智能合约安全审计报告

1. 概要

慢雾安全团队于 2021 年 03 月 08 日，收到 BXHash 团队对 BXHash 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

BXHash 是一种基于以太坊的协议，通过提供流动性挖矿激励，在以太坊/Heco 上自动提供流动性。

审计合约文件：

项目源代码

https://github.com/BXHash/contracts

审计初始版本：

commit: 6b7c40c2f3ed83f85004a88e0becb29ac2dbbef7

审计最终版本：

commit: 3d18fa14ae2aac5d1e9b69016074d05d70cc6f02

3.2 项目结构

```
.
├── BXH.sol
├── DelegateERC20.sol
├── airdrop.sol
├── bxhpool.sol
├── interfaces
│   ├── IBXH.sol
│   └── IMasterChefHeco.sol
├── mocks
│   ├── ERC20Mock.sol
│   ├── HUSD.sol
│   └── WHT.sol
├── timelock
│   └── TeamTimeLock.sol
└── uniswapv2
    ├── LICENSE
```

```
├── README.md
├── UniswapV2ERC20.sol
├── UniswapV2Factory.sol
├── UniswapV2Pair.sol
├── UniswapV2Router02.sol
├── interfaces
│   ├── IERC20.sol
│   ├── IUniswapV2Callee.sol
│   ├── IUniswapV2ERC20.sol
│   ├── IUniswapV2Factory.sol
│   ├── IUniswapV2Pair.sol
│   ├── IUniswapV2Router01.sol
│   ├── IUniswapV2Router02.sol
│   └── IWETH.sol
└── libraries
    ├── Math.sol
    ├── SafeMath.sol
    ├── TransferHelper.sol
    ├── UQ112x112.sol
    └── UniswapV2Library.sol
```

3.3 项目架构

BXHash 项目主要分为三个部分，分别为代币合约、兑换合约、挖矿合约、锁仓合约。代币合约基于 ERC2.0 标准开发，用于 BXH 代币管理。兑换合约基于 sushiswap 合约开发，主要用于代币兑换。挖矿合约主要用于流动性锁仓挖取 BXH 代币。锁仓合约主要用于相关代币持仓方解锁代币。

4. 代码概述

4.1 主网合约地址

factory: https://hecoinfo.com/address/0xe0367ec2bd4Ba22B1593E4fEFcB91D29DE6C512a

router: https://hecoinfo.com/address/0x00eFB96dBFE641246E961b472C0C3fC472f6a694

Airdrop: https://hecoinfo.com/address/0xcA1530D5282C703bf3c73c6A08794020dae8b397

Airdrop Pool: https://hecoinfo.com/address/0x0Ef67c16904Af312796560dF80E60581C43C4e24

BXHToken: https://hecoinfo.com/address/0xcBD6Cb9243d8e3381Fea611EF023e17D1B7AeDF0

BXHPool: https://hecoinfo.com/address/0xe3e75Ab09FA0E045523A6B4E81bdB5F11a8bc99c

investTeamLock:

https://hecoinfo.com/address/0xD7B6192601F6e671E42926797a2462a5b6B7b13d

marketTeamLock:

https://hecoinfo.com/address/0xee73ae5C86fd78DbFF1e07a6e9e42D4F1EafDeb0

devTeamLock:

https://hecoinfo.com/address/0x186Dc1ebF9281F98167cfD0A0794B9934587A142

4.2 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:

Airdrop			
Function Name	Visibility	Mutability	Modifiers
newAirdrop	public	Can Modify State	onlyOwner
renounceOwnership	public virtual	Can Modify State	onlyOwner
sweepToken	public	Can Modify State	onlyOwner
transferOwnership	public virtual	Can Modify State	onlyOwner

withdraw	public	Can Modify State	-

BXHToken			
Function Name	Visibility	Mutability	Modifiers
mint	public	Can Modify State	onlyMinter
addMinter	public	Can Modify State	onlyOwner
delMinter	public	Can Modify State	onlyOwner
delegate	Standard ERC20 function		
delegateBySig			
renounceOwnership			
transferOwnership			
transfer			
approve			
transferFrom			
increaseAllowance			
decreaseAllowance			

