智能合约安全审计报告

目录

1 前言

慢雾安全团队于 2020 年 11 月 25 日，收到 WePiggy 团队对 WePiggy 第二期代码进行安全审计的申请，根据项目特点慢雾安全团队制定如下审计方案。

慢雾安全团队将采用“白盒为主，黑灰为辅”的策略，以最贴近真实攻击的方式，对项目进行安全审计。

慢雾科技 DeFi 项目测试方法：

黑盒测试	站在外部从攻击者角度进行安全测试。
灰盒测试	通过脚本工具对代码模块进行安全测试，观察内部运行状态，挖掘弱点。
白盒测试	基于项目的源代码，进行脆弱性分析和漏洞挖掘。

慢雾科技 DeFi 漏洞风险等级：

严重漏洞	严重漏洞会对项目的安全造成重大影响，强烈建议修复严重漏洞。
高危漏洞	高危漏洞会影响项目的正常运行，强烈建议修复高危漏洞。
中危漏洞	中危漏洞会影响项目的运行，建议修复中危漏洞。
低危漏洞	低危漏洞可能在特定场景中会影响项目的业务操作，建议项目方自行评估和考虑这些问题是否需要修复。
弱点	理论上存在安全隐患，但工程上极难复现。
增强建议	编码或架构存在更好的实践方法。

2. 审计方法

慢雾安全团队智能合约安全审计流程包含两个步骤:

- 使用开源或内部自动化分析的工具对合约代码中常见的安全漏洞进行扫描和测试。
- 人工审计代码的安全问题，通过人工分析合约代码，发现代码中潜在的安全问题。

如下是合约代码审计过程中慢雾安全团队会重点审查的漏洞列表:

（其他未知安全漏洞不包含在本次审计责任范围）

- 重入攻击
- 重放攻击
- 重排攻击
- 短地址攻击
- 拒绝服务攻击
- 交易顺序依赖
- 条件竞争攻击
- 权限控制攻击
- 整数上溢/下溢攻击
- 时间戳依赖攻击
- Gas 使用，Gas 限制和循环
- 冗余的回调函数
- 不安全的接口使用
- 函数状态变量的显式可见性
- 逻辑缺陷
- 未声明的存储指针
- 算术精度误差
- tx.origin 身份验证
- 假充值漏洞
- 变量覆盖

3. 项目背景

3.1 项目介绍

WePiggy 是一个开源，非托管的加密资产借贷市场协议。在 WePiggy 的市场上，用户可存入特定的加密资产赚取利息，也可以支付一定的利息借取某种加密资产。

项目官网地址:

https://wepiggy.com

审计版本代码:

https://github.com/WePiggy/wepiggy-contracts/tree/826534cee92b01665a6f7b8258e33b5e612

dee59

修复版本代码:

https://github.com/WePiggy/wepiggy-contracts/tree/844a23c63ce6e9347bc6cfafd6fb1c40de172672

4. 代码概述

4.1 合约可见性分析

在审计过程中，慢雾安全团队对核心合约的可见性进行分析，结果如下:

PiggyBreeder			
Function Name	Visibility	Mutability	Modifiers
poolLength	External	-	-
usersLength	External	-	-
setDevAddr	Public	Can modify state	-
setMigrator	Public	Can modify state	onlyOwner
setEnableClaimBlock	Public	Can modify state	onlyOwner
setReduceIntervalBlock	Public	Can modify state	onlyOwner
setAllocPoint	Public	Can modify state	onlyOwner
setReduceRate	Public	Can modify state	onlyOwner
setDevMiningRate	Public	Can modify state	onlyOwner
replaceMigrate	Public	Can modify state	onlyOwner
migrate	Public	Can modify state	onlyOwner
safePiggyTransfer	Internal	Can modify state	-
getPiggyPerBlock	Public	-	-
getMultiplier	Public	-	-
allPendingPiggy	External	-	-
pendingPiggy	External	-	-
_pending	Internal	-	-
massUpdatePools	Public	Can modify state	-
updatePool	Public	Can modify state	-
add	Public	Can modify state	onlyOwner
stake	Public	Can modify state	-
unStake	Public	Can modify state	-
claim	Public	Can modify state	-

emergencyWithdraw	Public	Can modify state	-

FundingHolder			
Function Name	Visibility	Mutability	Modifiers
transfer	Public	Can modify state	onlyOwner

