Smart Contract Security Audit Report

Contents

1. Executive Summary

The SlowMist Security Team received the OpenMoneyMarket team's application for smart contract security audit of the OpenMoneyMarket on Feb. 27, 2021. The following are the details and results of this smart contract security audit.

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

Omm is a money market powered by blockchain, where anyone can lend and borrow assets, earn interest, and own a share of the protocol. It will be a fair launch protocol with no pre-mined tokens. It will start off as a pillar of the ICON DeFi ecosystem, but plans to evolve as a cross-chain money market. Its ultimate goal is to bridge the gap between high interest rates offered in the crypto space and regular consumers who are looking for attractive high yields. It will have a simple user onboarding experience, including fiat on-ramp and private key management, so that end users don't have to worry about interacting with a blockchain service. 10% of Borrowers' interest will be distributed to the DAO fund operated by OMM token holders.

Audit version code:

https://github.com/openmoneymarket/openmoneymarket-mono (development)

git commit: c7ca308107a36fdd9d06cd31fa104092fcc655a0

Fixed version code:

https://github.com/openmoneymarket/openmoneymarket-mono (audit-bug-fixes)

git commit: 2e7bb5a18b1a1c38314516f29839d07c2a11d9f2

3.2 Project Structure

```
score
├── CleanInstall.ipynb
├── D_cleaninstall.ipynb
├── Deployer.ipynb
├── README.md
├── Rewards.ipynb
├── SCORE.ipynb
├── addressProvider
│   ├── __init__.py
│   ├── addressProvider.py
│   ├── package.json
│   └── utils
│       └── checks.py
├── contracts_20201106101156.pkl
├── contracts_20201106101231.pkl
├── contracts_20201108141815.pkl
├── contracts_20201109050405.pkl
├── contracts_20201109120425.pkl
├── custom_contracts_20201110125748.pkl
├── custom_contracts_20201122161857.pkl
├── custom_contracts_20201125111607.pkl
├── custom_contracts_20210111125649.pkl
├── customcontracts_20210118060735.pkl
├── daoFund
│   ├── __init__.py
│   ├── daoFund.py
│   └── package.json
├── delegation
│   ├── Math.py
│   ├── __init__.py
│   ├── delegation.py
│   ├── package.json
```

```
|   └── utils
|       ├── __init__.py
|       └── checks.py
├── feeProvider
|   ├── Math.py
|   ├── __init__.py
|   ├── feeProvider.py
|   ├── package.json
|   └── utils
|       ├── __init__.py
|       └── checks.py
├── governance
|   ├── __init__.py
|   ├── governance.py
|   ├── package.json
|   └── utils
|       ├── __init__.py
|       └── checks.py
├── lendingPool
|   ├── __init__.py
|   ├── lendingPool.py
|   ├── package.json
|   └── utils
|       └── checks.py
├── lendingPoolCore
|   ├── Math.py
|   ├── ReserveData.py
|   ├── UserData.py
|   ├── __init__.py
|   ├── lendingPoolCore.py
|   ├── package.json
|   └── utils
|       ├── __init__.py
|       └── checks.py
├── lendingPoolDataProvider
|   ├── Math.py
|   ├── __init__.py
|   ├── lendingPoolDataProvider.py
|   ├── package.json
|   └── utils
|       ├── __init__.py
|       └── checks.py
├── liquidationManager
```

```
│   ├── Math.py
│   ├── __init__.py
│   ├── liquidationManager.py
│   ├── package.json
│   └── utils
│       ├── __init__.py
│       └── checks.py
├── oToken
│   ├── IIRC2.py
│   ├── Math.py
│   ├── __init__.py
│   ├── oToken.py
│   ├── package.json
│   └── utils
│       ├── __init__.py
│       └── checks.py
├── ommToken
│   ├── __init__.py
│   ├── omm.py
│   ├── package.json
│   ├── tokens
│   │   ├── IIRC2.py
│   │   ├── IRC2.py
│   │   ├── IRC2mintable.py
│   │   └── __init__.py
│   └── utils
│       ├── __init__.py
│       ├── checks.py
│       └── consts.py
├── priceOracle
│   ├── __init__.py
│   ├── package.json
│   ├── priceOracle.py
│   └── utils
│       ├── __init__.py
│       └── checks.py
├── repeater.py
├── rewards
│   ├── Math.py
│   ├── __init__.py
│   ├── package.json
│   ├── rewards.py
│   └── utils
```

```
|       └── checks.py
├── sample_token
|   ├── __init__.py
|   ├── package.json
|   └── sample_token.py
├── shubh_yeouido_contracts_20210112084238.pkl
├── sicx
|   ├── __init__.py
|   ├── package.json
|   └── sICX.py
├── snapshot
|   ├── __init__.py
|   ├── package.json
|   ├── snapshot.py
|   └── utils
|       └── checks.py
├── superAddressProvider
|   ├── __init__.py
|   ├── package.json
|   ├── superAddressProvider.py
|   └── utils
|       └── checks.py
├── updated20201125_custom_contracts_20201122161857.pkl
├── worker_token
|   ├── __init__.py
|   ├── package.json
|   └── worker_token.py
└── yeouido_contracts_20210112084203.pkl
```

