智能合约安全审计报告

1. 概要

慢雾安全团队于 2020 年 02 月 18 日，收到 Dogeswap 团队对 Dogeswap 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

Dogewap 是一种基于以太坊的协议,旨在促进 ETH 和 ERC20 代币数字资产之间的自动兑换交易,在以太坊/Heco 上自动提供流动性。

审计合约文件:

项目源代码

审计初始版本:

SHA256(Contracts.zip)=

c854afeb9f22ac89cfe35a36f5e8ef5e2510e7b534065a70f2db51e3c361e353

审计最终版本:

SHA256(V1.0Contracts.zip)=

80b1bc8bfe9756a5d18f1b0c2bedf362f11daa2e066f305a812469335d0e68a0

3.2 项目结构

```
├── Factory.sol
└── Router.sol
```

3.3 项目架构

Dogeswap 项目主要分为三个部分，分别为交易对合约、工厂合约和兑换合约。交易对合约基于 ERC2.0 标准开发，主要进行交易流动性管理合约。工厂合约进行交易费率管理、合建交易对合约、迁移配置等。兑换合约根据用户输入信息选择最佳交易路径，进行代币兑换，添加流动性等。

4. 代码概述

4.1 主网合约地址

Factory: https://hecoinfo.com/address/0x0419082bb45f47fe5c530ea489e16478819910f3

Router: https://hecoinfo.com/address/0x539a9fbb81d1d2dc805c698b55c8df81cba6b350

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:

UniswapV2Pair			
Function Name	Visibility	Mutability	Modifiers
getReserves	Public view	-	-
initialize	External	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
transfer	External	Can modify state	-
transferFrom	External	Can modify state	-

approve	External	Can modify state	-
permit	External	Can modify state	-

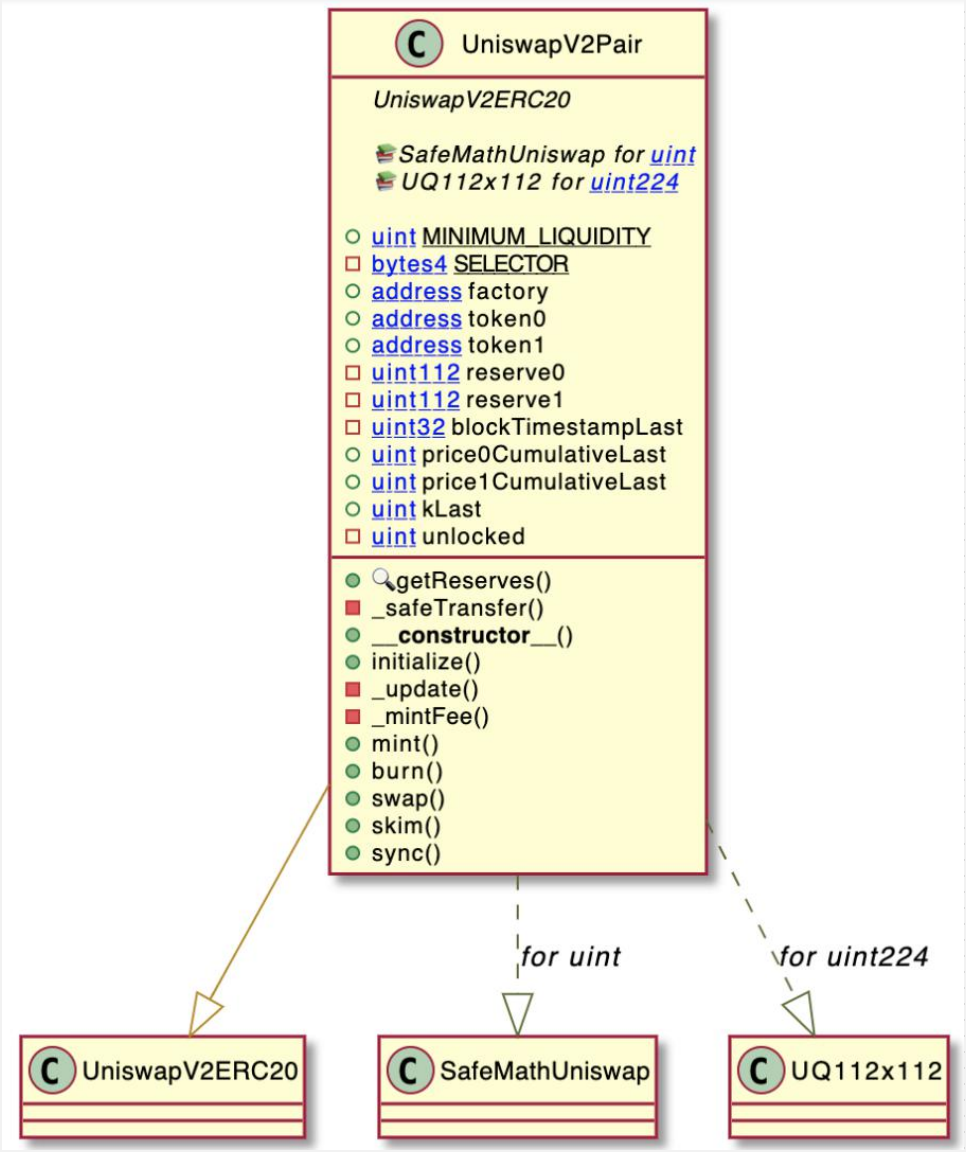
UniswapV2Factory			
Function Name	Visibility	Mutability	Modifiers
allPairsLength	External view	-	-
pairCodeHash	External pure	-	-
createPair	External	Can modify state	-
setFeeTo	External	Can modify state	-
setFeeRate	External	Can modify state	-
setMigrator	External	Can modify state	-
setFeeToSetter	External	Can modify state	-