FundingManager			
Function Name	Visibility	Mutability	Modifiers
safePiggyTransfer	Internal	Can modify state	-
addFunding	Public	Can modify state	onlyOwner
setFunding	Public	Can modify state	onlyOwner
getPendingBalance	Public	-	-
claim	Public	Can modify state	-

WePiggyToken			
Function Name	Visibility	Mutability	Modifiers
mint	Public	Can modify state	-
_transfer	Internal	Can modify state	-
delegates	External	-	-
delegate	External	Can modify state	-
delegateBySig	External	Can modify state	-
getCurrentVotes	External	-	-
getPriorVotes	External	-	-
_delegate	Internal	Can modify state	-
_moveDelegates	Internal	Can modify state	-
_writeCheckpoint	Internal	Can modify state	-
safe32	Internal	-	-
getChainId	Internal	-	-

Timelock			
Function Name	Visibility	Mutability	Modifiers
setDelay	Public	Can modify state	-
acceptAdmin	Public	Can modify state	-
setPendingAdmin	Public	Can modify state	-
queueTransaction	Public	Can modify state	-
cancelTransaction	Public	Can modify state	-

executeTransaction	Public	Payable	-
getBlockTimestamp	Internal	-	-
receive()	External	Payable	-

AToken2PTokenMigrator			
Function Name	Visibility	Mutability	Modifiers
migrate	Public	Can modify state	-
_getTokenBalance	Internal	Can modify state	-
Receive	External	Payable	-
compareStrings	Internal	-	-

ATokenMigrator			
Function Name	Visibility	Mutability	Modifiers
replaceMigrate	External	Payable	-
migrate	External	Payable	-
receive	External	Payable	-

CErc20Migrator			
Function Name	Visibility	Mutability	Modifiers
replaceMigrate	External	Can modify state	-
migrate	External	Can modify state	-

CEthMigrator			
Function Name	Visibility	Mutability	Modifiers
replaceMigrate	External	Payable	-
migrate	External	Payable	-
receive	External	Payable	-

Comptroller			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
enterMarkets	Public	Can modify state	-
addToMarketInternal	Internal	Can modify state	-
exitMarket	External	Can modify state	-

getAssetsIn	External	-	-
checkMembership	External	-	-
mintAllowed	External	Can modify state	-
mintVerify	External	Can modify state	-
redeemAllowed	External	Can modify state	-
redeemAllowedInternal	Internal	-	-
redeemVerify	External	Can modify state	-
borrowAllowed	External	Can modify state	-
borrowVerify	External	Can modify state	-
repayBorrowAllowed	External	Can modify state	-
repayBorrowVerify	External	Can modify state	-
liquidateBorrowAllowed	External	Can modify state	-
liquidateBorrowVerify	External	Can modify state	-
seizeAllowed	External	Can modify state	-
seizeVerify	External	Can modify state	-
transferAllowed	External	Can modify state	-
transferVerify	External	Can modify state	-
getAccountLiquidity	Public	-	-
getAccountLiquidityInternal	Internal	-	-
getHypotheticalAccountLiquidity	Public	-	-
getHypotheticalAccountLiquidityInternal	Internal	-	-
liquidateCalculateSeizeTokens	External	-	-
_setPriceOracle	Public	Can modify state	onlyOwner
_setCloseFactor	External	Can modify state	onlyOwner
_setCollateralFactor	External	Can modify state	onlyOwner
_setMaxAssets	External	Can modify state	onlyOwner
_setLiquidationIncentive	External	Can modify state	onlyOwner
_supportMarket	External	Can modify state	onlyOwner
_addMarketInternal	Internal	Can modify state	onlyOwner
_setMarketBorrowCaps	External	Can modify state	-
_setBorrowCapGuardian	External	Can modify state	onlyOwner
_setPauseGuardian	Public	Can modify state	onlyOwner
_setMintPaused	Public	Can modify state	-
_setBorrowPaused	Public	Can modify state	-
_setTransferPaused	Public	Can modify state	-
_setSeizePaused	Public	Can modify state	-
_setDistributeWpcPaused	Public	Can modify state	-
_setPiggyDistribution	Public	Can modify state	onlyOwner
getAllMarkets	Public	-	-

isMarketMinted	Public	-	-
isMarketListed	Public	-	-
_setMarketMinted	Public	Can modify state	-