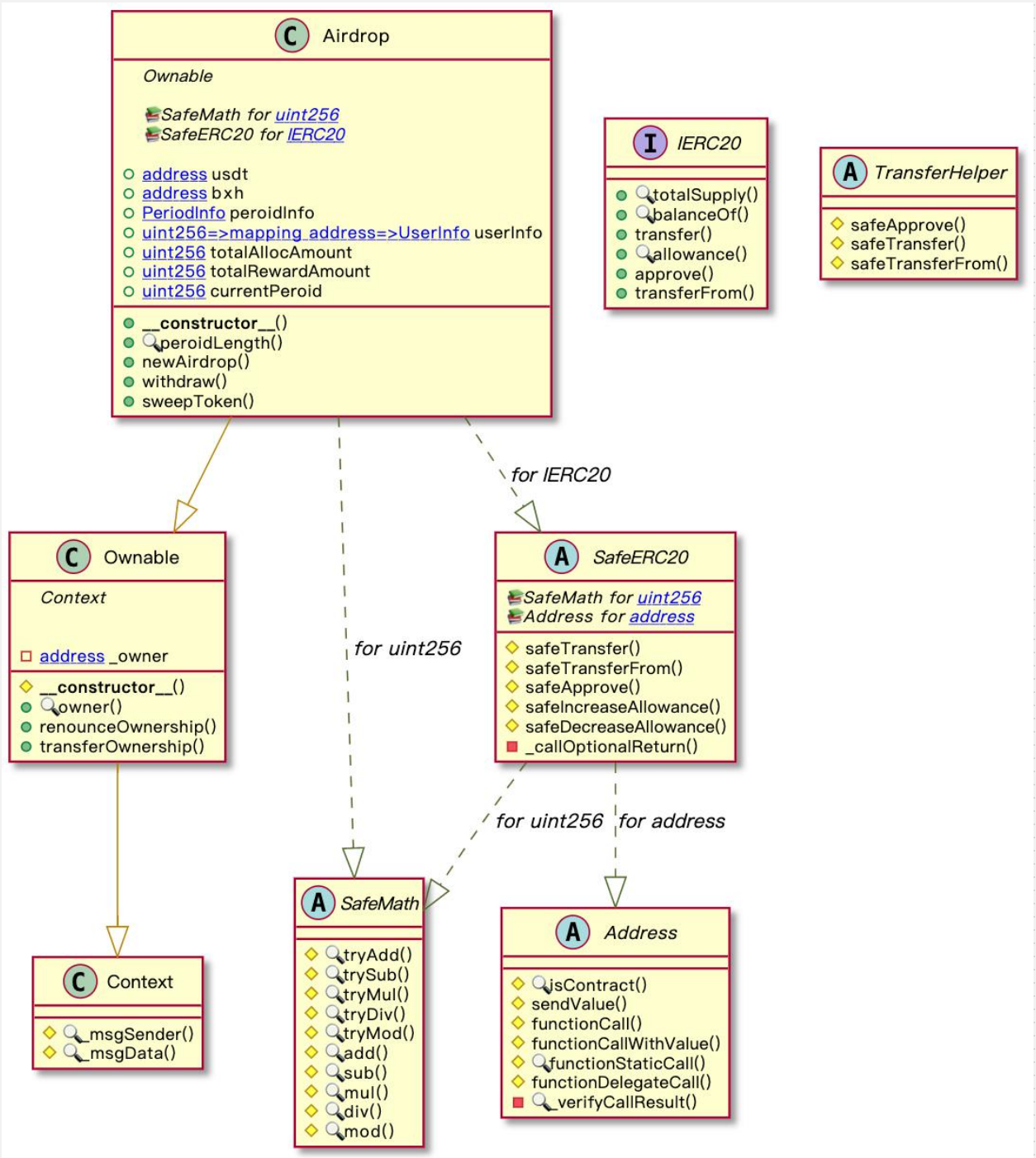
BXHPool			
Function Name	Visibility	Mutability	Modifiers
setDecayPeriod	public	Can Modify State	onlyOwner
setDecayRatio	public	Can Modify State	onlyOwner
setBXHPerBlock	public	Can Modify State	onlyOwner
addMultLP	public	Can Modify State	onlyOwner
setPause	public	Can Modify State	onlyOwner
setMultLP	public	Can Modify State	onlyOwner
replaceMultLP	public	Can Modify State	onlyOwner
add	public	Can Modify State	onlyOwner
set	public	Can Modify State	onlyOwner
setPoolCorr	public	Can Modify State	onlyOwner
batchPrepareRewardTable	public	Can Modify State	-
safeGetBXHBlockReward	public	Can Modify State	-
massUpdatePools	public	Can Modify State	-
updatePool	public	Can Modify State	-
deposit	public	Can Modify State	notPause
withdraw	public	Can Modify State	notPause

emergencyWithdraw	public	Can Modify State	notPause
renounceOwnership	Standard ERC20 function		
transferOwnership			
transfer			
approve			
transferFrom			
increaseAllowance			
decreaseAllowance			

