知道创宇

密级 ★ | 项目组公开

智能合约审计报告

安全状态

安全

★ ★ ★ ★ ★

主测人： 知道创宇区块链安全研究团队

版本说明

修订内容	时间	修订者	版本号
编写文档	20210531	知道创宇区块链安全研究团队	V2.0

文档信息

文档名称	文档版本	报告编号	保密级别
NBF 智能合约审计报告	V2.0	c470f04297494f55b37e552f6a6179b9	项目组公开

声明

目录

1. 综述

本次报告有效测试时间是从 2021 年 5 月 7 日开始到 2021 年 5 月 31 日结束，在此期间针对 **NBF 智能合约控制器、质押矿池、策略和金库合约代码**的安全性和规范性进行审计并以此作为报告统计依据。

本次智能合约安全审计的范围，不包括外部合约调用，不包含未来可能出现的新型攻击方式，不包含合约升级或篡改后的代码（随着项目方的发展，智能合约可能会增加新的 pool、新的功能模块，新的外部合约调用等），不包含前端安全与服务器安全。

此次测试中，知道创宇工程师对智能合约的常见漏洞（见第三章节）进行了全面的分析，综合评定为**通过**。

本次智能合约安全审计结果： 通过

由于本次测试过程在非生产环境下进行，所有代码均为最新备份，测试过程均与相关接口人进行沟通，并在操作风险可控的情况下进行相关测试操作，以规避测试过程中的生产运营风险、代码安全风险。

本次审计的报告信息：

报告编号：c470f04297494f55b37e552f6a6179b9

报告查询地址链接:

https://attest.im/attestation/searchResult?qurey=c470f04297494f55b37e552f6a6179b9

本次审计的目标信息：

条目	描述	
项目名称	NBF	
代码类型	DeFi 协议代码、BSC 智能合约代码	
合约地址	Controller	0x8d6AdC4A7d6b87e37a8DC5Ee6ee51601d5ff0498

	Vault:bnb-cake	0x6eC9f515F0c8803f8b2D49dE61d794eDf2A0119d
	Strategy:bnb-cake	0xeB5355e96C6833b606897277689be666aB924e89
	Vault:BUSD-bnb	0x0B1a3904B519546C237f2FF34ba0f3b4eFB99919
	Strategy:BUSD-bnb	0xd8A35C2F7Cd0657AD5898Cc076bb98BE244Dd8Ec
	Vault:btcb-busd	0x5932665A95d97e8830ef48f0E11a640D6ceEEeA5
	Strategy:btcb-busd	0x107bbf40DCe15b5f98F408c91f7158d36B764721
	Vault:usdc-busd	0x4e759842E6eDE8df415432f2fd6b4CBb5e2557D3
	Strategy:usdc-busd	0x3cceFBd61158441E64686f6329C22da9e21621B6
	Vault:dai-busd	0x16638EF0C74906F8410e89645e1Da54C5c6E33E1
	Strategy:dai-busd	0x4f55E7677408f76F4fb12335DE8B87720072DEcF
	Vault:usdt-busd	0x8Ff463d46bA4697612D0Dac626eeBb3F0193e2a6
	Strategy:usdt-busd	0xbA7556f847E8f98EF56413c510F929d50A99c617
代码语言	Solidity	

合约文件及哈希：

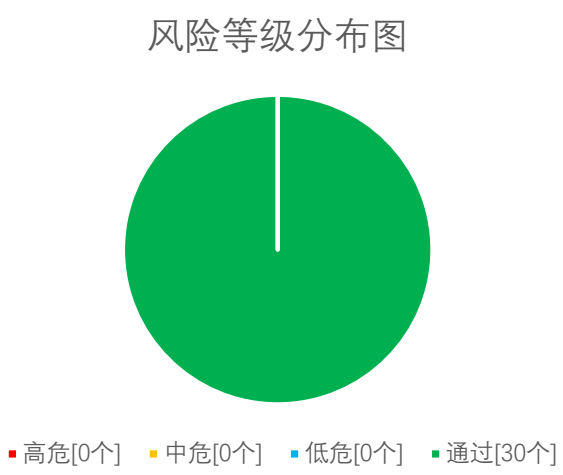
合约文件	MD5
Controller.sol	97f652f62f7933e439bcf2fa1ffcd31d
StrategyLP.sol	c991182f4e46fb2ffc5541db468833f6
zVault.sol	01dce63e29d83f38bfc95b7966cc00d2

2. 代码漏洞分析

2.1 漏洞等级分布

本次漏洞风险按等级统计：

安全风险等级个数统计表			
高危	中危	低危	通过
0	0	0	30



2.2 审计结果汇总说明

审计结果			
审计项目	审计内容	状态	描述
业务安全性检测	控制器合约	通过	经检测，不存在安全问题。
	策略合约	通过	经检测，不存在安全问题。
	金库合约	通过	经检测，不存在安全问题。
代码基本漏洞检测	编译器版本安全	通过	经检测，不存在该安全问题。
	冗余代码	通过	经检测，不存在该安全问题。
	安全算数库的使用	通过	经检测，不存在该安全问题。
	不推荐的编码方式	通过	经检测，不存在该安全问题。
	require/assert 的合理使用	通过	经检测，不存在该安全问题。
	fallback 函数安全	通过	经检测，不存在该安全问题。
	tx.orgin 身份验证	通过	经检测，不存在该安全问题。
	owner 权限控制	通过	经检测，不存在该安全问题。
	gas 消耗检测	通过	经检测，不存在该安全问题。
	call 注入攻击	通过	经检测，不存在该安全问题。
	低级函数安全	通过	经检测，不存在该安全问题。
	增发代币漏洞	通过	经检测，不存在该安全问题。
	访问控制缺陷检测	通过	经检测，不存在该安全问题。
	数值溢出检测	通过	经检测，不存在该安全问题。
	算数精度误差	通过	经检测，不存在该安全问题。
	错误使用随机数检测	通过	经检测，不存在该安全问题。
	不安全的接口使用	通过	经检测，不存在该安全问题。
	变量覆盖	通过	经检测，不存在该安全问题。
	未初始化的存储指针	通过	经检测，不存在该安全问题。

	返回值调用验证	通过	经检测，不存在该安全问题。
	交易顺序依赖检测	通过	经检测，不存在该安全问题。
	时间戳依赖攻击	通过	经检测，不存在该安全问题。
	拒绝服务攻击检测	通过	经检测，不存在该安全问题。
	假充值漏洞检测	通过	经检测，不存在该安全问题。
	重入攻击检测	通过	经检测，不存在该安全问题。
	重放攻击检测	通过	经检测，不存在该安全问题。
	重排攻击检测	通过	经检测，不存在该安全问题。