SimplePriceOracle			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
getUnderlyingPrice	Public	-	-
setUnderlyingPrice	Public	Can modify state	onlyOwner
setPrice	Public	Can modify state	onlyOwner
getPrice	External	-	-
get	External	-	-
compareStrings	Internal	-	-

WePiggyPriceOracleV1			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
getPrice	External	-	-
setPrice	External	Can modify state	onlyOwner
setTokenConfig	Public	Can modify state	onlyOwner

WePiggyPriceProviderV1			
Function Name	Visibility	Mutability	Modifiers
getUnderlyingPrice	External	-	-
_getUnderlyingPriceInternal	Internal	-	-
_getCustomerPriceInternal	Internal	-	-
_getCompoundPriceInternal	Internal	-	-
_getChainlinkPriceInternal	Internal	-	-
addTokenConfig	Public	Can modify state	onlyOwner
addOrUpdateTokenConfigSource	Public	Can modify state	onlyOwner
updateTokenConfigBaseUnit	Public	Can modify state	onlyOwner
updateTokenConfigFixedUsd	Public	Can modify state	onlyOwner
getOracleSourcePrice	Public	-	-
compareStrings	Internal	-	-
oracleLength	Public	-	-

PiggyDistribution			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
distributeMintWpc	Public	Can modify state	-
distributeRedeemWpc	Public	Can modify state	-
distributeBorrowWpc	Public	Can modify state	-
distributeRepayBorrowWpc	Public	Can modify state	-
distributeSeizeWpc	Public	Can modify state	-
distributeTransferWpc	Public	Can modify state	-
_stakeTokenToPiggyBreeder	Public	Can modify state	onlyOwner
_claimWpcFromPiggyBreeder	Public	Can modify state	onlyOwner
_refreshWpcSpeeds	Public	Can modify state	onlyOwner
refreshWpcSpeedsInternal	Internal	Can modify state	-
updateWpcSupplyIndex	Internal	Can modify state	-
updateWpcBorrowIndex	Internal	Can modify state	-
distributeSupplierWpc	Internal	Can modify state	-
distributeBorrowerWpc	Internal	Can modify state	-
transferWpc	Internal	Can modify state	-
claimWpc	Public	Can modify state	-
claimWpc	Public	Can modify state	-
claimWpc	Public	Can modify state	-
_setWpcRate	Public	Can modify state	onlyOwner
_addWpcMarkets	Public	Can modify state	onlyOwner
_addWpcMarketInternal	Internal	Can modify state	-
_dropWpcMarket	Public	Can modify state	onlyOwner
transferALLWPC	Public	Can modify state	onlyOwner

BaseJumpRateModel			
Function Name	Visibility	Mutability	Modifiers
updateJumpRateModel	External	Can modify state	-
utilizationRate	Public	-	-
getBorrowRateInternal	Internal	-	-
getBorrowRate	External	-	-
getSupplyRateInternal	Internal	-	-
getSupplyRate	External	-	-
updateJumpRateModelInternal	Internal	Can modify state	onlyOwner

DAIInterestRateModel			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
updateDAIJumpRateModel	External	Can modify state	-
getSupplyRate	External	-	-
dsrPerBlock	Public	-	-
poke	Public	Can modify state	-

JumpRateModel			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer

PERC20			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
mint	External	Can modify state	-
mintForMigrate	External	Can modify state	-
redeem	External	Can modify state	-
redeemUnderlying	External	Can modify state	-
borrow	External	Can modify state	-
repayBorrow	External	Can modify state	-
repayBorrowBehalf	External	Can modify state	-
liquidateBorrow	External	Can modify state	-
_addReserves	External	Can modify state	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can modify state	-
doTransferOut	Internal	Can modify state	-