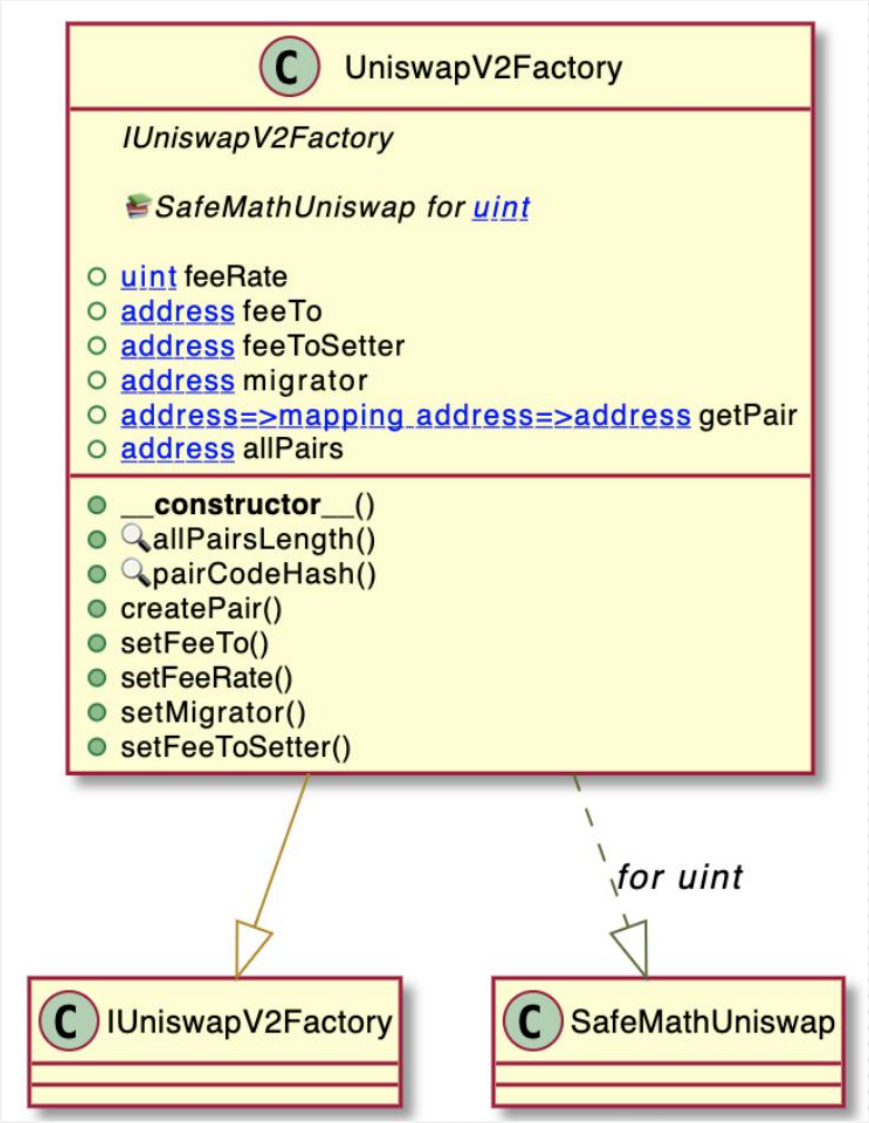
UniswapV2Router02			
Function Name	Visibility	Mutability	Modifiers
addLiquidity	external virtual	Can modify state	ensure
addLiquidityETH	External pure	payable	ensure
removeLiquidity	public virtual	Can modify state	ensure