4. Code Overview

4.1 Contract Information

Currently, the contracts have not yet been deployed to the main network.

4.2 Contracts Description

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

follows:

AddressProvider			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
setLendingPool	external	can modify state	only_owner
setLendingPoolDataProvider	external	can modify state	only_owner
setUSDb	external	can modify state	only_owner
setsICX	external	can modify state	only_owner
setoUSDb	external	can modify state	only_owner
setoICX	external	can modify state	only_owner
setStaking	external	can modify state	only_owner
setIUSDC	external	can modify state	only_owner
setoIUSDC	external	can modify state	only_owner
setOmmToken	external	can modify state	only_owner
setDelegation	external	can modify state	only_owner
setRewards	external	can modify state	only_owner
getAllAddresses	external	-	-

DaoFund			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
FundReceived	internal	-	-
tokenFallback	external	can modify state	-

Delegation			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
_require	internal	-	-
name	external	-	-

setOmmToken	external	can modify state	only_owner
getOmmToken	external	-	-
setLendingPoolCore	external	can modify state	only_owner
getLendingPoolCore	external	-	-
clearPrevious	internal	-	-
getPrepList	external	-	-
updateDelegations	external	can modify state	-
prepVotes	external	-	-
getUserDelegationDetails	external	-	-
computeDelegationPercentages	external	-	-

FeeProvider			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
setLoanOriginationFeePercentage	external	can modify state	only_owner
calculateOriginationFee	external	-	-
getLoanOriginationFeePercentage	external	-	-
tokenFallback	external	can modify state	-

Governance			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
setSnapshot	external	can modify state	only_owner
getSnapshot	external	-	-
setRewards	external	can modify state	only_owner
getRewards	external	-	-
setStartTimestamp	external	can modify state	only_owner

LendingPool			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-

on_update	internal	-	-
Deposit	internal	-	-
Borrow	internal	-	-
RedeemUnderlying	internal	-	-
Repay	internal	-	-
setLendingPoolCore	external	can modify state	only_owner
getLendingPoolCore	external	-	-
setLiquidationManager	external	can modify state	only_owner
getLiquidationManager	external	-	-
setSICX	external	can modify state	only_owner
getSICX	external	-	-
setOICX	external	can modify state	only_owner
getOICX	external	-	-
setStaking	external	can modify state	only_owner
getStaking	external	-	-
setReward	external	can modify state	only_owner
getReward	external	-	-
setLendingPoolDataProvider	external	can modify state	only_owner
getLendingPoolDataProvider	external	-	-
setFeeProvider	external	can modify state	only_owner
getFeeProvider	external	-	-
setSnapshot	external	can modify state	only_owner
getSnapshot	external	-	-
getBorrowWallets	external	-	-
getDepositWallets	external	-	-
deposit	external	payable	-
_deposit	internal	-	-
redeemUnderlying	external	can modify state	-
_require	internal	-	-
borrow	external	can modify state	-
_repay	internal	-	-
liquidationCall	external	payable	-
_updateSnapshot	internal	-	-
tokenFallback	external	can modify state	-

