Smart Contract Security Audit Report

Contents

1. Executive Summary

On June. 04, 2021, the SlowMist security team received the WePiggy team's security audit application for WePiggy phase III code, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

SlowMist Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce

	in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack
- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision

- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

WePiggy is an open source, non-custodial crypto asset lending market protocol. In WePiggy's market, users can deposit their crypto assets to earn interest, or borrow others by paying interests.

Project website:

https://wepiggy.com

Audit version code:

https://github.com/WePiggy/wepiggy-contracts/tree/04daaef5253d5dd91e4ddf2482bf022ff088cbe7

Fixed version code:

https://github.com/WePiggy/wepiggy-contracts/tree/528c2557a2b27fb87a1e25ccd50f77c5799e170c

4. Code Overview

4.1 Contracts Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

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	-
_setMarketMintCaps	external	Can modify state	onlyOwner

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
setWpcSpeedInternal	Internal	Can Modify State	-
updateWpcSupplyIndex	Internal	Can Modify State	-
updateWpcBorrowIndex	Internal	Can Modify State	-
distributeSupplierWpc	Internal	Can Modify State	-
distributeBorrowerWpc	Internal	Can Modify State	-
grantWpcInternal	Internal	Can Modify State	-
claimWpc	Public	Can Modify State	-
claimWpc	Public	Can Modify State	-
claimWpc	Public	Can Modify State	-
_setWpcSpeed	Public	Can Modify State	onlyOwner
_setEnableWpcClaim	Public	Can Modify State	onlyOwner
_setEnableDistributeMintWpc	Public	Can Modify State	onlyOwner

_setEnableDistributeRedeemWpc	Public	Can Modify State	onlyOwner
_setEnableDistributeBorrowWpc	Public	Can Modify State	onlyOwner
_setEnableDistributeRepayBorrowWpc	Public	Can Modify State	onlyOwner
_setEnableDistributeSeizeWpc	Public	Can Modify State	onlyOwner
_setEnableDistributeTransferWpc	Public	Can Modify State	onlyOwner
_setEnableAll	Public	Can Modify State	onlyOwner
_transferWpc	Public	Can Modify State	onlyOwner
_transferToken	Public	Can Modify State	onlyOwner
pendingWpcAccrued	Public	-	-
pendingWpcInternal	Internal	-	-
pendingWpcBorrowInternal	Internal	-	-
pendingWpcBorrowIndex	Internal	-	-
pendingWpcSupplyInternal	Internal	-	-
pendingWpcSupplyIndex	Internal	-	-

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	-
flashloan	external	Can modify state	nonReentrant
_setFlashloan	public	Can modify state	onlyOwner

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	-	-
flashloan	external	Can modify state	nonReentrant
_setFlashloan	public	Can modify state	onlyOwner

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 Contract Information

4.3 Code Audit

The deployment of ptoken adopts an upgradable model, The PToken and COMPTROLLER Owner is a Multi-sign contract(0xF4Fa2fFAdBCafcD6C0D615C9A262DcB8e5F0f94d). The admin of the adminupgradability proxy contract is managed by the EOA address of the project party.

Address of the contract on the heco main network:

Contract Name	Contract Address
WePiggyToken	0xb205d0AeF84C666FBBe441C61DC04fEb844444E6
WePiggyPriceProviderV1	0x4C78015679FabE22F6e02Ce8102AFbF7d93794eA
AdminUpgradeabilityProxy (WePiggyPriceOracleV1)	0xFfceAcfD39117030314A07b2C86dA36E51787948
WePiggyPriceOracleV1(Implementation)	0xd58fb16eace4693b2c641cae6850a82763c00a34
AdminUpgradeabilityProxy (Comptroller)	0x3401D01E31BB6DefcFc7410c312C0181E19b9dd5
Comptroller(Implementation)	0x8c925623708a94c7de98a8e83e8200259ff716e0
AdminUpgradeabilityProxy(PiggyDistribution)	0x8b4397A92D53916f24a8E06777CEf4485281224C
PiggyDistribution(Implementation)	0x92aabcd4c83da8859fb44e9142c00eeb8f52114a
AdminUpgradeabilityProxy(STABLECOIN_JUMP_RATE_MODEL)	0xd1121aDe04EE215524aeFbF7f8D45029214d668D
MAX_IMILLION	0x8158B34fF8A36dD9E4519d62C52913C24ad5554b
JumpRateModel(Implementation)	0xc1b02e52e9512519edf99671931772e452fb4399

AdminUpgradeabilityProxy(BTC_ETH_JUMP_RATE_MODEL)	0x621CE6596E0B9CcF635316BFE7FdBC80C3029Bec
AdminUpgradeabilityProxy(MAINSTREAM_JUMP_RATE_MODEL)	0x8e1e582879Cb8baC6283368e8ede458B63F499a5
pHT	0x75DCd2536a5f414B8F90Bb7F2F3c015a26dc8c79
pHUSD	0x311aEA58Ca127B955890647413846E351df32554
pUSDT	0x12D803497D1e58dD4D4A4F455D754f1d0F937C8b
pUSDC	0x2a8Cd78bFb91ACF53f589961D213d87c956e0d7f
pETH	0x2B7F68170a598E507B19Bca41ED745eABc936B3F
pHBTC	0x2dd8FFA7923a17739F70C34759Af7650e44EA3BE
pHPT	0x811Cd5CB4cC43F44600Cfa5eE3F37a402C82aec2
pHDOT	0x17933112E9780aBd0F27f2B7d9ddA9E840D43159
pHLTC	0x417FDfC74503d8008AeEB53248E5C0f1960c2C1d
pHBCH	0xe212829Ca055eD63279753971672c693C6C6d088
pMDX	0x30ac79B557973771c931D8d765E0728261A742a0
pHFIL	0x0C8c1ab017c3C0c8A48dD9F1DB2F59022D190f0b
pUNI	0xd828F7029CC58C4E9Cab3B1E0726CEFab411bc65
PEther(Implementation)	0x33a32f0ad4aa704e28c93ed8ffa61d50d51622a7
PERC20(Implementation)	0x849c37a029b38d3826562697ccc40c34477c6293

