智能合约安全审计报告

NUT 智能合约安全审计报告

审计团队：零时科技安全团队

时间： 2021-05-13

NUT智能合约安全审计报告

1.概述

零时科技安全团队于2021年05月10日，接到 **NUT项目**的安全审计需求，团队与05月13日完成了 **NUT智能合约**安全审计，审计过程中零时科技安全审计专家与NUT项目相关接口人进行沟通，保持信息对称，在操作风险可控的情况下进行安全审计工作，尽量规避在测试过程中对项目产生和运营造成风险。

经过与NUT项目方沟通反馈确认审计过程中发现的漏洞及风险均已修复或在可承受范围内，本次NUT智能合约安全审计结果：通过安全审计。

合约报告MD5：AC838D1E84E7C29007B9051FA99008ED

2.项目背景

2.1 项目简介

> 项目名称：NUT
>
> 合约类型：代币合约，DeFi合约
>
> 代码语言：Solidity
>
> 项目官方Github：https://github.com/nutmoneydefi/launching/tree/main/contracts
>
> 合约文件： NutToken.sol，StakingRewards.sol，UniswapV2Factory.sol，UniswapV2Router02.sol

2.2 审计范围

NUT官方提供的合约文件及对应MD5：

NutToken.sol	A070EE16011C0861377FB3965425B9C3
StakingRewards.sol	71D2BF7C88EAF84131E7ADC908BF3254
UniswapV2Factory.sol	D6CDE918C62CDD7CCC80A60E034B3BC4
UniswapV2Router02.sol	75869E2A1631FA8963C09C56BC39E2BD

2.3 安全审计项

零时科技安全专家对约定内的安全审计项目进行安全审计，本次智能合约安全审计的范围，不包含未来可能出现的新型攻击方式，不包含合约升级活篡改后的代码，不包括在后续跨链部署时的操作过程，不包含项目前端代码安全与项目平台服务器安全。

本次智能合约安全审计项目包括如下：

- 整数溢出
- 重入攻击
- 浮点数和数值精度
- 默认可见性
- Tx.origin身份验证
- 错误的构造函数
- 未验证返回值
- 不安全的随机数
- 时间戳依赖
- 交易顺序依赖
- Delegatecall调用
- Call调用
- 拒绝服务
- 逻辑设计缺陷
- 假充值漏洞
- 短地址攻击
- 未初始化的存储指针
- 代币增发
- 冻结账户绕过
- 权限控制
- Gas使用

3.合约架构分析

3.1 目录结构

```
|
└─NUT
    | NutToken.sol
    | StakingRewards.sol
    | UniswapV2Factory.sol
    | UniswapV2Router02.sol
```

3.2 NUT 合约

Contract

NutToken

- mint(address _to, uint256 _amount)
- delegates(address delegator)
- delegate(address delegatee)
- getCurrentVotes(address account)
- getPriorVotes(address account, uint blockNumber)
- _delegate(address delegator, address delegatee)
- _moveDelegates(address srcRep, address dstRep, uint256 amount)
- safe32(uint n, string memory errorMessage)
- getChainId()

StakingRewards

- totalSupply()
- balanceOf(address account)
- lastTimeRewardApplicable()
- rewardPerToken()
- earned(address account)
- getRewardForDuration()
- stakeWithPermit(uint256 amount, uint deadline, uint8 v, bytes32 r, bytes32 s)
- stake(uint256 amount)
- withdraw(uint256 amount)
- getReward()
- exit()
- notifyRewardAmount(uint256 reward)

UniswapV2ERC20

- _mint(address to, uint value)
- _burn(address from, uint value)
- _approve(address owner, address spender, uint value)
- _transfer(address from, address to, uint value)
- approve(address spender, uint value)
- transfer(address to, uint value)
- transferFrom(address from, address to, uint value)
- permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s)

UniswapV2Pair

- function getReserves()
- function _safeTransfer(address token, address to, uint value)
- function initialize(address _token0, address _token1)
- function _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1)
- function _mintFee(uint112 _reserve0, uint112 _reserve1)
- function mint(address to)
- function burn(address to)
- function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)
- function skim(address to)
- function sync()

UniswapV2Factory

- function allPairsLength()
- function createPair(address tokenA, address tokenB)
- function setFeeTo(address _feeTo)
- function setFeeToSetter(address _feeToSetter)

UniswapV2Router02

- _addLiquidity(address tokenA,address tokenB,uint amountADesired,uint amountBDesired,uint amountAMin,uint amountBMin)
- addLiquidity(address tokenA,address tokenB,uint amountADesired,uint amountBDesired,uint amountAMin,uint amountBMin,address to,uint deadline)
- addLiquidityETH(address token,uint256 amountTokenDesired,uint256 amountTokenMin,uint256 amountETHMin,address to,uint256 deadline)
- removeLiquidity(address tokenA,address tokenB,uint256 liquidity,uint256 amountAMin,uint256 amountBMin,address to,uint256 deadline)

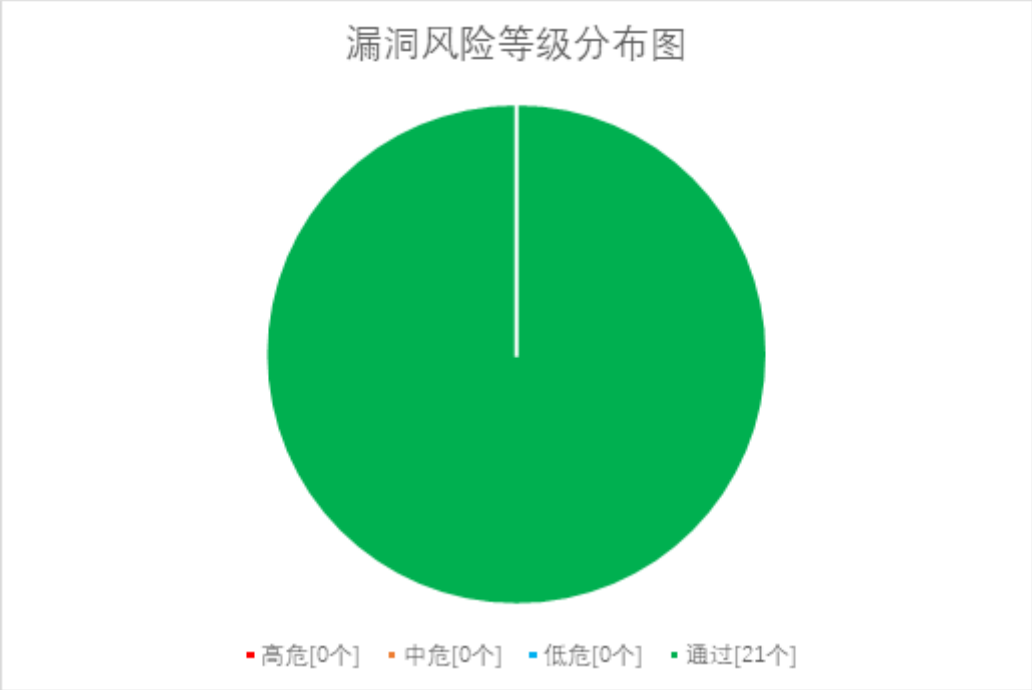
- removeLiquidityETH(address token,uint256 liquidity,uint256 amountTokenMin,uint256 amountETHMin,address to,uint256 deadline)
- removeLiquidityWithPermit(address tokenA,address tokenB,uint256 liquidity,uint256 amountAMin,uint256 amountBMin,address to,uint256 deadline,bool approveMax,uint8 v,bytes32 r,bytes32 s)
- removeLiquidityETHWithPermit(address token,uint256 liquidity,uint256 amountTokenMin,uint256 amountETHMin,address to,uint256 deadline,bool approveMax,uint8 v,bytes32 r,bytes32 s)
- removeLiquidityETHSupportingFeeOnTransferTokens(address token,uint liquidity,uint amountTokenMin,uint amountETHMin,address to,uint deadline)
- removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address token,uint liquidity,uint amountTokenMin,uint amountETHMin,address to,uint deadline,bool approveMax, uint8 v, bytes32 r, bytes32 s)
- _swap(uint[] memory amounts, address[] memory path, address _to)
- swapExactTokensForTokens(uint256 amountIn,uint256 amountOutMin,address[] calldata path,address to,uint256 deadline)
- swapTokensForExactTokens(uint256 amountOut,uint256 amountInMax,address[] calldata path,address to,uint256 deadline)
- swapExactETHForTokens(uint256 amountOutMin,address[] calldata path,address to,uint256 deadline)
- swapTokensForExactETH(uint256 amountOut,uint256 amountInMax,address[] calldata path,address to,uint256 deadline)
- swapExactTokensForETH(uint256 amountIn,uint256 amountOutMin,address[] calldata path,address to,uint256 deadline)
- swapETHForExactTokens(uint256 amountOut,address[] calldata path,address to,uint256 deadline)
- _swapSupportingFeeOnTransferTokens(address[] memory path, address _to)
- swapExactTokensForTokensSupportingFeeOnTransferTokens(uint amountIn,uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapExactETHForTokensSupportingFeeOnTransferTokens(uint amountOutMin,address[] calldata path,address to,uint deadline)
- swapExactTokensForETHSupportingFeeOnTransferTokens(uint amountIn,uint amountOutMin,address[] calldata path,address to,uint deadline)
- quote(uint256 amountA,uint256 reserveA,uint256 reserveB)
- getAmountOut(uint256 amountIn,uint256 reserveIn,uint256 reserveOut)
- getAmountIn(uint256 amountOut,uint256 reserveIn,uint256 reserveOut)
- getAmountsOut(uint256 amountIn, address[] calldata path)
- getAmountsIn(uint256 amountOut, address[] calldata path)