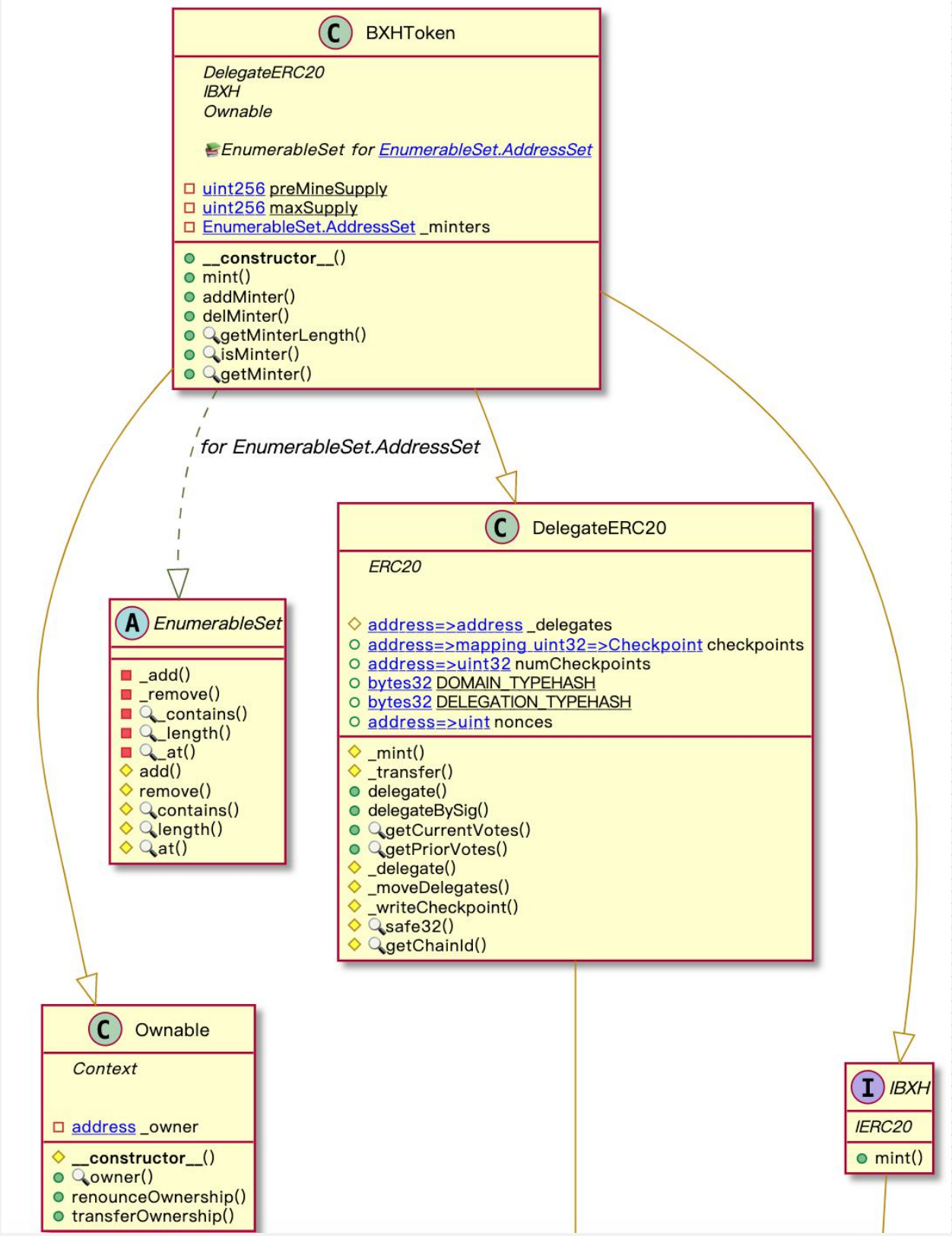
DelegateERC20			
Function Name	Visibility	Mutability	Modifiers
delegate	external	Can Modify State	-
delegateBySig	external	Can Modify State	-
transfer	Standard ERC20 function		
approve			
transferFrom			
increaseAllowance			
decreaseAllowance			