PEther			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can modify state	initializer
mint	External	Payable	-
mintForMigrate	External	Payable	-
redeem	External	Can modify state	-
redeemUnderlying	External	Can modify state	-

borrow	External	Can modify state	-
repayBorrow	External	Payable	-
repayBorrowBehalf	External	Payable	-
liquidateBorrow	External	Payable	-
	External	Payable	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can modify state	-
doTransferOut	Internal	Can modify state	-
require-Error	Internal	-	-

PToken			
Function Name	Visibility	Mutability	Modifiers
init	Public	Can modify state	onlyOwner
transferTokens	Internal	Can modify state	-
transfer	External	Can modify state	nonReentrant
transferFrom	External	Can modify state	nonReentrant
approve	External	Can modify state	-
allowance	External	-	-
balanceOf	External	-	-
balanceOfUnderlying	External	Can modify state	-
getAccountSnapshot	External	-	-
getBlockNumber	Internal	-	-
borrowRatePerBlock	External	-	-
supplyRatePerBlock	External	-	-
totalBorrowsCurrent	External	Can modify state	nonReentrant
borrowBalanceCurrent	External	Can modify state	nonReentrant
borrowBalanceStored	Public	-	-
borrowBalanceStoredInternal	Internal	-	-
borrowInterestBalancePriorInternal	Internal	-	-
exchangeRateCurrent	Public	-	-
exchangeRateStored	Public	-	-
exchangeRateStoredInternal	Internal	-	-
getCash	External	-	-
accrueInterestSnapshot	Public	-	-
accrueInterest	Public	Can modify state	-
mintInternal	Internal	Can modify state	nonReentrant
mintInternalForMigrate	Internal	Can modify state	nonReentrant
mintFresh	Internal	Can modify state	-

redeemInternal	Internal	Can modify state	nonReentrant
redeemUnderlyingInternal	Internal	Can modify state	nonReentrant
redeemFresh	Internal	Can modify state	-
borrowInternal	Internal	Can modify state	nonReentrant
borrowFresh	Internal	Can modify state	-
repayBorrowInternal	Internal	Can modify state	nonReentrant
repayBorrowBehalfInternal	Internal	Can modify state	nonReentrant
repayBorrowFresh	Internal	Can modify state	-
liquidateBorrowInternal	Internal	Can modify state	nonReentrant
liquidateBorrowFresh	Internal	Can modify state	-
seize	External	Can modify state	nonReentrant
seizeInternal	Internal	Can modify state	-
_setComptroller	Public	Can modify state	onlyOwner
_setReserveFactor	External	Can modify state	nonReentrant
_setReserveFactorFresh	Internal	Can modify state	onlyOwner
_addReservesInternal	Internal	Can modify state	nonReentrant
_addReservesFresh	Internal	Can modify state	-
_reduceReserves	External	Can modify state	nonReentrant
_reduceReservesFresh	Internal	Can modify state	onlyOwner
_setInterestRateModel	Public	Can modify state	-
_setInterestRateModelFresh	Internal	Can modify state	onlyOwner
_setMigrator	Public	Can modify state	onlyOwner
_setMinInterestAccumulated	Public	Can modify state	onlyOwner
getCashPrior	Internal	-	-
doTransferIn	Internal	Can modify state	-
doTransferOut	Internal	Can modify state	-

4.2 合约信息

第一期已经审计了如下的合约，且已经部署到主网。

合约主网地址：

Contract Name	Contract Address
WePiggyToken	0x4C78015679FabE22F6e02Ce8102AFbF7d93794eA
FundingHolder[InsurancePayment]	0xb205d0AeF84C666FBBe441C61DC04fEb844444E6
FundingHolder[ResourceExpansion]	0x7a603D06007fc09f896Fb75644365AB091A7b91a
FundingHolder[TeamVote]	0x8C14A40c488E16b055bc4e250563B5480047f01E

FundingHolder[TeamSpending]	0x209EE6f924a39BdDC9A57c0E263dd5E29CEAc78A
FundingHolder[CommunityRewards]	0x1fe0F15546858d9A1E84CE2E7908b160608267c5
FundingManager	0xf18D727C034f47AE2C0FE221C1cf4A15f0557b5F
PiggyBreeder	0x451032C55F813338b6e73c1c4B24217614165454
TimeLock	0x311aEA58Ca127B955890647413846E351df32554