3. 业务安全性检测

3.1. 控制器合约【通过】

审计分析：控制器合约在 Controller.sol 合约文件中实现，参考 yearn 协议的控制器合约，删除部分功能。

```
contract Controller {
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;

    address public governance;
    address public strategist;

    mapping(address => address) public vaults;
    mapping(address => address) public strategies;
    mapping(address => mapping(address => address)) public converters;

    mapping(address => mapping(address => bool)) public approvedStrategies;

    constructor() {
        governance = msg.sender;
        strategist = msg.sender;
    }

    function setStrategist(address _strategist) public {///
        require(msg.sender == governance, "!governance");
        require(_strategist != address(0), "ADDRESS ERROR!");
        strategist = _strategist;
    }
```

```
function setGovernance(address _governance) public {//
    require(msg.sender == governance, "!governance");
    require(_governance != address(0), "ADDRESS ERROR!");
    governance = _governance;
}

function setVault(address _token, address _vault) public {//
    require(msg.sender == strategist || msg.sender == governance, "!strategist");
    require(_vault != address(0), "ADDRESS ERROR!");
    require(_token != address(0), "ADDRESS ERROR!");
    require(vaults[_token] == address(0), "vault");
    vaults[_token] = _vault;
}

function approveStrategy(address _token, address _strategy) public {
    require(msg.sender == governance, "!governance");
    require(_strategy != address(0), "ADDRESS ERROR!");
    require(_token != address(0), "ADDRESS ERROR!");
    approvedStrategies[_token][_strategy] = true;
}

function revokeStrategy(address _token, address _strategy) public {
    require(msg.sender == governance, "!governance");
    require(_strategy != address(0), "ADDRESS ERROR!");
    require(_token != address(0), "ADDRESS ERROR!");
    approvedStrategies[_token][_strategy] = false;
}

function setConverter(address _input, address _output, address _converter) public {
    require(msg.sender == strategist || msg.sender == governance, "!strategist");
    require(_input != address(0), "ADDRESS ERROR!");
    require(_output != address(0), "ADDRESS ERROR!");
    require(_converter != address(0), "ADDRESS ERROR!");
```

```
        converters[_input][_output] = _converter;
    }

    function setStrategy(address _token, address _strategy) public {//
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(approvedStrategies[_token][_strategy] == true, "!approved");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");

        address _current = strategies[_token];
        if (_current != address(0)) {//knownsec//  若已有策略合约,则先提取所有
            Strategy(_current).withdrawAll();
        }
        strategies[_token] = _strategy;
    }

    function earn(address _token, uint _amount) public {
        require(_token != address(0), "ADDRESS ERROR!");
        address _strategy = strategies[_token];
        address _want = Strategy(_strategy).want();
        if (_want != _token) {
            address converter = converters[_token][_want];
            IERC20(_token).safeTransfer(converter, _amount);
            _amount = Converter(converter).convert(_strategy);
            IERC20(_want).safeTransfer(_strategy, _amount);
        } else {
            IERC20(_token).safeTransfer(_strategy, _amount);
        }
        Strategy(_strategy).deposit();
    }

    function balanceOf(address _token) external view returns (uint) {
        require(_token != address(0), "ADDRESS ERROR!");
```

```
        return Strategy(strategies[_token]).balanceOf();
    }

    function withdrawAll(address _token) public {
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(strategies[_token]).withdrawAll();
    }

    function inCaseTokensGetStuck(address _token, uint _amount) public {
        require(msg.sender == strategist || msg.sender == governance, "!governance");
        require(_token != address(0), "ADDRESS ERROR!");
        IERC20(_token).safeTransfer(msg.sender, _amount);
    }

    function inCaseStrategyTokenGetStuck(address _strategy, address _token) public {
        require(msg.sender == strategist || msg.sender == governance, "!governance");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(_strategy).withdraw(_token);
    }

    function withdraw(address _token, uint _amount) public {
        require(msg.sender == vaults[_token], "!vault");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(strategies[_token]).withdraw(_amount);
    }

    function rewards() public view returns (address) {
        return strategist;
    }
}
```

安全建议：无。

3.2. 策略合约【通过】

审计分析：策略合约在 StrategyLP.sol 合约文件中实现，基于 yearn 协议的策略合约，实现了一个通用的策略合约，策略的输入资产和输出资产均在部署时由构造函数传入。

```
contract StrategyLP{
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;


    address public constant newb = 0x545f90dC35CA1e6129f1fEd354b3e2DF12034261;
    address public constant busd = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;
    address public constant usdt = 0x55d398326f99059fF775485246999027B3197955;
    address public constant uniRouter = 0x10ED43C718714eb63d5aA57B78B54704E256024E;

    uint256 public strategistReward = 10;
    uint256 public restake = 90;
    uint256 public withdrawalFee = 500;
    uint256 public constant FEE_DENOMINATOR = 10000;

    address public out;

    address public pool;
    uint256 public pid;
    address public want;

    address public token0Address;
    address public token1Address;
```

```
address public governance;
address public controller;
address public strategist;

mapping(address => bool) public farmers;

constructor(
    address _controller,
    address _pool,
    uint _pid,
    address _want,
    address _out,
    address _token0Address,
    address _token1Address
) {
    governance = msg.sender;
    strategist = 0x243508EeF0ADc429A520c4a24f7AfdcC532edBC7;
    controller = _controller;
    pool = _pool;
    pid = _pid;
    want = _want;
    out = _out;
    token0Address = _token0Address;
    token1Address = _token1Address;

    doApprove();
}
.....
function harvest() public {

    require(!Address.isContract(msg.sender),"!contract");//knownsec// 校验非合约调用
    dRewards(pool).withdraw(pid,0);
```

```
        uint256 _2reward = IERC20(out).balanceOf(address(this)).mul(strategistReward).div(100);//knownsec// 10%
        uint256 _2want = IERC20(out).balanceOf(address(this)).mul(restake).div(100);//knownsec// 90%
        if(_2reward > 0)
        {//knownsec// out<>busd<>newb
            address[] memory path = new address[](2);
                path[0] = out;
                path[1] = busd;
                Uni(uniRouter).swapExactTokensForTokens(
                    _2reward,
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
                );
                IERC20(busd).safeApprove(uniRouter, 0);
                IERC20(busd).safeApprove(uniRouter, uint256(-1));
                path[0] = busd;
                path[1] = newb;
                Uni(uniRouter).swapExactTokensForTokens(
                    IERC20(busd).balanceOf(address(this)),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
                );
            uint256 _2strategistReward = IERC20(newb).balanceOf(address(this));

            IERC20(newb).safeTransfer(strategist, _2strategistReward);

        }
```

```
if(_2want > 0)
{//knownsec// out 一半转为token0 一半转为token1
    if (out != token0Address) {
        // Swap half earned to token0
        if(token0Address == busd)
        {//knownsec// out<>busd
            address[] memory path = new address[](2);
            path[0] = out;
            path[1] = busd;
            Uni(uniRouter).swapExactTokensForTokens(
                    _2want.div(2),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
            );
        }else if(token0Address == usdt)
        {//knownsec// out<>busd<>usdt
            address[] memory path = new address[](3);
            path[0] = out;
            path[1] = busd;
            path[2] = usdt;
            Uni(uniRouter).swapExactTokensForTokens(
                    _2want.div(2),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
            );
        }else{//knownsec// out<>busd<>usdt<>token0
            address[] memory path = new address[](4);
            path[0] = out;
            path[1] = busd;
```

```
                path[2] = usdt;
                path[3] = token0Address;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }
        }

    if (out != token1Address) {
        // Swap half earned to token1
            if(token1Address == busd)
            {//knownsec// out<>busd
                address[] memory path = new address[](2);
                path[0] = out;
                path[1] = busd;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }else if(token1Address == usdt)
            {//knownsec// out<>busd<>usdt
                address[] memory path = new address[](3);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                Uni(uniRouter).swapExactTokensForTokens(
```

```
                    _2want.div(2),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
                );
            }else{//knownsec// out<>busd<>usdt<>token1
                address[] memory path = new address[](4);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                path[3] = token1Address;
                Uni(uniRouter).swapExactTokensForTokens(
                    _2want.div(2),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
                );
            }
        }

        uint256 token0Amt = IERC20(token0Address).balanceOf(address(this));
        uint256 token1Amt = IERC20(token1Address).balanceOf(address(this));
        if (token0Amt > 0 && token1Amt > 0) {//knownsec//  添加 token0-token1 流动性
            IERC20(token0Address).safeApprove(uniRouter, 0);
            IERC20(token0Address).safeApprove(uniRouter, uint256(-1));
            IERC20(token1Address).safeApprove(uniRouter, 0);
            IERC20(token1Address).safeApprove(uniRouter, uint256(-1));
            Uni(uniRouter).addLiquidity(
                token0Address,
                token1Address,
                token0Amt,
```

```
                    token1Amt,
                    0,
                    0,
                    address(this),
                    block.timestamp.add(1800)
                );
            }
        }

        _deposit();
    }
    ......
}
```