LendingPoolCore			
Function Name	Visibility	Mutability	Modifiers

__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
ReserveUpdated	internal	-	-
DaoFundTransfer	internal	-	-
name	external	-	-
set_id	external	can modify state	only_owner
get_id	external	-	-
setSymbol	external	can modify state	only_owner
getSymbol	external	-	-
setStaking	external	can modify state	only_owner
getStaking	external	-	-
setLendingPool	external	can modify state	only_owner
getLendingPool	external	-	-
setLiquidationManager	external	can modify state	only_owner
getLiquidationManager	external	-	-
setDelegation	external	can modify state	only_owner
getDelegation	external	-	-
setPriceOracle	external	can modify state	only_owner
getPriceOracle	external	-	-
getDaoFund	external	-	-
setDaoFund	external	can modify state	only_owner
reservePrefix	internal	-	-
userReservePrefix	internal	-	-
updateTotalBorrows	internal	-	-
updateLastUpdateTimestamp	internal	-	-
updateLiquidityRate	internal	-	-
updateBorrowRate	internal	-	-
updateBorrowCumulativeIndex	internal	-	-
updateLiquidityCumulativeIndex	internal	-	-
updateBaseLTVasCollateral	internal	-	-
updateLiquidationThreshold	internal	-	-
updateLiquidationBonus	internal	-	-
updateDecimals	internal	-	-
updateBorrowingEnabled	internal	-	-
updateUsageAsCollateralEnabled	internal	-	-
updateIsFreezed	internal	-	-
updateIsActive	internal	-	-

updateOtokenAddress	internal	-	-
updateUserPrincipalBorrowBalance	internal	-	-
updateUserBorrowCumulativeIndex	internal	-	-
updateUserLastUpdateTimestamp	internal	-	-
updateUserOriginationFee	internal	-	-
updateUserReserveUseAsCollateral	internal	-	-
_check_reserve	internal	-	-
getReserves	external	-	-
_addNewReserve	internal	-	-
isReserveBorrowingEnabled	external	-	-
addReserveData	external	can modify state	only_owner
getReserveData	external	-	-
addUserReserveData	external	can modify state	-
getUserReserveData	external	-	-
enableAsCollateral	external	can modify state	-
disableAsCollateral	external	can modify state	-
enableBorrowing	external	can modify state	-
disableBorrowing	external	can modify state	-
calculateLinearInterest	internal	-	-
calculateCompoundedInterest	internal	-	-
getNormalizedIncome	external	-	-
updateCumulativeIndexes	internal	-	-
_increaseTotalBorrows	internal	-	-
_decreaseTotalBorrows	internal	-	-
getCompoundedBorrowBalance	external	-	-
getReserveAvailableLiquidity	external	-	-
getReserveTotalLiquidity	internal	-	-
getReserveNormalizedIncome	internal	-	-
getReserveUtilizationRate	internal	-	-
getReserveConfiguration	external	-	-
updateReserveInterestRatesAndTimestampInternal	internal	-	-
setReserveConstants	external	can modify state	only_owner
getReserveConstants	external	-	-
transferToUser	external	can modify state	only_lending_pool
liquidateFee	external	can modify state	only_liquidation
updateStateOnDeposit	external	can modify state	only_lending_pool
updateStateOnRedeem	external	can modify state	only_lending_pool
updateStateOnBorrow	external	can modify state	only_lending_pool

updateStateOnRepay	external	can modify state	only_lending_pool
updateReserveStateOnBorrowInternal	internal	-	-
updateReserveStateOnRepayInternal	internal	-	-
updateReserveTotalBorrows	internal	-	-
getCurrentBorrowRate	internal	-	-
updateUserStateOnBorrowInternal	internal	-	-
updateUserStateOnRepayInternal	internal	-	-
getReserveOTokenAddress	external	-	-
updateStateOnLiquidation	external	can modify state	only_lending_pool
updatePrincipalReserveStateOnLiquidationInternal	internal	-	-
updateCollateralReserveStateOnLiquidationInternal	internal	-	-
updateUserStateOnLiquidationInternal	internal	-	-
setUserUseReserveAsCollateral	internal	-	-
getUserUnderlyingAssetBalance	external	-	-
getUserOriginationFee	external	-	-
getUserBasicReserveData	external	-	-
getUserBorrowBalances	external	-	-
calculateInterestRates	internal	-	-
updatePrepDelegations	external	can modify state	only_delegation
tokenFallback	external	can modify state	-