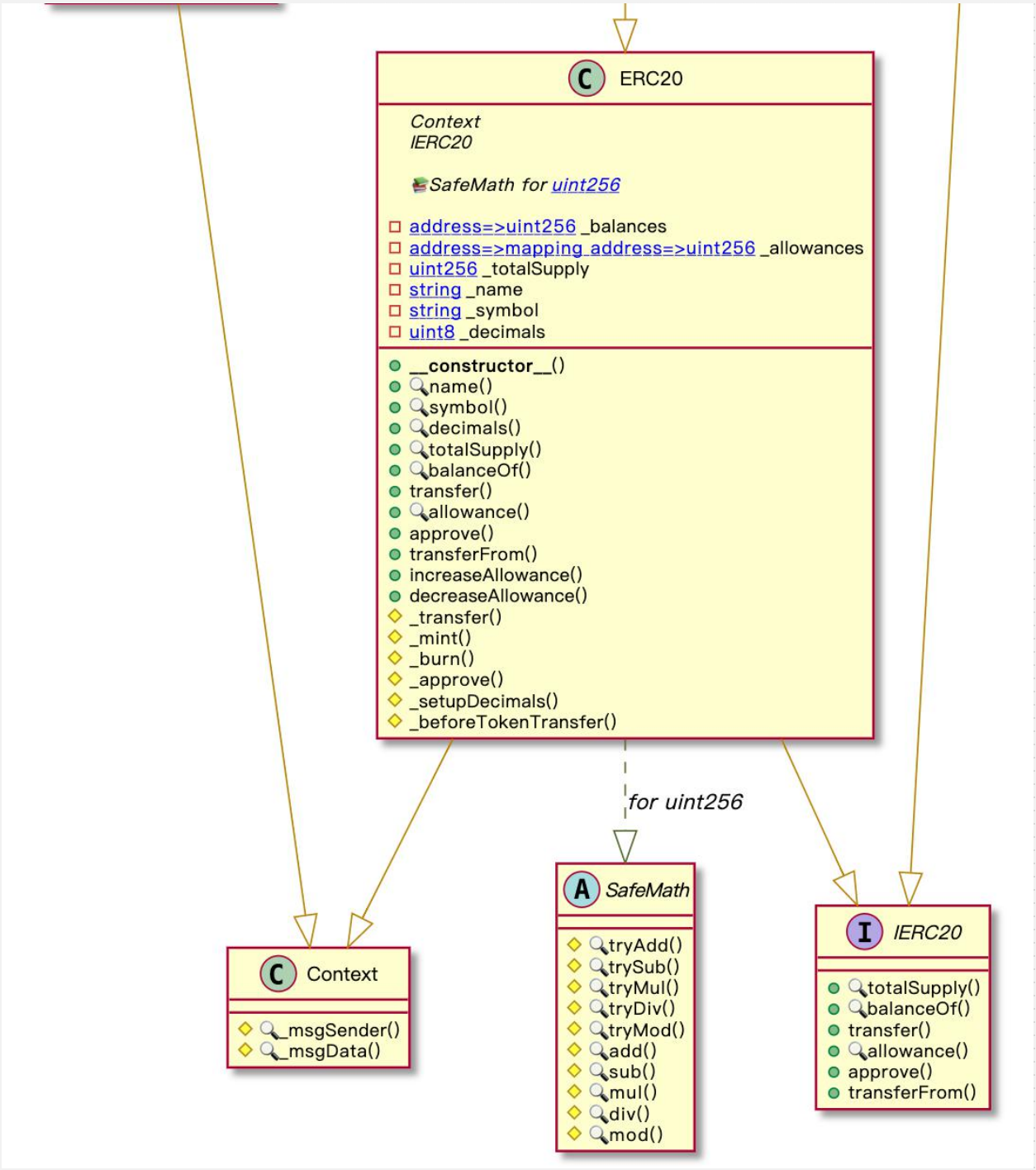
4.3 主要合约结构分析

airdrop.sol

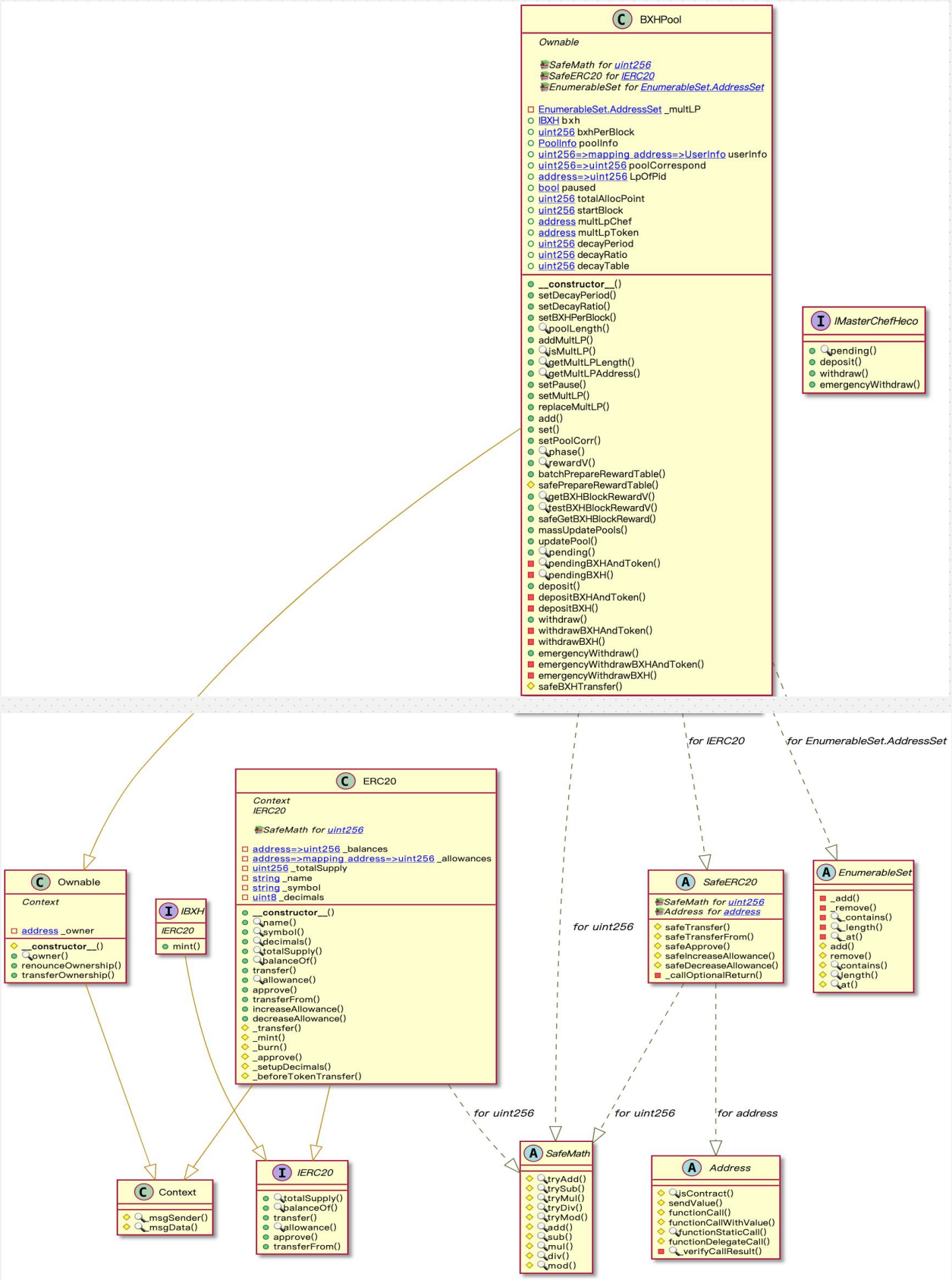


Bxh.sol

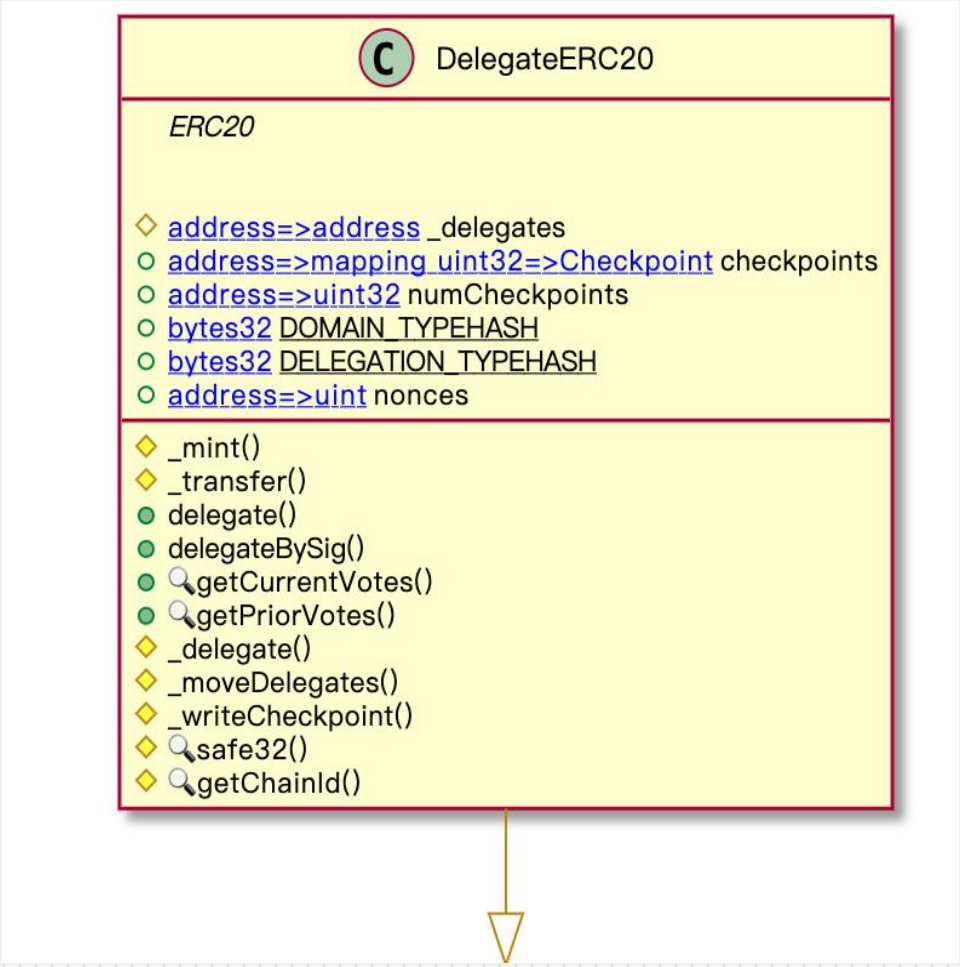

BXHToken
DelegateERC20
IBXH
Ownable
EnumerableSet for EnumerableSet.AddressSet
uint256 preMineSupply
uint256 maxSupply
EnumerableSet.AddressSet _minters
__constructor__()
mint()
addMinter()
delMinter()
getMinterLength()
isMinter()
getMinter()
for EnumerableSet.AddressSet
EnumerableSet
_add()
_remove()
_contains()
_length()
_at()
add()
remove()
contains()
length()
at()
DelegateERC20
ERC20
address=>address _delegates
address=>mapping uint32=>Checkpoint checkpoints
address=>uint32 numCheckpoints
bytes32 DOMAIN_TYPEHASH
bytes32 DELEGATION_TYPEHASH
address=>uint nonces
_mint()
_transfer()
delegate()
delegateBySig()
getCurrentVotes()
getPriorVotes()
_delegate()
_moveDelegates()
_writeCheckpoint()
safe32()
getChainId()
Ownable
Context
address _owner
__constructor__()
owner()
renounceOwnership()
transferOwnership()
IBXH
IERC20
mint()