安全建议：无。

3.3. 金库合约【通过】

审计分析：金库合约在 zVault.sol 合约文件中实现，实现了一个通用的金库合约，金库保存的代币在部署时由构造函数传入。

```
contract zVault is ERC20, ERC20Detailed {
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;

    IERC20 public token;

    uint public min = 10000;
    uint public constant max = 10000;
    uint public earnLowerlimit; //池内空余资金到这个值就自动 earn

    address public governance;
```

```
    address public controller;

    constructor (address _token,uint _earnLowerlimit ,address _controller) public
ERC20Detailed(
        string(abi.encodePacked("newb ", ERC20Detailed(_token).name())),
        string(abi.encodePacked("n", ERC20Detailed(_token).symbol())),
        ERC20Detailed(_token).decimals()
    ) {
        token = IERC20(_token);
        governance = msg.sender;
        controller = _controller;
        earnLowerlimit = _earnLowerlimit;
    }
    .....
}
```

安全建议：无。

4. 代码基本漏洞检测

4.1. 编译器版本安全【通过】

检查合约代码实现中是否使用了安全的编译器版本

检测结果：经检测，智能合约代码中制定了编译器版本^0.5.16、>=0.7.0、^0.7.3 等，不存在该安全问题。

安全建议：无。

4.2. 冗余代码【通过】

检查合约代码实现中是否包含冗余代码

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.3. 安全算数库的使用【通过】

检查合约代码实现中是否使用了 SafeMath 安全算数库

检测结果：经检测，智能合约代码中已使用 SafeMath 安全算数库，不存在该安全问题。

安全建议：无。

4.4. 不推荐的编码方式【通过】

检查合约代码实现中是否有官方不推荐或弃用的编码方式

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.5. require/assert 的合理使用【通过】

检查合约代码实现中 require 和 assert 语句使用的合理性

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.6. fallback 函数安全【通过】

检查合约代码实现中是否正确使用 fallback 函数

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.7. tx.origin 身份验证【通过】

tx.origin 是 Solidity 的一个全局变量，它遍历整个调用栈并返回最初发送调用（或事务）的帐户的地址。在智能合约中使用此变量进行身份验证会使合约容易受到类似网络钓鱼的攻击。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.8. owner 权限控制【通过】

检查合约代码实现中的 owner 是否具有过高的权限。例如，任意修改其他账户余额等。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.9. gas 消耗检测【通过】

检查 gas 的消耗是否超过区块最大限制

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.10. call 注入攻击【通过】

call 函数调用时，应该做严格的权限控制，或直接写死 call 调用的函数。

检测结果：经检测，智能合约不存在此漏洞。

安全建议：无。

4.11. 低级函数安全【通过】

检查合约代码实现中低级函数（call/delegatecall）的使用是否存在安全漏洞

call 函数的执行上下文是在被调用的合约中；而 delegatecall 函数的执行上下文是在当前调用该函数的合约中

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.12. 增发代币漏洞【通过】

检查在初始化代币总量后，代币合约中是否存在可能使代币总量增加的函数。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.13. 访问控制缺陷检测【通过】

合约中不同函数应设置合理的权限

检查合约中各函数是否正确使用了 public、private 等关键词进行可见性修饰，检查合约是否正确定义并使用了 modifier 对关键函数进行访问限制，避免越权导致的问题。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.14. 数值溢出检测【通过】

智能合约中的算数问题是指整数溢出和整数下溢。

Solidity 最多能处理 256 位的数字（2^256-1），最大数字增加 1 会溢出得到 0。同样，当数字为无符号类型时，0 减去 1 会下溢得到最大数字值。

整数溢出和下溢不是一种新类型的漏洞，但它们在智能合约中尤其危险。溢出情况会导致不正确的结果，特别是如果可能性未被预期，可能会影响程序的可靠性和安全性。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.15. 算术精度误差【通过】

Solidity 作为一门编程语言具备和普通编程语言相似的数据结构设计，比如：变量、常量、函数、数组、函数、结构体等等，Solidity 和普通编程语言也有一个较大的区别——Solidity 没有浮点型，且 Solidity 所有的数值运算结果都只会是整数，不会出现小数的情况，同时也不允许定义小数类型数据。合约中的数值运算必不可少，而数值运算的设计有可能造成相对误差，例如同级运算：5/2*10=20，而 5*10/2=25，从而产生误差，在数据更大时产生的误差也会更大，更明显。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.16. 错误使用随机数【通过】

智能合约中可能需要使用随机数，虽然 Solidity 提供的函数和变量可以访问明显难以预测的值，如 block.number 和 block.timestamp，但是它们通常或者比看起来更公开，或者受到矿工的影响，即这些随机数在一定程度上是可预测的，所以恶意用户通常可以复制它并依靠其不可预知性来攻击该功能。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.17. 不安全的接口使用【通过】

检查合约代码实现中是否使用了不安全的接口

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.18. 变量覆盖【通过】

检查合约代码实现中是否存在变量覆盖导致的安全问题

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.19. 未初始化的储存指针【通过】

在 solidity 中允许一个特殊的数据结构为 struct 结构体，而函数内的局部变量默认使用 storage 或 memory 储存。

而存在 storage(存储器)和 memory(内存)是两个不同的概念，solidity 允许指针指向一个未初始化的引用，而未初始化的局部 stroage 会导致变量指向其他储存变量，导致变量覆盖，甚至其他更严重的后果，在开发中应该避免在函数中初始化 struct 变量。