LendingPoolDataProvider			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
name	external	-	-
setSymbol	external	can modify state	only_owner
getSymbol	external	-	-
setLendingPoolCore	external	can modify state	only_owner
setLendingPool	external	can modify state	only_owner
setFeeProvider	external	can modify state	only_owner
setPriceOracle	external	can modify state	only_owner
setStaking	external	can modify state	only_owner
getLendingPoolCore	external	-	-
getLendingPool	external	-	-
getPriceOracle	external	-	-

getFeeProvider	external	-	-
setLiquidationManager	external	can modify state	only_owner
getLiquidationManager	external	-	-
getStaking	external	-	-
getReserveAccountData	external	-	-
getUserAccountData	external	-	-
getUserReserveData	external	-	-
balanceDecreaseAllowed	external	-	-
calculateCollateralNeededUSD	external	-	-
getUserAllReserveData	external	-	-
getUserLiquidationData	external	-	-
liquidationList	external	-	-
calculateHealthFactorFromBalancesInternal	internal	-	-
calculateBorrowingPowerFromBalancesInternal	internal	-	-
getReserveData	external	-	-
getAllReserveData	external	-	-
getReserveConfigurationData	external	-	-
getAllReserveConfigurationData	external	-	-
getUserUnstakeInfo	external	-	-
getLoanOriginationFeePercentage	external	-	-

LiquidationManager			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
OriginationFeeLiquidated	internal	-	-
LiquidationCall	internal	-	-
name	external	-	-
setLendingPoolDataProvider	external	can modify state	only_owner
getLendingPoolDataProvider	external	-	-
setFeeProvider	external	can modify state	only_owner
getFeeProvider	external	-	-
setLendingPoolCore	external	can modify state	only_owner
getLendingPoolCore	external	-	-
setPriceOracle	external	can modify state	only_owner
getPriceOracle	external	-	-

calculateBadDebt	external	-	-
calculateAvailableCollateralToLiquidate	internal	-	-
calculateCurrentLiquidationThreshold	internal	-	-
liquidationCall	external	can modify state	-

OToken			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
Transfer	internal	-	-
Mint	internal	-	-
Burn	internal	-	-
Redeem	internal	-	-
MintOnDeposit	internal	-	-
BurnOnLiquidation	internal	-	-
BalanceTransfer	internal	-	-
onlyOwner	internal	-	-
__wrapper	internal	-	-
name	external	-	-
symbol	external	-	-
decimals	external	-	-
totalSupply	external	-	-
setLendingPoolCore	external	can modify state	onlyOwner
getLendingPoolCore	external	-	-
setLiquidation	external	can modify state	onlyOwner
getLiquidation	external	-	-
setReserve	external	can modify state	onlyOwner
getReserve	external	-	-
setLendingPoolDataProvider	external	can modify state	onlyOwner
getLendingPoolDataProvider	external	-	-
setLendingPool	external	can modify state	onlyOwner
getLendingPool	external	-	-
getUserLiquidityCumulativeIndex	external	-	-
_calculateCumulatedBalanceInternal	internal	-	-
_cumulateBalanceInternal	internal	-	-
balanceOf	external	-	-

principalBalanceOf	external	-	-
isTransferAllowed	external	-	-
redeem	external	can modify state	-
_resetDataOnZeroBalanceInternal	internal	-	-
mintOnDeposit	external	can modify state	only_lending_pool
burnOnLiquidation	external	can modify state	only_liquidation
_executeTransfer	internal	-	-
transfer	external	can modify state	-
_transfer	internal	-	-
_mint	internal	-	-
_burn	internal	-	-

PriceOracle			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
toggleOraclePriceBool	external	can modify state	only_owner
getOraclePriceBool	external	-	-
setBandOracle	external	can modify state	only_owner
getBandOracle	internal	-	-
set_reference_data	external	can modify state	only_owner
get_reference_data	external	-	-