第二期审计的合约部署到主网，且根据沟通后续主网的部署也会将合约的 Owner 权限转移给 timelock 合约，目前的 Owner 是一个多签合约，PToken 部署采用的了可升级的模型，AdminUpgradeabilityProxy 合约的 admin 由项目方的普通地址进行管理。

合约主网地址：

Contract Name	Contract Address
AWT1	0x12D803497D1e58dD4D4A4F455D754f1d0F937C8b
WPC_AIRDROP_FUNDING_MANAGER	0x2dd8FFA7923a17739F70C34759Af7650e44EA3BE
WP_PRICE_PROVIDER_V1	0xe212829Ca055eD63279753971672c693C6C6d088
WE_PIGGY_PRICE_ORACLE_V1	0xE4A1E73157EB4b58b1347E2BE2df7ac83467b288
COMPTROLLER	0x0C8c1ab017c3C0c8A48dD9F1DB2F59022D190f0b
STABLECOIN_JUMP_RATE_MODEL	0x8158B34fF8A36dD9E4519d62C52913C24ad5554b
BTC_ETH_JUMP_RATE_MODEL	0xA0a75821220bfC74f8012d5D5745FE472F510075
P_DAI	0x85166b72c87697a6acfF24101B43Fd54fE28a179
P_USDT	0x5cFad792C4Df1323188180778AeC58E00eAcE32a
P_USDC	0xf8E5b9738BF63ADFFf36a849F9b9C9617c8D8c1f
P_WBTC	0xc12B9D620bFCB48be3e0CCbf0ea80C717333b46F
P_ETH	0x27A94869341838D5783368a8503FdA5fbCd7987c
MAX_IMILLION	0x97F3763F8C0bE87Cab0e99Ee4b7806acA772FeDA
C_ETH_MIGRATOR	0xe70ADe95E3D038398eb0ACD17534200b0c87A7c4
C_DAI_MIGRATOR	0x836490DcF3C63da80A8BEE9C666Fbfe294D80c69
C_USDT_MIGRATOR	0x8ac4cdf74ce4B4691591384122f3eFeAae351C25
C_USDC_MIGRATOR	0x271e498c2FCaa9f2dbB4f6b9046ba65Fa2c79dBE
C_WBTC_MIGRATOR	0xab80D132eDd440DD7E80ADF77BE91a7445921334
A_TOKEN_MIGRATOR	0x894c9FB45aeCc4F757C2d07A692504E8e29A5747
PIGGY_DISTRIBUTION	0xfe584fc2b820aef5e4976beb9379ecd22aed0c6c

4.3 代码审计

4.3.1 高危漏洞

4.3.1.1 权限校验缺失

distributeTransferWpc 函数没有进行权限控制，允许任意用户自行分发奖励，建议对 distributeTransferWpc 添加权限控制。

- contracts/farm/PiggyDistribution.sol

```
function distributeTransferWpc(address pToken, address src, address dst, bool distributeAll) public override(IPiggyDistribution) {
        updateWpcSupplyIndex(pToken);
        distributeSupplierWpc(pToken, src, distributeAll);
        distributeSupplierWpc(pToken, dst, distributeAll);
    }
```

修复状态: 该问题在 commit: 192d06294bb48b2c8fc555b3dff010d4136d197e 中已修复。

4.3.1.2 条件竞争问题

setTargetToken 函数可以设置迁移的 targetToken 合约地址，存在竞态问题，可以在正常调用迁移函数的时候，攻击者使用更高的 gas price 来将 targetToken 设置为一个恶意合约，等待迁移完成后就可以将底层资产从恶意合约中取走。

- contracts/farm/migrator/CErc20Migrator.sol

```
function setTargetToken(address _targetToken) public {
        targetToken = _targetToken;
    }
```

修复状态: 这个问题在 commit: ee167fa88735065350b9880a5402ede923ae7b8d 中已修复。

4.3.2 中危漏洞

4.3.2.1 业务逻辑缺陷

ATokenMigrator 合约通过 receive 接收以太币，如果有 eth 打进 ATokenMigrator 合约中但是还没有调用 replaceMigrate 函数，这样的话在调用 replaceMigrate 函数的时候迁移的是底层资产和错误(恶意)打入合约的 eth，但是最终调用 mintForMigrate 的时候使用的 lp 是通过 oldLpToken.balanceOf(breeder);获取的，没有进行重新的计算，这样会使用超出预期的 eth 进行迁移，但是 lp 还是使用旧的数值，会导致最终迁移的数据和预期的不一致，同理 token 也是如此，建议对合约中的 eth 和 token 资产进行检查，避免迁移的时候因为合约中的存在资产导致迁移后数值不一致的情况。