检测结果：经检测，智能合约代码不存在该问题。

安全建议：无。

4.20. 返回值调用验证【通过】

此问题多出现在和转币相关的智能合约中，故又称作静默失败发送或未经检查发送。

在 Solidity 中存在 transfer()、send()、call.value()等转币方法，都可以用于向某一地址发送 Ether，其区别在于： transfer 发送失败时会 throw，并且进行状态回滚；只会传递 2300gas 供调用，防止重入攻击；send 发送失败时会返回 false；只会传递 2300gas 供调用，防止重入攻击；call.value 发送失败时会返回 false；

传递所有可用 gas 进行调用（可通过传入 gas_value 参数进行限制），不能有效防止重入攻击。

如果在代码中没有检查以上 send 和 call.value 转币函数的返回值，合约会继续执行后面的代码，可能由于 Ether 发送失败而导致意外的结果。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.21. 交易顺序依赖【通过】

由于矿工总是通过代表外部拥有地址（EOA）的代码获取 gas 费用，因此用户可以指定更高的费用以便更快地开展交易。由于以太坊区块链是公开的，每个人都可以看到其他人未决交易的内容。这意味着，如果某个用户提交了一个有价值的解决方案，恶意用户可以窃取该解决方案并以较高的费用复制其交易，以抢占原始解决方案。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.22. 时间戳依赖攻击【通过】

数据块的时间戳通常来说都是使用矿工的本地时间，而这个时间大约能有 900 秒的范围波动，当其他节点接受一个新区块时，只需要验证时间戳是否晚于之前的区块并且与本地时间误差在 900 秒以内。一个矿工可以通过设置区块的时间戳来尽可能满足有利于他的条件来从中获利。

检查合约代码实现中是否存在有依赖于时间戳的关键功能

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.23. 拒绝服务攻击【通过】

在以太坊的世界中，拒绝服务是致命的，遭受该类型攻击的智能合约可能永远无法恢复正常工作状态。导致智能合约拒绝服务的原因可能有很多种，包括在作为交易接收方时的恶意行为，人为增加计算功能所需 gas 导致 gas 耗尽，滥用访问控制访问智能合约的 private 组件，利用混淆和疏忽等等。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.24. 假充值漏洞【通过】

在代币合约的 transfer 函数对转账发起人(msg.sender)的余额检查用的是 if 判断方式，当 balances[msg.sender] < value 时进入 else 逻辑部分并 return false，最终没有抛出异常，我们认为仅 if/else 这种温和的判断方式在 transfer 这类敏感函数场景中是一种不严谨的编码方式。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.25. 重入攻击检测【通过】

重入漏洞是最著名的以太坊智能合约漏洞，曾导致了以太坊的分叉（The DAO hack）。

Solidity 中的 call.value()函数在被用来发送 Ether 的时候会消耗它接收到的所有 gas，当调用 call.value()函数发送 Ether 的操作发生在实际减少发送者账户的余额之前时，就会存在重入攻击的风险。

检测结果：经检测，智能合约代码中不存在该安全问题。

安全建议：无。

4.26. 重放攻击检测【通过】

合约中如果涉及委托管理的需求，应注意验证的不可复用性，避免重放攻击

在资产管理体系中，常有委托管理的情况，委托人将资产给受托人管理，委托人支付一定的费用给受托人。这个业务场景在智能合约中也比较普遍。。

检测结果：经检测，智能合约不存在此漏洞。

安全建议：无。

4.27. 重排攻击检测【通过】

重排攻击是指矿工或其他方试图通过将自己的信息插入列表(list)或映射(mapping)中来与智能合约参与者进行“竞争”，从而使攻击者有机会将自己的信息存储到合约中。

检测结果:经检测，智能合约代码中不存在相关漏洞。

安全建议:无。

5. 附录 A：合约代码

本次测试代码来源：

Controller.sol

```
/**
 *Submitted for verification at BscScan.com on 2021-05-17
*/

pragma solidity >=0.7.0;

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint256 c = a / b;

        return c;
    }
}

library Address {
    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
        return (codehash != 0x0 && codehash != accountHash);
    }
    function toPayable(address account) internal pure returns (address payable) {
        return address(uint160(account));
    }
}

library SafeERC20 {
    using SafeMath for uint256;
    using Address for address;

    function safeTransfer(IERC20 token, address to, uint256 value) internal {
        callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to, value));
    }

    function safeTransferFrom(IERC20 token, address from, address to, uint256 value) internal {
```

```
        callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to, value));
    }

    function safeApprove(IERC20 token, address spender, uint256 value) internal {
        require((value == 0) || (token.allowance(address(this), spender) == 0),
            "SafeERC20: approve from non-zero to non-zero allowance"
        );
        callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));
    }
    function callOptionalReturn(IERC20 token, bytes memory data) private {
        require(address(token).isContract(), "SafeERC20: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = address(token).call(data);
        require(success, "SafeERC20: low-level call failed");

        if (returndata.length > 0) { // Return data is optional
            // solhint-disable-next-line max-line-length
            require(abi.decode(returndata, (bool)), "SafeERC20: ERC20 operation did not succeed");
        }
    }
}

interface Strategy {
    function want() external view returns (address);
    function deposit() external;
    function withdraw(address) external;
    function withdraw(uint) external;
    function withdrawAll() external returns (uint);
    function balanceOf() external view returns (uint);
}

interface Converter {
    function convert(address) external returns (uint);
}

contract Controller {
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;

    address public governance;
    address public strategist;

    mapping(address => address) public vaults;
    mapping(address => address) public strategies;
    mapping(address => mapping(address => address)) public converters;

    mapping(address => mapping(address => bool)) public approvedStrategies;

    constructor() {
        governance = msg.sender;
        strategist = msg.sender;
    }

    function setStrategist(address _strategist) public {//
        require(msg.sender == governance, "!governance");
        require(_strategist != address(0), "ADDRESS ERROR!");
        strategist = _strategist;
    }

    function setGovernance(address _governance) public {//
        require(msg.sender == governance, "!governance");
        require(_governance != address(0), "ADDRESS ERROR!");
        governance = _governance;
    }

    function setVault(address _token, address _vault) public {//
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(_vault != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");
        require(vaults[_token] == address(0), "vault");
        vaults[_token] = _vault;
    }

    function approveStrategy(address _token, address _strategy) public {
        require(msg.sender == governance, "!governance");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");
        approvedStrategies[_token][_strategy] = true;
    }

    function revokeStrategy(address _token, address _strategy) public {
        require(msg.sender == governance, "!governance");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");
        approvedStrategies[_token][_strategy] = false;
```

```
    }

    function setConverter(address _input, address _output, address _converter) public {
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(_input != address(0), "ADDRESS ERROR!");
        require(_output != address(0), "ADDRESS ERROR!");
        require(_converter != address(0), "ADDRESS ERROR!");
        converters[_input][_output] = _converter;
    }

    function setStrategy(address _token, address _strategy) public {//
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(approvedStrategies[_token][_strategy] == true, "!approved");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");

        address _current = strategies[_token];
        if (_current != address(0)) {//knownsec//  若已有策略合约,则先提取所有
            Strategy(_current).withdrawAll();
        }
        strategies[_token] = _strategy;
    }

    function earn(address _token, uint _amount) public {
        require(_token != address(0), "ADDRESS ERROR!");
        address _strategy = strategies[_token];
        address _want = Strategy(_strategy).want();
        if (_want != _token) {
            address converter = converters[_token][_want];
            IERC20(_token).safeTransfer(converter, _amount);
            _amount = Converter(converter).convert(_strategy);
            IERC20(_want).safeTransfer(_strategy, _amount);
        } else {
            IERC20(_token).safeTransfer(_strategy, _amount);
        }
        Strategy(_strategy).deposit();
    }

    function balanceOf(address _token) external view returns (uint) {
        require(_token != address(0), "ADDRESS ERROR!");
        return Strategy(strategies[_token]).balanceOf();
    }

    function withdrawAll(address _token) public {
        require(msg.sender == strategist || msg.sender == governance, "!strategist");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(strategies[_token]).withdrawAll();
    }

    function inCaseTokensGetStuck(address _token, uint _amount) public {
        require(msg.sender == strategist || msg.sender == governance, "!governance");
        require(_token != address(0), "ADDRESS ERROR!");
        IERC20(_token).safeTransfer(msg.sender, _amount);
    }

    function inCaseStrategyTokenGetStuck(address _strategy, address _token) public {
        require(msg.sender == strategist || msg.sender == governance, "!governance");
        require(_strategy != address(0), "ADDRESS ERROR!");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(_strategy).withdraw(_token);
    }

    function withdraw(address _token, uint _amount) public {
        require(msg.sender == vaults[_token], "!vault");
        require(_token != address(0), "ADDRESS ERROR!");
        Strategy(strategies[_token]).withdraw(_amount);
    }

    function rewards() public view returns (address) {
        return strategist;
    }
}
```