Rewards			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
Distribution	internal	-	-
State	internal	-	-
setDistPercentage	external	can modify state	only_owner
setRecipients	external	can modify state	only_owner
getRecipients	external	-	-
getDistPercentage	external	-	-
setStartTimestamp	external	can modify state	only_admin
getStartTimestamp	external	-	-

setLendingPool	external	can modify state	only_owner
getLendingPool	external	-	-
setOmm	external	can modify state	only_owner
getOmm	external	-	-
setAdmin	external	can modify state	only_owner
getAdmin	external	-	-
setLendingPoolCore	external	can modify state	only_owner
getLendingPoolCore	external	-	-
setDaoFund	external	can modify state	only_owner
getDaoFund	external	-	-
setLpToken	external	can modify state	only_owner
getLpToken	external	-	-
setWorkerToken	external	can modify state	only_owner
getWorkerToken	external	-	-
setSnapshot	external	can modify state	only_owner
getSnapshot	external	-	-
_check	internal	-	-
distribute	external	can modify state	-
claimRewards	external	can modify state	-
getRewards	external	-	-
_getDay	external	-	-
_initialize	internal	-	-
_depositBalance	internal	-	-
_borrowBalance	internal	-	-
_calculateLinearInterest	internal	-	-
_calculateCompoundedInterest	internal	-	-
tokenDistributionPerDay	external	-	-
tokenFallback	external	can modify state	-

SampleToken			
Function Name	Visibility	Mutability	Modifiers
Transfer	internal	-	-
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
name	external	-	-
symbol	external	-	-

decimals	external	-	-
totalSupply	external	-	-
balanceOf	external	-	-
transfer	external	can modify state	-
_transfer	internal	-	-

Sicx			
Function Name	Visibility	Mutability	Modifiers
Transfer	internal	-	-
Mint	internal	-	-
Burn	internal	-	-
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
name	external	-	-
symbol	external	-	-
decimals	external	-	-
totalSupply	external	-	-
balanceOf	external	-	-
transfer	external	can modify state	-
_transfer	internal	-	-
add_collateral	external	can modify state	-
_mint	internal	-	-
_burn	internal	-	-

Snapshot			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
setStartTimestamp	external	can modify state	can modify state
getStartTimestamp	external	-	-
setAdmin	external	can modify state	can modify state
getAdmin	external	-	-
setGovernance	external	can modify state	can modify state
getGovernance	external	-	-
userDataAt	external	-	-

reserveDataAt	external	-	-
updateUserSnapshot	external	can modify state	can modify state
updateReserveSnapshot	external	can modify state	can modify state
_getDay	external	-	-
getStartTimestamp	external	-	-

SuperAddressProvider			
Function Name	Visibility	Mutability	Modifiers
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
addEnv	external	can modify state	only_owner
getIdList	external	can modify state	-
getEnv	external	-	-

WorkerToken			
Function Name	Visibility	Mutability	Modifiers
Transfer	internal	-	-
__init__	internal	-	-
on_install	internal	-	-
on_update	internal	-	-
name	external	-	-
symbol	external	-	-
decimals	external	-	-
totalSupply	external	-	-
balanceOf	external	-	-
transfer	external	can modify state	-
_transfer	internal	-	-
getWallets	external	-	-

4.3 Code Audit

[High] [N03] Anyone can set LoanOriginationFeePercentage

Content

score/feeProvider/feeProvider.py 20-22 lines

```
@external
def setLoanOriginationFeePercentage(self, _percentage: int) -> None:
    self._originationFeePercent.set(_percentage)
```

Solution

Add onlyOwner modifier

Status

Fixed on pull/28

[High] [N04] Governance external set functions don't have onlyOwner or onlyAdmin modifier

Content

score/governance/governance.py, 35-37 lines,

```
@external
def setSnapshot(self, _address: Address):
    self._snapshot.set(_address)
```

score/governance/governance.py, 43-45 lines,

```
  @external
   def setRewards(self, _address: Address):
       self._rewards.set(_address)
```

score/governance/governance.py 52-57 lines,

```
@external
def setStartTimestamp(self) -> None:
    snapshot = self.create_interface_score(self._snapshot.get(), SnapshotInterface)
    rewards = self.create_interface_score(self._rewards.get(), RewardInterface)
    snapshot.setStartTimestamp(self.now())
    rewards.setStartTimestamp(self.now())
```

Look like anyone can call these functions.

Solution

Add @onlyOwner or @onlyAdmin modifier.