- contracts/farm/migrator/ATokenMigrator.sol

```
function replaceMigrate(ATokenInterface oldLpToken) external payable returns (PToken, uint){
        require(msg.sender == breeder, "not from breeder");
        require(block.number >= notBeforeBlock, "too early to migrate");
        address self = address(this);
        uint256 lp = oldLpToken.balanceOf(breeder);
        require(lp > 0, "balance must bigger than 0");
        oldLpToken.transferFrom(breeder, self, lp);
        oldLpToken.redeem(lp);
        address underlyingAssetAddress = oldLpToken.underlyingAssetAddress();
        if (underlyingAssetAddress == address(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE)) {
            PEther newLpToken = PEther(targetToken);
            // 获得赎回了多少代币
            uint redeemBal = self.balance;
            // 将赎回的代币，抵押到 wePiggy 中，生成 pToken
            newLpToken.mintForMigrate{value : redeemBal}(lp);
            // 获得抵押生成的 pToken 有多少
            uint mintBal = newLpToken.balanceOf(self);
            //将余额转到挖矿合约
            newLpToken.transferFrom(self, breeder, mintBal);
            //返回占比
            return (newLpToken, mintBal);
        } else {
            PERC20 newLpToken = PERC20(targetToken);
            require(underlyingAssetAddress == newLpToken.underlying(), "not match");
```

```
            //获得赎回了多少代币
            uint redeemBal = 0;
            IERC20 token = IERC20(underlyingAssetAddress);
            redeemBal = token.balanceOf(self);
            // 将赎回的代币，抵押到 wePiggy 中，生成 pToken
            token.approve(address(newLpToken), redeemBal);
            newLpToken.mintForMigrate(redeemBal, lp);
            // 获得抵押生成的 pToken 有多少
            uint mintBal = newLpToken.balanceOf(self);
            //将余额转到挖矿合约
            newLpToken.transferFrom(self, breeder, mintBal);
            return (newLpToken, mintBal);
        }
    }
```

修复状态: 这个问题在 commit: ee167fa88735065350b9880a5402ede923ae7b8d 中进行了修复。

4.3.2.2 算术精度问题

先进行除法再进行减法最终进行乘法，这里会因为 solidity 的特性不会保留小数，导致的算术精度缺失的情况，

建议先用乘法再用除法，避免精度缺失的情况。

- contracts/token/PToken.sol

```
if (borrowSnapshot.interestIndex == 0) {
            return (MathError.NO_ERROR, 0);
        }
(mathErr, interestTimesIndex) = divUInt(borrowIndex, borrowSnapshot.interestIndex);
if (mathErr != MathError.NO_ERROR) {
            return (mathErr, 0);
        }
(mathErr, principalTimesIndex) = subUInt(interestTimesIndex, 1);
if (mathErr != MathError.NO_ERROR) {
            return (mathErr, 0);
        }
(mathErr, interestAmountPrior) = mulUInt(principalTimesIndex, borrowSnapshot.principal);
if (mathErr != MathError.NO_ERROR) {
            return (mathErr, 0);
        }
```

修复状态: 这个问题在 commit: 222159f4dbcf54ac254185fbadd00a52ab46a2ae 中进行了修复。

4.3.3 低危漏洞

4.3.3.1 权限过大问题

获取价格的方式采用了 chainlink 的 Oracle，compound 的 Oracle，以及中心化的方式，通过 Token 的配置来决定要使用哪个方式获取价格，Owner 可以配置 Token 的取价方式，存在权限过大的风险。