4.审计详情

4.1 漏洞分布

本次安全审计漏洞风险按危险等级分布：

漏洞风险等级分布			
高危	中危	低危	通过
0	0	0	21



本次智能合约安全审计高危漏洞0个，中危0个，低危0个，通过21个，安全等级高。

4.2 漏洞详情

对约定内的智能合约进行安全审计，未发现可以直接利用并产生安全问题的安全漏洞，通过安全审计。

4.3 其他风险

其他风险是指合约安全审计人员认为有风险的代码，在特定情况下可能会影响项目稳定性，但不能构成直接危害的安全问题。

4.3.1 转账数额判断

- **问题点**

StakingRewards合约withdraw()方法主要功能是进行取款操作，该方法中只判断了取款数值是否大于零，未判断该账户和资金总额是否大于取款额度。

```
    function withdraw(uint256 amount) public nonReentrant
updateReward(msg.sender) {
        require(amount > 0, "Cannot withdraw 0");
        _totalSupply = _totalSupply.sub(amount);
        _balances[msg.sender] = _balances[msg.sender].sub(amount);
        stakingToken.safeTransfer(msg.sender, amount);
        emit Withdrawn(msg.sender, amount);
    }
```

- **安全建议**

建议在该方法中添加判断账户资金数额功能。

- **修复状态**

通过与NUT团队沟通，已采用安全建议。

4.3.2 冗余代码

- **问题点**

UniswapV2Factory合约，定义了INIT_CODE_PAIR_HASH变量，但项目合约代码中并未使用该变量。

```
bytes32 public constant INIT_CODE_PAIR_HASH =
keccak256(abi.encodePacked(type(UniswapV2Pair).creationCode));
```

- **安全建议**

建议移除INIT_CODE_PAIR_HASH变量定义。

- **修复状态**

通过与NUT团队沟通，已采用安全建议。

5.安全审计工具

工具名称	功能
Oyente	可以用来检测智能合约中常见bug
securify	可以验证以太坊智能合约的常见类型
MAIAN	可以查找多个智能合约漏洞并进行分类
零时内部工具包	零时(鹰眼系统)自研发工具包+https://audit.noneage.com

6.漏洞风险评估标准

漏洞等级	漏洞风险描述
高危	能直接导致代币合约或者用户数字资产损失的漏洞，比如：整数溢出漏洞、假充值漏洞、重入漏洞、代币违规增发等。 能直接造成代币合约所属权变更或者验证绕过的漏洞，比如：权限验证绕过、call代码注入、变量覆盖、未验证返回值等。 能直接导致代币正常工作的漏洞，比如：拒绝服务漏洞、不安全的随机数等。
中危	需要一定条件才能触发的漏洞，比如代币所有者高权限触发的漏洞，交易顺序依赖漏洞等。不能直接造成资产损失的漏洞，比如函数默认可见性错误漏洞，逻辑设计缺陷漏洞等。
低危	难以触发的漏洞，或者不能导致资产损失的漏洞，比如需要高于攻击收益的代价才能触发的漏洞 无法导致安全漏洞的错误编码问题。

免责声明:

零时科技仅就本报告出具之前发生或存在的事实出具报告并承担相应责任，对于出具报告之后发生的事实由于无法判断智能合约安全状态，因此不对此承担责任。零时科技对该项目约定内的安全审计项进行安全审计，不对该项目背景及其他情况进行负责，项目方后续的链上部署以及运营方式不在本次审计范围。本报告只基于信息提供者截止出具报告时向零时科技提供的信息进行安全审计，对于此项目的信息有隐瞒，或反映的情况与实际情况不符的，零时科技对由此而导致的损失和不利影响不承担任何责任。

市场有风险，投资需谨慎，此报告仅对智能合约代码进行安全审计和结果公示，不作投资建议和依据。

零时科技
NONEAGE

咨询电话：86-17391945345 18511993344
邮　　箱：support@noneage.com
官　　网：www.noneage.com
微　　博：weibo.com/noneage