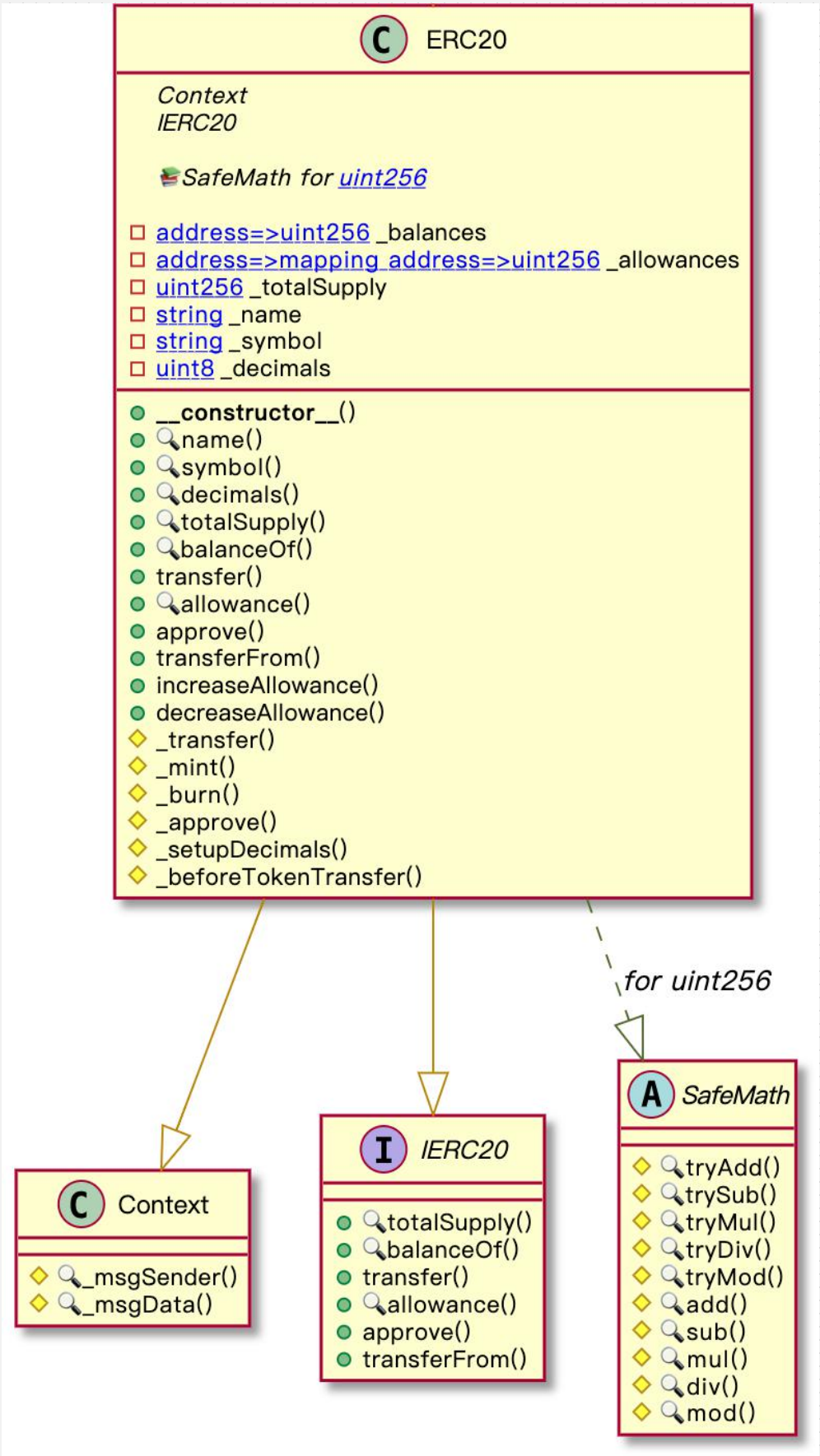

ERC20
Context
IERC20
SafeMath for uint256
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
uint256 _totalSupply
string _name
string _symbol
uint8 _decimals
__constructor__()
name()
symbol()
decimals()
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
_transfer()
_mint()
_burn()
_approve()
_setupDecimals()
_beforeTokenTransfer()
for uint256
Context
_msgSender()
_msgData()
SafeMath
tryAdd()
trySub()
tryMul()
tryDiv()
tryMod()
add()
sub()
mul()
div()
mod()
IERC20
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()