SimplePriceOracle 合约中的 Owner 可以任意设置价格，存在权限过大的风险。

- contracts/oracle/SimplePriceOracle.sol

```
function setUnderlyingPrice(PToken pToken, uint price) public onlyOwner {
    address asset = _pETHUnderlying;
    if (!compareStrings(pToken.symbol(), "pETH")) {
        asset = address(PERC20(address(pToken)).underlying());
    }
    uint bt = block.timestamp;
    data[asset] = Datum(bt, price);
    emit PricePosted(asset, data[asset].price, price, price, bt);
}
function setPrice(address asset, uint price) public onlyOwner {
    uint bt = block.timestamp;
    emit PricePosted(asset, data[asset].price, price, price, bt);
    data[asset] = Datum(bt, price);
}
```

这个问题在 commit: 03b8b4d744e53436c6c78b25384f1d2a257b1cc8 中提供了新的设计方案，WePiggyPriceOracleV1 的 Owner 可以任意设置 Token 价格，设置 Token 的配置，存在权限过大的风险

- contracts/oracle/WePiggyPriceOracleV1.sol

```
function setPrice(address token, uint price, bool force) external override(WePiggyPriceOracleInterface) onlyOwner {
        Datum storage datum = data[token];
        if (force) {
            datum.value = price;
            datum.timestamp = block.timestamp;
        } else {
            TokenConfig storage config = configs[token];
            require(config.token == token, "bad params");
            uint upper = datum.value.mul(config.upperBoundAnchorRatio).div(1e2);
```

```
            uint lower = datum.value.mul(config.lowerBoundAnchorRatio).div(1e2);
            require(price.sub(lower) >= 0, "the price must greater than the old*lowerBoundAnchorRatio");
            require(upper.sub(price) >= 0, "the price must less than the old*upperBoundAnchorRatio");
            datum.value = price;
            datum.timestamp = block.timestamp;
        }
        emit PriceUpdated(token, price);
    }
function setTokenConfig(address token, string memory symbol, uint upperBoundAnchorRatio, uint lowerBoundAnchorRatio)
public onlyOwner {
        require(minLowerBoundAnchorRatio <= lowerBoundAnchorRatio, "lowerBoundAnchorRatio must greater or equal to
minLowerBoundAnchorRatio");
        require(maxUpperBoundAnchorRatio >= upperBoundAnchorRatio, "upperBoundAnchorRatio must Less than or equal to
maxUpperBoundAnchorRatio");
        TokenConfig storage config = configs[token];
        config.token = token;
        config.symbol = symbol;
        config.upperBoundAnchorRatio = upperBoundAnchorRatio;
        config.lowerBoundAnchorRatio = lowerBoundAnchorRatio;
        emit ConfigUpdated(token, symbol, upperBoundAnchorRatio, lowerBoundAnchorRatio);
    }
```

在 commit:72a028a8ea34765d73eef654ab3f48a1c575191b 中提供了 Provider 的代码，WePiggyPriceProviderV1 的 Owner 可以添加，更新 Token 的配置，存在权限过大的风险。

- contracts/oracle/WePiggyPriceProviderV1.sol

```
function addTokenConfig(address pToken, address underlying, string memory underlyingSymbol, uint256 baseUnit, bool
fixedUsd,
        address[] memory sources, PriceOracleType[] calldata sourceTypes) public onlyOwner {
        require(sources.length == sourceTypes.length, "sourceTypes.length must equal than sources.length");
        // add TokenConfig
        TokenConfig storage tokenConfig = tokenConfigs[pToken];
        require(tokenConfig.pToken == address(0), "bad params");
        tokenConfig.pToken = pToken;
        tokenConfig.underlying = underlying;
        tokenConfig.underlyingSymbol = underlyingSymbol;
        tokenConfig.baseUnit = baseUnit;
        tokenConfig.fixedUsd = fixedUsd;
        // add priceOracles
        require(oracles[pToken].length < 1, "bad params");
        for (uint i = 0; i < sources.length; i++) {
            PriceOracle[] storage list = oracles[pToken];
```

```
            list.push(PriceOracle({
            source : sources[i],
            sourceType : sourceTypes[i]
            }));
        }
        emit ConfigUpdated(pToken, underlying, underlyingSymbol, baseUnit, fixedUsd);
        emit PriceOracleUpdated(pToken, oracles[pToken]);
    }
function addOrUpdateTokenConfigSource(address pToken, uint256 index, address source, PriceOracleType _sourceType) public
onlyOwner {
        PriceOracle[] storage list = oracles[pToken];
        if (list.length > index) {//will update
            PriceOracle storage oracle = list[index];
            oracle.source = source;
            oracle.sourceType = _sourceType;
        } else {//will add
            list.push(PriceOracle({
            source : source,
            sourceType : _sourceType
            }));
        }
    }
function updateTokenConfigBaseUnit(address pToken, uint256 baseUnit) public onlyOwner {
        TokenConfig storage tokenConfig = tokenConfigs[pToken];
        require(tokenConfig.pToken != address(0), "bad params");
        tokenConfig.baseUnit = baseUnit;
        emit ConfigUpdated(pToken, tokenConfig.underlying, tokenConfig.underlyingSymbol, baseUnit, tokenConfig.fixedUsd);
    }
function updateTokenConfigFixedUsd(address pToken, bool fixedUsd) public onlyOwner {
        TokenConfig storage tokenConfig = tokenConfigs[pToken];
        require(tokenConfig.pToken != address(0), "bad params");
        tokenConfig.fixedUsd = fixedUsd;
        emit ConfigUpdated(pToken, tokenConfig.underlying, tokenConfig.underlyingSymbol, tokenConfig.baseUnit, fixedUsd);
    }
```