StrategyLp.sol

```
/**
 *Submitted for verification at BscScan.com on 2021-05-17
*/

pragma solidity ^0.7.3;
pragma experimental ABIEncoderV2;

interface IERC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address account) external view returns (uint256);
```

```
    function transfer(address recipient, uint256 amount)
        external
        returns (bool);

    function allowance(address owner, address spender)
        external
        view
        returns (uint256);

    function approve(address spender, uint256 amount) external returns (bool);

    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );
}

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    function sub(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;
        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    function div(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;

        return c;
    }
}

library Address {
    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
        bytes32 accountHash =
            0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            codehash := extcodehash(account)
        }
        return (codehash != accountHash && codehash != 0x0);
    }

    function toPayable(address account)
```

```
        internal
        pure
        returns (address payable)
    {
        return address(uint160(account));
    }
}

library SafeERC20 {
    using SafeMath for uint256;
    using Address for address;

    function safeTransfer(
        IERC20 token,
        address to,
        uint256 value
    ) internal {
        callOptionalReturn(
            token,
            abi.encodeWithSelector(token.transfer.selector, to, value)
        );
    }

    function safeTransferFrom(
        IERC20 token,
        address from,
        address to,
        uint256 value
    ) internal {
        callOptionalReturn(
            token,
            abi.encodeWithSelector(token.transferFrom.selector, from, to, value)
        );
    }

    function safeApprove(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        // safeApprove should only be called when setting an initial allowance,
        // or when resetting it to zero. To increase and decrease it, use
        // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
        // solhint-disable-next-line max-line-length
        require(
            (value == 0) || (token.allowance(address(this), spender) == 0),
            "SafeERC20: approve from non-zero to non-zero allowance"
        );
        callOptionalReturn(
            token,
            abi.encodeWithSelector(token.approve.selector, spender, value)
        );
    }

    function safeIncreaseAllowance(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        uint256 newAllowance =
            token.allowance(address(this), spender).add(value);
        callOptionalReturn(
            token,
            abi.encodeWithSelector(
                token.approve.selector,
                spender,
                newAllowance
            )
        );
    }

    function safeDecreaseAllowance(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        uint256 newAllowance =
            token.allowance(address(this), spender).sub(
                value,
                "SafeERC20: decreased allowance below zero"
            );
        callOptionalReturn(
            token,
            abi.encodeWithSelector(
                token.approve.selector,
                spender,
                newAllowance
```

```
                )
            );
        }

        function callOptionalReturn(IERC20 token, bytes memory data) private {
            require(address(token).isContract(), "SafeERC20: call to non-contract");

            // solhint-disable-next-line avoid-low-level-calls
            (bool success, bytes memory returndata) = address(token).call(data);
            require(success, "SafeERC20: low-level call failed");

            if (returndata.length > 0) {
                // Return data is optional
                // solhint-disable-next-line max-line-length
                require(
                    abi.decode(returndata, (bool)),
                    "SafeERC20: ERC20 operation did not succeed"
                );
            }
        }
    }

    //
    interface IController {
        function withdraw(address, uint256) external;

        function balanceOf(address) external view returns (uint256);

        function earn(address, uint256) external;

        function want(address) external view returns (address);

        function rewards() external view returns (address);

        function vaults(address) external view returns (address);

        function strategies(address) external view returns (address);
    }

    //
    interface Uni {
        function swapExactTokensForTokens(
            uint256,
            uint256,
            address[] calldata,
            address,
            uint256
        ) external;
        function addLiquidity(
            address tokenA,
            address tokenB,
            uint amountADesired,
            uint amountBDesired,
            uint amountAMin,
            uint amountBMin,
            address to,
            uint deadline
        ) external returns (uint amountA, uint amountB, uint liquidity);
    }


    struct UserInfo {
            uint256 amount;     // How many LP tokens the user has provided.
            uint256 rewardDebt; // Reward debt. See explanation below.
            //
            // We do some fancy math here. Basically, any point in time, the amount of CAKEs
            // entitled to a user but is pending to be distributed is:
            //
            //     pending reward = (user.amount * pool.accCakePerShare) - user.rewardDebt
            //
            // Whenever a user deposits or withdraws LP tokens to a pool. Here's what happens:
            //     1. The pool's `accCakePerShare` (and `lastRewardBlock`) gets updated.
            //     2. User receives the pending reward sent to his/her address.
            //     3. User's `amount` gets updated.
            //     4. User's `rewardDebt` gets updated.
        }


    interface dRewards {
        function userInfo(uint256,address) external view returns(UserInfo calldata);
        // function stake(address _pair, uint256 _amount) external;
        // function unstake(address _pair, uint256 _amount) external;
        // function pendingToken(address _pair, address _user) external returns (uint256);

        function deposit(uint256 _pid, uint256 _amount) external;
        function withdraw(uint256 _pid, uint256 _amount) external;
        function pendingCake(uint256 _pid, address _user) external view returns (uint256);
```

```
}

interface IWETH {
    function deposit() external payable;

    function transfer(address to, uint256 value) external returns (bool);

    function withdraw(uint256 amt) external;
}

contract StrategyLP{
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;


    address public constant newb = 0x545f90dC35CA1e6129f1fEd354b3e2DF12034261;
    address public constant busd = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;
    address public constant usdt = 0x55d398326f99059fF775485246999027B3197955;
    address public constant uniRouter = 0x10ED43C718714eb63d5aA57B78B54704E256024E;

    uint256 public strategistReward = 10;
    uint256 public restake = 90;
    uint256 public withdrawalFee = 500;
    uint256 public constant FEE_DENOMINATOR = 10000;

    address public out;

    address public pool;
    uint256 public pid;
    address public want;

    address public token0Address;
    address public token1Address;

    address public governance;
    address public controller;
    address public strategist;

    mapping(address => bool) public farmers;

    constructor(
        address _controller,
        address _pool,
        uint _pid,
        address _want,
        address _out,
        address _token0Address,
        address _token1Address
    ) {
        governance = msg.sender;
        strategist = 0x243508EeF0ADc429A520c4a24f7AfdcC532edBC7;
        controller = _controller;
        pool = _pool;
        pid = _pid;
        want = _want;
        out = _out;
        token0Address = _token0Address;
        token1Address = _token1Address;

        doApprove();
    }

    function doApprove () internal{
        IERC20(out).safeApprove(uniRouter, 0);
        IERC20(out).safeApprove(uniRouter, uint(-1));
    }

    function addFarmer(address f) public {//knownsec//  添加farmer
        require(
            msg.sender == governance || msg.sender == strategist,
            "!authorized"
        );
        require(f != address(0), "address error");