removeLiquidityETH	public virtual	payable	ensure
removeLiquidityWithPermit	external virtual	Can modify state	-
removeLiquidityETHWithPermit	external virtual	Can modify state	-
removeLiquidityETHSupportingFeeOnTransferTokens	public virtual	Can modify state	ensure
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external virtual	Can modify state	-
swapExactTokensForTokens	external virtual	Can modify state	ensure
swapTokensForExactTokens	external virtual	payable	ensure
swapTokensForExactETH	external virtual	Can modify state	ensure
swapExactTokensForETH	external virtual	Can modify state	ensure
swapETHForExactTokens	external virtual	Can modify state	ensure
swapExactTokensForTokensSupportingFeeOnTransferTokens	external virtual	Can modify state	ensure
swapExactETHForTokensSupportingFeeOnTransferTokens	external virtual	payable	ensure
swapExactTokensForETHSupportingFeeOnTransferTokens	external virtual	Can modify state	ensure
quote	public pure	-	
getAmountOut	public view	-	
getAmountIn	public view	-	
getAmountsOut	public view	-	
getAmountsIn	public view	-	

4.2 主要合约结构分析

Factory.sol




UniswapV2Factory
IUniswapV2Factory
SafeMathUniswap for uint
uint feeRate
address feeTo
address feeToSetter
address migrator
address=>mapping address=>address getPair
address allPairs
__constructor__()
allPairsLength()
pairCodeHash()
createPair()
setFeeTo()
setFeeRate()
setMigrator()
setFeeToSetter()
for uint
IUniswapV2Factory
SafeMathUniswap

Router.sol


UniswapV2Router02
IUniswapV2Router02
SafeMathUniswap for uint
address factory
address WETH
__constructor__()
_addLiquidity()
addLiquidity()
addLiquidityETH()
removeLiquidity()
removeLiquidityETH()
removeLiquidityWithPermit()
removeLiquidityETHWithPermit()
removeLiquidityETHSupportingFeeOnTransferTokens()
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens()
_swap()
swapExactTokensForTokens()
swapTokensForExactTokens()
swapExactETHForTokens()
swapTokensForExactETH()
swapExactTokensForETH()
swapETHForExactTokens()
_swapSupportingFeeOnTransferTokens()
swapExactTokensForTokensSupportingFeeOnTransferTokens()
swapExactETHForTokensSupportingFeeOnTransferTokens()
swapExactTokensForETHSupportingFeeOnTransferTokens()
quote()
getAmountOut()
getAmountIn()
getAmountsOut()
getAmountsIn()
for uint
IUniswapV2Router02
SafeMathUniswap

4.3 代码审计详情

4.3.1 中危漏洞

4.3.1.1 migrator 权限过大

migrator 可以将合约里的流动性迁移到其它合约，从而获取流动性提供者的资产。

代码位置: Factory.sol

```
function mint(address to) external lock returns (uint liquidity) {
(uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
uint balance0 = IERC20Uniswap(token0).balanceOf(address(this));
uint balance1 = IERC20Uniswap(token1).balanceOf(address(this));
uint amount0 = balance0.sub(_reserve0);
uint amount1 = balance1.sub(_reserve1);

bool feeOn = _mintFee(_reserve0, _reserve1);
uint _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in
_mintFee
if (_totalSupply == 0) {
address migrator = IUniswapV2Factory(factory).migrator();
if (msg.sender == migrator) {
liquidity = IMigrator(migrator).desiredLiquidity();
require(liquidity > 0 && liquidity != uint256(-1), "Bad desired liquidity");
} else {
require(migrator == address(0), "Must not have migrator");
liquidity = Math.sqrt(amount0.mul(amount1)).sub(MINIMUM_LIQUIDITY);
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
}
} else {
liquidity = Math.min(amount0.mul(_totalSupply) / _reserve0, amount1.mul(_totalSupply) / _reserve1);
}
require(liquidity > 0, 'UniswapV2: INSUFFICIENT_LIQUIDITY_MINTED');
_mint(to, liquidity);
```

```
_update(balance0, balance1, _reserve0, _reserve1);
if (feeOn) kLast = uint(reserve0).mul(reserve1); // reserve0 and reserve1 are up-to-date
emit Mint(msg.sender, amount0, amount1);
}
```

修复状态: 项目方移除了流动性迁移逻辑。

5. 审计结果

5.1 总结

审计结论 ：通过

审计编号 ：0X002102240001

审计时间 ：2021 年 02 月 24 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 1 个问题，其中包含 1 个中危漏洞。经过与项目方沟通反馈确认审计过程中发现的风险均已修复或在可承受范围内。

6. 声明

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