Bxhpool.sol

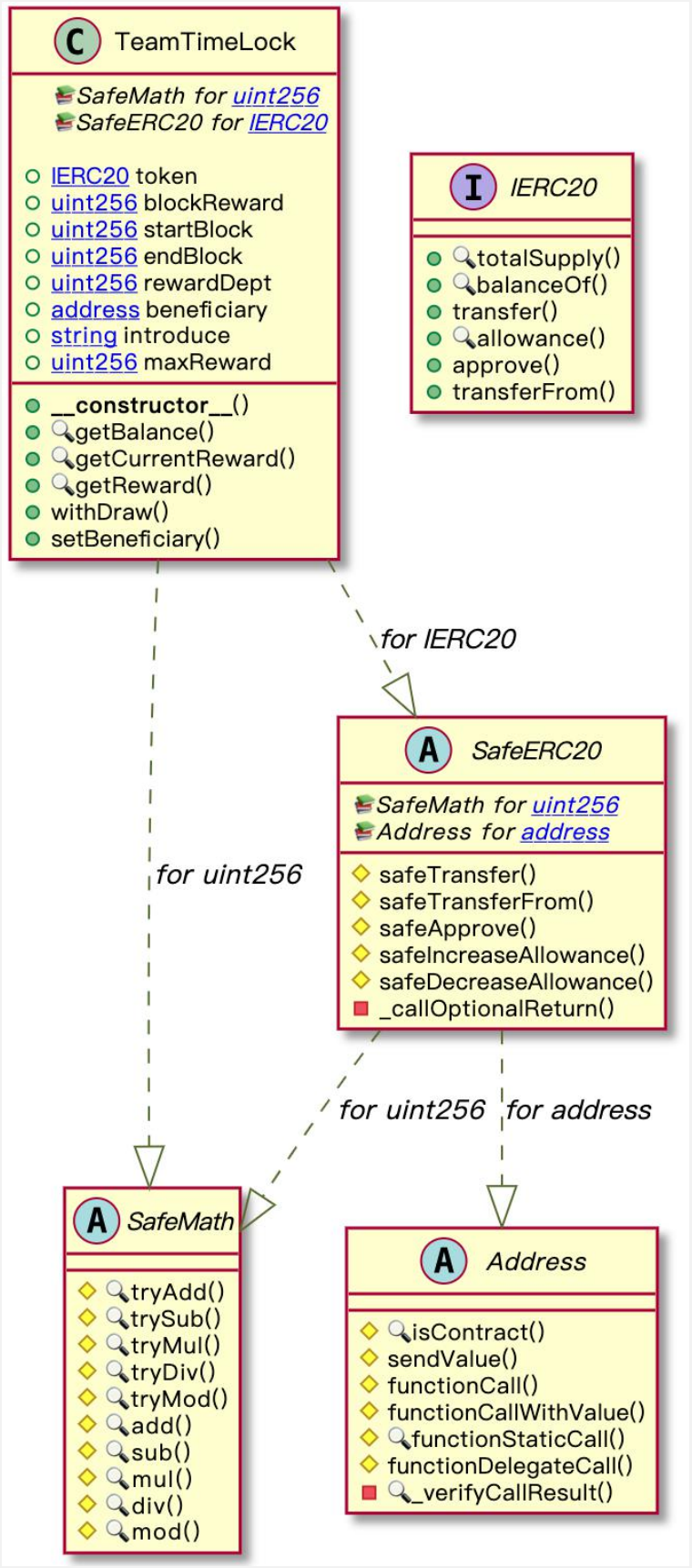


DelegateERC20.sol


DelegateERC20
ERC20
address=>address _delegates
address=>mapping uint32=>Checkpoint checkpoints
address=>uint32 numCheckpoints
bytes32 DOMAIN_TYPEHASH
bytes32 DELEGATION_TYPEHASH
address=>uint nonces
_mint()
_transfer()
delegate()
delegateBySig()
getCurrentVotes()
getPriorVotes()
_delegate()
_moveDelegates()
_writeCheckpoint()
safe32()
getChainId()


ERC20
Context
IERC20
SafeMath for uint256
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
uint256 _totalSupply
string _name
string _symbol
uint8 _decimals
__constructor__()
name()
symbol()
decimals()
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
_transfer()
_mint()
_burn()
_approve()
_setupDecimals()
_beforeTokenTransfer()
for uint256
Context
_msgSender()
_msgData()
IERC20
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
SafeMath
tryAdd()
trySub()
tryMul()
tryDiv()
tryMod()
add()
sub()
mul()
div()
mod()

TeamTimeLock.sol



4.4 代码审计详情

4.4.1 中危漏洞

4.4.1.1 管理员权限过大

管理员（owner）权限可以直接对多个合约参数进行修改，进而影响用户资金安全。

例如 factory 合约 feeToSetter 权限可以设置 migrator 账号，而 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: UniswapV2Pair.sol

```
// this low-level function should be called from a contract which performs important safety checks
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: 项目方已将 migrator 设置为 0 地址（销毁权限），后续将通过 DAO 社区自治来转移 owner 权限。

4.4.2 低危漏洞

4.4.2.1 Airdrop owner 权限过大

sweepToken 函数 owner 可以转移合约代币余额。

代码位置: airdrop.sol

```
function sweepToken(address token,address to) public onlyOwner { //SlowMist// 低危：权限过大，owner 可以转移合约代币余额
uint256 bal = IERC20(token).balanceOf(address(this));
require(bal>0,"not enough balance");
TransferHelper.safeTransfer(token,to, bal);
}
```

修复状态: 项目方移除了 sweepToken 函数。

4.4.2.2 未限制 totalAllocPoint 允许的最大值

add/set 函数未限制 totalAllocPoint 允许的最大值

代码位置: bxhpool.sol

```
// Add a new lp to the pool. Can only be called by the owner.
```

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

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

修复状态: 项目方反馈这部分只有管理员根据社区投票结果来严格调用，操作上不会出现偏差。

4.4.3 增强建议

4.4.3.1 空投领取 BXH 门槛过低

每个钱包地址可以用私募价格领取一定数量的 BXH，用户可通过使用大量地址领取大量份额的代币，造成持

仓集中化。

修复状态: 增加 requestWithdraw()用于登记空投，项目方在链下评估空投数量。原空投函数 withdraw 限制为只能管理员调用。

4.4.3.2 Gas 消耗过大问题

使用 if-else 逻辑进行检查，这将导致执行过程中不会直接抛出错误，与 require 检查相比，这也将消耗更多的 gas。

代码位置: BXH.sol

```
function mint(address _to, uint256 _amount) public onlyMinter override returns (bool) {
if (_amount.add(totalSupply()) > maxSupply) {
return false;
}
_mint(_to, _amount);
return true;
}
```

修复状态: 项目方已将 if-else 逻辑检查修改为 require。

5. 审计结果

5.1 总结

审计结论 ：管理员权限过大

审计编号 ：0X002103150004

审计时间 ：2021 年 03 月 15 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 5 个问题，其中包含 1 个中危漏洞、2 个低危漏洞，并提出了 2 点增强建议。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

慢雾仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，慢雾无法判断其智能合约安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料(简称“已提供资料”)。慢雾假设：已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任。

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号