        farmers[f] = true;
    }

    function removeFarmer(address f) public {//knownsec//  移除farmer
        require(
            msg.sender == governance || msg.sender == strategist,
            "!authorized"
        );
        farmers[f] = false;
    }

    function setGovernance(address _governance) external {
```

```
        require(msg.sender == governance, "!governance");
        require(_governance != address(0), "address error");
        governance = _governance;
    }

    function setStrategist(address _strategist) external {
        require(
            msg.sender == governance || msg.sender == strategist,
            "!authorized"
        );
        require(_strategist != address(0), "address error");
        strategist = _strategist;
    }

    function setWithdrawalFee(uint256 _withdrawalFee) external {
        require(msg.sender == governance, "!governance");
        withdrawalFee = _withdrawalFee;
    }

    function setStrategistReward(uint256 _strategistReward) external {
        require(msg.sender == governance, "!governance");
        strategistReward = _strategistReward;
    }

    //
    function setRestake(uint256 _restake) external {
        require(msg.sender == governance, "!governance");
        restake = _restake;
    }

    function balanceOfPool() public view returns (uint256) {
        UserInfo memory user = dRewards(pool).userInfo(pid,address(this));
        return user.amount;
    }

    function balanceOfWant() public view returns (uint256) {
        return IERC20(want).balanceOf(address(this));
    }

    function balanceOf() public view returns (uint256) {
        return balanceOfWant().add(balanceOfPool());
    }

    modifier onlyBenevolent {
        require(
            farmers[msg.sender] ||
                msg.sender == governance ||
                msg.sender == strategist
        );
        _;
    }

    function getNumOfRewards() public view returns (uint256 pending) {
        pending = dRewards(pool).pendingCake(pid,address(this));
    }


    function harvest() public {

        require(!Address.isContract(msg.sender),"!contract");//knownsec//  校验非合约调用
        dRewards(pool).withdraw(pid,0);

        uint256 _2reward = IERC20(out).balanceOf(address(this)).mul(strategistReward).div(100);//knownsec//
10%
        uint256 _2want = IERC20(out).balanceOf(address(this)).mul(restake).div(100);//knownsec// 90%
        if(_2reward > 0)
        {//knownsec// out<>busd<>newb
            address[] memory path = new address[](2);
                path[0] = out;
                path[1] = busd;
                Uni(uniRouter).swapExactTokensForTokens(
                    _2reward,
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
                );
                IERC20(busd).safeApprove(uniRouter, 0);
                IERC20(busd).safeApprove(uniRouter, uint256(-1));
                path[0] = busd;
                path[1] = newb;
                Uni(uniRouter).swapExactTokensForTokens(
                    IERC20(busd).balanceOf(address(this)),
                    uint256(0),
                    path,
                    address(this),
                    block.timestamp.add(1800)
```

```
            );
        uint256 _2strategistReward = IERC20(newb).balanceOf(address(this));

        IERC20(newb).safeTransfer(strategist, _2strategistReward);

    }
    if(_2want > 0)
    {//knownsec// out 一半转为token0 一半转为token1
        if (out != token0Address) {
            // Swap half earned to token0
            if(token0Address == busd)
            {//knownsec// out<>busd
                address[] memory path = new address[](2);
                path[0] = out;
                path[1] = busd;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }else if(token0Address == usdt)
            {//knownsec// out<>busd<>usdt
                address[] memory path = new address[](3);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }else{//knownsec// out<>busd<>usdt<>token0
                address[] memory path = new address[](4);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                path[3] = token0Address;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }
        }

    if (out != token1Address) {
        // Swap half earned to token1
            if(token1Address == busd)
            {//knownsec// out<>busd
                address[] memory path = new address[](2);
                path[0] = out;
                path[1] = busd;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }else if(token1Address == usdt)
            {//knownsec// out<>busd<>usdt
                address[] memory path = new address[](3);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
                        path,
                        address(this),
                        block.timestamp.add(1800)
                );
            }else{//knownsec// out<>busd<>usdt<>token1
                address[] memory path = new address[](4);
                path[0] = out;
                path[1] = busd;
                path[2] = usdt;
                path[3] = token1Address;
                Uni(uniRouter).swapExactTokensForTokens(
                        _2want.div(2),
                        uint256(0),
```

```
                                path,
                                address(this),
                                block.timestamp.add(1800)
                        );
                }
        }

        uint256 token0Amt = IERC20(token0Address).balanceOf(address(this));
        uint256 token1Amt = IERC20(token1Address).balanceOf(address(this));
        if (token0Amt > 0 && token1Amt > 0) {//knownsec//  添加 token0-token1 流动性
                IERC20(token0Address).safeApprove(uniRouter, 0);
                IERC20(token0Address).safeApprove(uniRouter, uint256(-1));
                IERC20(token1Address).safeApprove(uniRouter, 0);
                IERC20(token1Address).safeApprove(uniRouter, uint256(-1));
                Uni(uniRouter).addLiquidity(
                        token0Address,
                        token1Address,
                        token0Amt,
                        token1Amt,
                        0,
                        0,
                        address(this),
                        block.timestamp.add(1800)
                );
            }
        }

        _deposit();
}

function deposit() public {

        _deposit();
}

function _deposit() internal returns (uint) {//knownsec//  质押本合约全部 want
        uint256 _want = IERC20(want).balanceOf(address(this));
        if (_want > 0) {
                IERC20(want).safeApprove(pool, 0);
                IERC20(want).safeApprove(pool, _want);
                dRewards(pool).deposit(pid,_want);
        }
        return _want;
}

function _withdrawSome(uint256 _amount) internal returns (uint256) {
        uint _before = IERC20(want).balanceOf(address(this));
        dRewards(pool).withdraw(pid, _amount);
        uint _after = IERC20(want).balanceOf(address(this));
        uint _withdrew = _after.sub(_before);
        return _withdrew;
}

function withdrawAll() external returns (uint256 balance) {
        require(msg.sender == controller, "!controller");
        _withdrawAll();

        balance = IERC20(want).balanceOf(address(this));

        address _vault = IController(controller).vaults(address(want));
        require(_vault != address(0), "!vault"); // additional protection so we don't burn the funds
        IERC20(want).safeTransfer(_vault, balance);
}

function _withdrawAll() internal {
        uint256 wamount = balanceOfPool();
        dRewards(pool).withdraw(pid,wamount);
}

function withdraw(uint256 _amount) external {
        require(msg.sender == controller, "!controller");
        uint256 _balance = IERC20(want).balanceOf(address(this));
        if (_balance < _amount) {
                _amount = _withdrawSome(_amount.sub(_balance));
                _amount = _amount.add(_balance);
        }

        uint256 _fee = _amount.mul(withdrawalFee).div(FEE_DENOMINATOR);//knownsec// 5%

        if (_fee > 0) {
                IERC20(want).safeTransfer(IController(controller).rewards(), _fee);
        }
        address _vault = IController(controller).vaults(address(want));
        require(_vault != address(0), "!vault"); // additional protection so we don't burn the funds
        IERC20(want).safeTransfer(_vault, _amount.sub(_fee));
}
```

```
    function withdraw(IERC20 _asset) external returns (uint256 balance) {
        require(msg.sender == controller, "!controller");
        require(want != address(_asset), "want");
        balance = _asset.balanceOf(address(this));
        _asset.safeTransfer(controller, balance);
    }

}
```

zVault.sol

```
/**
 *Submitted for verification at BscScan.com on 2021-05-17
*/

pragma solidity ^0.5.16;

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract Context {
    constructor () internal { }
    // solhint-disable-previous-line no-empty-blocks

    function _msgSender() internal view returns (address payable) {
        return msg.sender;
    }

    function _msgData() internal view returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}

contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
    constructor () internal {
        _owner = _msgSender();
        emit OwnershipTransferred(address(0), _owner);
    }
    function owner() public view returns (address) {
        return _owner;
    }
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }
    function isOwner() public view returns (bool) {
        return _msgSender() == _owner;
    }
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }
    function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
    }
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
}

contract ERC20 is Context, IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }
```

```
    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }
    function transfer(address recipient, uint256 amount) public returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowances[owner][spender];
    }
    function approve(address spender, uint256 amount) public returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }
    function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
        return true;
    }
    function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
        return true;
    }
    function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
        return true;
    }
    function _transfer(address sender, address recipient, uint256 amount) internal {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }
    function _mint(address account, uint256 amount) internal {
        require(account != address(0), "ERC20: mint to the zero address");

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }
    function _burn(address account, uint256 amount) internal {
        require(account != address(0), "ERC20: burn from the zero address");

        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }
    function _approve(address owner, address spender, uint256 amount) internal {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
    function _burnFrom(address account, uint256 amount) internal {
        _burn(account, amount);
        _approve(account, _msgSender(), _allowances[account][_msgSender()].sub(amount, "ERC20: burn amount exceeds allowance"));
    }
}

contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }
    function name() public view returns (string memory) {
        return _name;
    }
    function symbol() public view returns (string memory) {
        return _symbol;
    }
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}

library SafeMath {
```

```
        function add(uint256 a, uint256 b) internal pure returns (uint256) {
            uint256 c = a + b;
            require(c >= a, "SafeMath: addition overflow");

            return c;
        }
        function sub(uint256 a, uint256 b) internal pure returns (uint256) {
            return sub(a, b, "SafeMath: subtraction overflow");
        }
        function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
            require(b <= a, errorMessage);
            uint256 c = a - b;

            return c;
        }
        function mul(uint256 a, uint256 b) internal pure returns (uint256) {
            if (a == 0) {
                return 0;
            }

            uint256 c = a * b;
            require(c / a == b, "SafeMath: multiplication overflow");

            return c;
        }
        function div(uint256 a, uint256 b) internal pure returns (uint256) {
            return div(a, b, "SafeMath: division by zero");
        }
        function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
            // Solidity only automatically asserts when dividing by 0
            require(b > 0, errorMessage);
            uint256 c = a / b;

            return c;
        }
        function mod(uint256 a, uint256 b) internal pure returns (uint256) {
            return mod(a, b, "SafeMath: modulo by zero");
        }
        function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
            require(b != 0, errorMessage);
            return a % b;
        }
}

library Address {
        function isContract(address account) internal view returns (bool) {
            bytes32 codehash;
            bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
            // solhint-disable-next-line no-inline-assembly
            assembly { codehash := extcodehash(account) }
            return (codehash != 0x0 && codehash != accountHash);
        }
        function toPayable(address account) internal pure returns (address payable) {
            return address(uint160(account));
        }
        function sendValue(address payable recipient, uint256 amount) internal {
            require(address(this).balance >= amount, "Address: insufficient balance");

            // solhint-disable-next-line avoid-call-value
            (bool success, ) = recipient.call.value(amount)("");
            require(success, "Address: unable to send value, recipient may have reverted");
        }
}

library SafeERC20 {
        using SafeMath for uint256;
        using Address for address;

        function safeTransfer(IERC20 token, address to, uint256 value) internal {
            callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to, value));
        }

        function safeTransferFrom(IERC20 token, address from, address to, uint256 value) internal {
            callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to, value));
        }

        function safeApprove(IERC20 token, address spender, uint256 value) internal {
            require((value == 0) || (token.allowance(address(this), spender) == 0),
                "SafeERC20: approve from non-zero to non-zero allowance"
            );
            callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));
        }

        function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {
            uint256 newAllowance = token.allowance(address(this), spender).add(value);
            callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));
        }
```

```
    function safeDecreaseAllowance(IERC20 token, address spender, uint256 value) internal {
        uint256 newAllowance = token.allowance(address(this), spender).sub(value, "SafeERC20: decreased allowance below zero");
        callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));
    }
    function callOptionalReturn(IERC20 token, bytes memory data) private {
        require(address(token).isContract(), "SafeERC20: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = address(token).call(data);
        require(success, "SafeERC20: low-level call failed");

        if (returndata.length > 0) { // Return data is optional
            // solhint-disable-next-line max-line-length
            require(abi.decode(returndata, (bool)), "SafeERC20: ERC20 operation did not succeed");
        }
    }
}

interface Controller {
    function withdraw(address, uint) external;
    function balanceOf(address) external view returns (uint);
    function earn(address, uint) external;
}

contract zVault is ERC20, ERC20Detailed {
    using SafeERC20 for IERC20;
    using Address for address;
    using SafeMath for uint256;

    IERC20 public token;

    uint public min = 10000;
    uint public constant max = 10000;
    uint public earnLowerlimit; //池内空余资金到这个值就自动 earn

    address public governance;
    address public controller;

    constructor (address _token,uint _earnLowerlimit ,address _controller) public ERC20Detailed(
        string(abi.encodePacked("newb ", ERC20Detailed(_token).name())),
        string(abi.encodePacked("n", ERC20Detailed(_token).symbol())),
        ERC20Detailed(_token).decimals()
    ) {
        token = IERC20(_token);
        governance = msg.sender;
        controller = _controller;
        earnLowerlimit = _earnLowerlimit;
    }

    function balance() public view returns (uint) {
        return token.balanceOf(address(this))
                    .add(Controller(controller).balanceOf(address(token)));
    }

    function setMin(uint _min) external {
        require(msg.sender == governance, "!governance");
        min = _min;
    }

    function setGovernance(address _governance) public {
        require(msg.sender == governance, "!governance");
        require(_governance != address(0), "ADDRESS ERROR");
        governance = _governance;
    }

    function setController(address _controller) public {
        require(msg.sender == governance, "!governance");
        require(_controller != address(0), "ADDRESS ERROR");
        controller = _controller;
    }
    function setEarnLowerlimit(uint256 _earnLowerlimit) public{
       require(msg.sender == governance, "!governance");
       earnLowerlimit = _earnLowerlimit;
   }

    // Custom logic in here for how much the vault allows to be borrowed
    // Sets minimum required on-hand to keep small withdrawals cheap
    function available() public view returns (uint) {
        return token.balanceOf(address(this)).mul(min).div(max);
    }

    function earn() public {
        uint _bal = available();
        token.safeTransfer(controller, _bal);
        Controller(controller).earn(address(token), _bal);
```

```
    }

    function depositAll() external {
        deposit(token.balanceOf(msg.sender));
    }

    function deposit(uint _amount) public {//knownsec//  质押
        require(_amount > 0, "error amount");
        uint _pool = balance();
        uint _before = token.balanceOf(address(this));
        token.safeTransferFrom(msg.sender, address(this), _amount);
        uint _after = token.balanceOf(address(this));
        _amount = _after.sub(_before); // Additional check for deflationary tokens
        uint shares = 0;
        if (totalSupply() == 0) {
            shares = _amount;
        } else {
            shares = (_amount.mul(totalSupply())).div(_pool);
        }
        _mint(msg.sender, shares);
        if (token.balanceOf(address(this))>earnLowerlimit){
           earn();
        }
    }

    function withdrawAll() external {
        withdraw(balanceOf(msg.sender));
    }


    // No rebalance implementation for lower fees and faster swaps
    function withdraw(uint _shares) public {//knownsec//  赎回
        require(_shares > 0, "error shares");
        uint r = (balance().mul(_shares)).div(totalSupply());
        _burn(msg.sender, _shares);

        // Check balance
        uint b = token.balanceOf(address(this));
        if (b < r) {
            uint _withdraw = r.sub(b);
            Controller(controller).withdraw(address(token), _withdraw);
            uint _after = token.balanceOf(address(this));
            uint _diff = _after.sub(b);
            if (_diff < _withdraw) {
                r = b.add(_diff);
            }
        }

        token.safeTransfer(msg.sender, r);
    }

    function getPricePerFullShare() public view returns (uint) {
        return balance().mul(1e18).div(totalSupply());
    }
}
```

6. 附录 B：安全风险评级标准

智能合约漏洞评级标准	
漏洞评级	**漏洞评级说明**
高危漏洞	能直接造成代币合约或用户资金损失的漏洞，如：能造成代币价值归零的数值溢出漏洞、能造成交易所损失代币的假充值漏洞、能造成合约账户损失 ETH 或代币的重入漏洞等； 能造成代币合约归属权丢失的漏洞，如：关键函数的访问控制缺陷、call 注入导致关键函数访问控制绕过等； 能造成代币合约无法正常工作的漏洞，如：因向恶意地址发送 ETH 导致的拒绝服务漏洞、因 gas 耗尽导致的拒绝服务漏洞。
中危漏洞	需要特定地址才能触发的高风险漏洞，如代币合约拥有者才能触发的数值溢出漏洞等；非关键函数的访问控制缺陷、不能造成直接资金损失的逻辑设计缺陷等。
低危漏洞	难以被触发的漏洞、触发之后危害有限的漏洞，如需要大量 ETH 或代币才能触发的数值溢出漏洞、触发数值溢出后攻击者无法直接获利的漏洞、通过指定高 gas 触发的事务顺序依赖风险等。

7. 附录 C：智能合约安全审计工具简介

6.1 Manticore

Manticore 是一个分析二进制文件和智能合约的符号执行工具，Manticore 包含一个符号以太坊虚拟机（EVM），一个 EVM 反汇编器/汇编器以及一个用于自动编译和分析 Solidity 的方便界面。它还集成了 Ethersplay，用于 EVM 字节码的 Bit of Traits of Bits 可视化反汇编程序，用于可视化分析。 与二进制文件一样，Manticore 提供了一个简单的命令行界面和一个用于分析 EVM 字节码的 Python API。

6.2 Oyente

Oyente 是一个智能合约分析工具，Oyente 可以用来检测智能合约中常见的 bug，比如 reentrancy、事务排序依赖等等。更方便的是，Oyente 的设计是模块化的，所以这让高级用户可以实现并插入他们自己的检测逻辑，以检查他们的合约中自定义的属性。

6.3 securify.sh

Securify 可以验证以太坊智能合约常见的安全问题，例如交易乱序和缺少输入验证，它在全自动化的同时分析程序所有可能的执行路径，此外，Securify 还具有用于指定漏洞的特定语言，这使 Securify 能够随时关注当前的安全性和其他可靠性问题。

6.4 Echidna

Echidna 是一个为了对 EVM 代码进行模糊测试而设计的 Haskell 库。

6.5 MAIAN

MAIAN 是一个用于查找以太坊智能合约漏洞的自动化工具，Maian 处理合

约的字节码，并尝试建立一系列交易以找出并确认错误。

6.6 ethersplay

ethersplay 是一个 EVM 反汇编器，其中包含了相关分析工具。

6.7 ida-evm

ida-evm 是一个针对以太坊虚拟机（EVM）的 IDA 处理器模块。

6.8 Remix-ide

Remix 是一款基于浏览器的编译器和 IDE，可让用户使用 Solidity 语言构建以太坊合约并调试交易。

6.9 知道创宇区块链安全审计人员专用工具包

知道创宇渗透测试人员专用工具包，由知道创宇渗透测试工程师研发，收集和使用，包含专用于测试人员的批量自动测试工具，自主研发的工具、脚本或利用工具等。

知道创宇

北京知道创宇信息技术股份有限公司

咨询电话 +86(10)400 060 9587

邮　　箱 sec@knownsec.com

官　　网 www.knownsec.com

地　　址 北京市 朝阳区 望京 SOHO T2-B座-2509