4.3.1 Low-risk vulnerabilities

4.3.1.1 Excessive authority auditing

The method of feeding prices uses chainlink's Oracle, compound's Oracle, and a centralized method. The token configuration determines which method to use to feed the price. The Owner can configure the token's feed price method, The Owner in the SimplePriceOracle contract can set the price arbitrarily, and there is a risk of excessive authority.

- 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);
    }
```

Fix Status: The implementation of the code has been modified in this commit:

03b8b4d744e53436c6c78b25384f1d2a257b1cc8

The Owner of WePiggyPriceOracleV1 can arbitrarily set the token price and set the token configuration, there is a risk of excessive authority.

- 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);
    }
```

Fix Status: The implementation of the code has been modified in commit:

72a028a8ea34765d73eef654ab3f48a1c575191b

The Owner of WePiggyPriceProviderV1 can add and update the token configuration, there is a risk of excessive authority.

- 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);
    }
```

The owner of PiggyDistribution, PToken and Comptroller contracts is not set to the timelock contract. It is recommended to transfer the owner authority of PiggyDistribution, PToken and Comptroller to the timelock contract.

Fix Status: Waiting for fix.

4.3.2 Enhancement Suggestions

4.3.2.1 Enhancement Point of DelegateBySig Function

The nonce in the delegateBySig function is input by the user. When the user input a larger nonce, the current transaction cannot be success but the relevant signature data will still remain on the chain, causing this signature to be available for some time in the future. It is recommended to fix it according to EIP-2612.

Reference: 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);
```

Fix Status: This issues has been ignore.

4.3.2.2 Missing event record

In the new business logic method of lightning loan business, event records are missing. It is recommended to add events to record so that community users can review the changes of the contract parameters.

- contracts/token/PERC20.sol

```
function flashloan(address _receiver, uint256 _amount, bytes memory _params) nonReentrant external {

        uint256 availableLiquidityBefore = getCashPrior();

        address payable fl = address(uint160(address(flashloanInstance)));
        doTransferOut(fl, _amount);
        flashloanInstance.flashloan(address(this), _receiver, underlying, _amount, _params);

        uint availableLiquidityAfter = getCashPrior();
        require(availableLiquidityAfter >= availableLiquidityBefore, "The actual balance of the protocol is inconsistent");

        accrueInterest();
    }

function _setFlashloan(address _flashloan) public onlyOwner {
        flashloanInstance = IFlashloan(_flashloan);
    }
```

- contracts/token/PEther.sol

```
function flashloan(address _receiver, uint256 _amount, bytes memory _params) nonReentrant external {
        uint256 cashBefore = getCashPrior();
        doTransferOut(address(uint160(address(flashloanInstance))), _amount);
        flashloanInstance.flashloan(address(this), _receiver, address(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE), _amount, _params);
        require(getCashPrior() >= cashBefore, "The actual balance is inconsistent");
        accrueInterest();
    }

function _setFlashloan(address _flashloan) public onlyOwner {
        flashloanInstance = IFlashloan(_flashloan);
    }
```

- contracts/flashloan/Flashloan.sol

```
    function registerActiveCaller(address[] memory callers) public onlyOwner {
```

```
        for (uint i = 0; i < callers.length; i++) {
            address caller = callers[i];
            activeCaller[caller] = true;
        }
    }

    function unRegisterActiveCaller(address[] memory callers) public onlyOwner {
        for (uint i = 0; i < callers.length; i++) {
            address caller = callers[i];
            activeCaller[caller] = false;
        }
    }

    function registerActiveReceiver(address[] memory receivers) public onlyOwner {
        for (uint i = 0; i < receivers.length; i++) {
            address receiver = receivers[i];
            activeReceiver[receiver] = true;
        }
    }

    function unRegisterActiveReceiver(address[] memory receivers) public onlyOwner {
        for (uint i = 0; i < receivers.length; i++) {
            address receiver = receivers[i];
            activeReceiver[receiver] = false;
        }
    }
```

Fix Status: The issue of missing event records in PEther.sol and PERC20.sol has not been fixed yet. The issue of missing event records in the Flashloan.sol has been fixed in commit: 528c2557a2b27fb87a1e25ccd50f77c5799e170c.

5. Audit Result

5.1 Conclusion

Audit Result : Low Risk

Audit Number : 0X002106110004

Audit Date : June. 11, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, 1 low-risk vulnerability and 2 enhancement suggestions were found during the audit, the owner authority has not been transferred to the timelock contract. The PToken and COMPTROLLER Owner is a Multi-sign contract ,There is an enhancement suggestion has been ignore.

6. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist