllerFee <= controllerFeeUL, "too high");
        controllerFee = _controllerFee;
    }

    function setbuyBackRate(uint256 _buyBackRate) public {
        require(msg.sender == govAddress, "!gov");
        require(buyBackRate <= buyBackRateUL, "too high");
        buyBackRate = _buyBackRate;
    }

    function setGov(address _govAddress) public {
        require(msg.sender == govAddress, "!gov");
        govAddress = _govAddress;
    }

    function setOnlyGov(bool _onlyGov) public {
        require(msg.sender == govAddress, "!gov");
        onlyGov = _onlyGov;
    }

    function setUniRouterAddress(address _uniRouterAddress) public {
```

```
        require(msg.sender == govAddress, "!gov");
        uniRouterAddress = _uniRouterAddress;
    }

    function setBuyBackAddress(address _buyBackAddress) public {
        require(msg.sender == govAddress, "!gov");
        buyBackAddress = _buyBackAddress;
    }

    function setRewardsAddress(address _rewardsAddress) public {
        require(msg.sender == govAddress, "!gov");
        rewardsAddress = _rewardsAddress;
    }
```

修复状态：已修复，已经将 Gov 权限移交给 timelock 合约。

参考：

https://hecoinfo.com/tx/0xfdf183915b5659473f9e8e3438c295cb859e022faa073a0a8f12c38e0a4c257d

4.2.2.2 DoS 风险

massUpdatePools 函数中没有限制 poolInfo 的长度，如果 poolInfo 的长度太大会导致 Out of gas，存在 DoS 的风险，建议对 poolInfo 的长度进行限制，避免 DoS 的风险。

- AutofarmV2_CrossChain/AutoFarmV2_CrossChain.sol

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

修复状态：该问题在 commit: b27f2cafcf51667726bd70220420707c89b75975 中进行了修复。

4.2.3 增强建议

4.2.3.1 事件记录缺失

"setEntranceFeeFactor", "setWithdrawFeeFactor" , "setControllerFee" , "setbuyBackRate", "setGov", "setOnlyGov", "setUniRouterAddress", "setBuyBackAddress", "setRewardsAddress" 函数没有使用事件进行记录，建议添加事件进行记录。

- AutofarmV2_CrossChain/StratX2_MDEX.sol
- AutofarmV2_CrossChain/StratVLEV2.sol

```
function setEntranceFeeFactor(uint256 _entranceFeeFactor) public {
    require(msg.sender == govAddress, "!gov");
    require(_entranceFeeFactor >= entranceFeeFactorLL, "!safe – too low");
    require(_entranceFeeFactor <= entranceFeeFactorMax, "!safe – too high");
    entranceFeeFactor = _entranceFeeFactor;
}

function setWithdrawFeeFactor(uint256 _withdrawFeeFactor) public {
    require(msg.sender == govAddress, "!gov");
    require(_withdrawFeeFactor >= withdrawFeeFactorLL, "!safe – too low");
    require(_withdrawFeeFactor <= withdrawFeeFactorMax, "!safe – too high");
    withdrawFeeFactor = _withdrawFeeFactor;
}

function setControllerFee(uint256 _controllerFee) public {
    require(msg.sender == govAddress, "!gov");
    require(_controllerFee <= controllerFeeUL, "too high");
    controllerFee = _controllerFee;
}

function setbuyBackRate(uint256 _buyBackRate) public {
    require(msg.sender == govAddress, "!gov");
    require(buyBackRate <= buyBackRateUL, "too high");
    buyBackRate = _buyBackRate;
}

function setGov(address _govAddress) public {
```

```
        require(msg.sender == govAddress, "!gov");
        govAddress = _govAddress;
    }

    function setOnlyGov(bool _onlyGov) public {
        require(msg.sender == govAddress, "!gov");
        onlyGov = _onlyGov;
    }

    function setUniRouterAddress(address _uniRouterAddress) public {
        require(msg.sender == govAddress, "!gov");
        uniRouterAddress = _uniRouterAddress;
    }

    function setBuyBackAddress(address _buyBackAddress) public {
        require(msg.sender == govAddress, "!gov");
        buyBackAddress = _buyBackAddress;
    }

    function setRewardsAddress(address _rewardsAddress) public {
        require(msg.sender == govAddress, "!gov");
        rewardsAddress = _rewardsAddress;
    }
```

修复状态：该问题在 commit: b27f2cafcf51667726bd70220420707c89b75975 中进行了修复。

4.2.3.2 缺失 nonReentrant 修饰器

deposit 函数缺失 nonReentrant 修饰器，建议添加 nonReentrant 修饰器。

- AutofarmV2_CrossChain/StratX2_MDEX.sol

```
function deposit(address _userAddress, uint256 _wantAmt)
        public
        onlyOwner
        whenNotPaused
        returns (uint256)
    {
        IERC20(wantAddress).safeTransferFrom(
            address(msg.sender),
            address(this),
            _wantAmt
```

```
        );

        uint256 sharesAdded = _wantAmt;
        if (wantLockedTotal > 0 && sharesTotal > 0) {
            sharesAdded = _wantAmt
                .mul(sharesTotal)
                .mul(entranceFeeFactor)
                .div(wantLockedTotal)
                .div(entranceFeeFactorMax);
        }
        sharesTotal = sharesTotal.add(sharesAdded);

        if (isAutoComp) {
            _farm();
        } else {
            wantLockedTotal = wantLockedTotal.add(_wantAmt);
        }

        return sharesAdded;
    }
```

修复状态: 该问题在 commit: b27f2cafcf51667726bd70220420707c89b75975 中进行了修复。

5. 审计结果

5.1 结论

审计结果: 通过

审计编号: 0X002103170002

审计日期: 2021 年 03 月 17 日

审计团队: 慢雾安全团队

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 1 个高危漏洞，2 个中危漏洞，2 个增强建议。目前权限过大的问题已经进行修复，已经将 Owner 和 Gov 权限移交给 timelock 合约。

6. 声明

厦门慢雾科技有限公司(下文简称 ”慢雾 ”) 仅就本报告出具前项目方已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后项目方发生或存在的未知漏洞及安全事件，慢雾无法判断其安全状况，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，慢雾仅对该项目的安全情况进行约定内的安全审计并出具了本报告，慢雾不对该项目背景及其他情况进行负责。

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