同理 PiggyDistribution，PToken 和 Comptroller 的 Owner 权限没有设置为 timelock 合约，建议将 PiggyDistribution，PToken 和 Comptroller 的 Owner 权限设置为 timelock 合约。

修复状态：暂未修复。

4.3.4 增强建议

4.3.4.1 签名重放问题

delegateBySig 函数 nonce 是由用户自己传入的参数进行签名的，当用户传了一个较大的 nonce 时，当前交易无法通过校验但是相关的签名数据仍会留在链上，导致此签名可能在未来某个时间段可用。建议参考 eip-2612 进行修复。

参考：https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2612.md#implementation

- wepiggy-contracts/contracts/token/WePiggyToken.sol

```
function delegateBySig(
        address delegatee,
        uint nonce,
        uint expiry,
        uint8 v,
        bytes32 r,
        bytes32 s
    )
    external
    {
        bytes32 domainSeparator = keccak256(
            abi.encode(
                DOMAIN_TYPEHASH,
                keccak256(bytes(name())),
                getChainId(),
                address(this)
            )
        );

        bytes32 structHash = keccak256(
            abi.encode(
                DELEGATION_TYPEHASH,
                delegatee,
                nonce,
                expiry
            )
        );
```

```
        bytes32 digest = keccak256(
            abi.encodePacked(
                "\x19\x01",
                domainSeparator,
                structHash
            )
        );

        address signatory = ecrecover(digest, v, r, s);
        require(signatory != address(0), "WePiggyToken::delegateBySig: invalid signature");
        require(nonce == nonces[signatory]++, "WePiggyToken::delegateBySig: invalid nonce");
        require(now <= expiry, "WePiggyToken::delegateBySig: signature expired");
        return _delegate(signatory, delegatee);
```

修复状态: 这个问题由于不直接影响项目的安全性，属于增强点，在保证签名的 nonce 准确的情况不会有该问题，因此暂时忽略。

5. 审计结果

5.1 结论

审计结果: 低风险

审计编号: 0X002012140003

审计日期: 2020 年 12 月 14 日

审计团队: 慢雾安全团队

总结: 慢雾安全团队采用人工结合内部工具对代码进行分析，审计期间发现了 2 个高危漏洞，2 个中危漏洞，1 个低危漏洞，1 个增强建议。其中高危，中危漏洞已经进行了修复，Owner 权限过大的问题经过沟通将后续将通过 timelock 机制进行缓解,目前 Owner 采用了多签合约进行管理，还未将权限移交给 timelock 合约，有 1 个增强建议暂时被忽略。

6. 声明

厦门慢雾科技有限公司(下文简称 ”慢雾 ”) 仅就本报告出具前项目方已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后项目方发生或存在的未知漏洞及安全事件，慢雾无法判断其安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料(简称“已提供资料”)。慢雾假设: 已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任，慢雾仅对该项目的安全情况进行约定内的安全审计并出具了本报告，慢雾不对该项目背景及其他情况进行负责。

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号