Status

Fixed on pull/28

[Suggestion] [N05] user = self.tx.origin: why not use self.msg.sender instead of self.tx.origin

Content

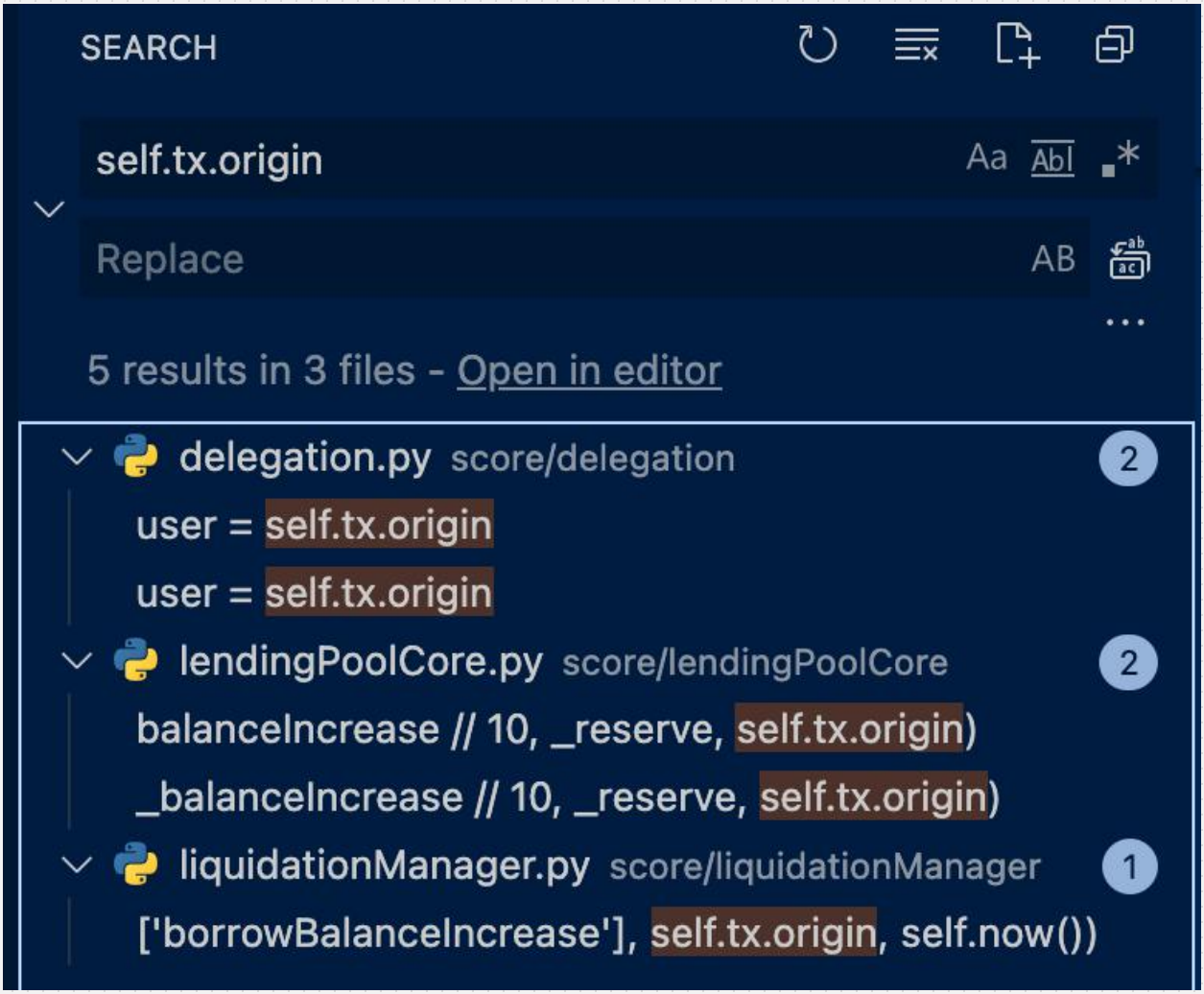
score/delegation/delegation.py 107-153 lines,

```
@external
def updateDelegations(self, _delegations: List[PrepDelegations] = None):
    delegations = []
    user = self.tx.origin

    ...
```

self.tx.origin is not the immediate call user, the origin user may be exploited to call this method. If the user means who calls the function, suggest use self.msg.sender.

There are also some other files that use self.tx.origin, please check all of them.



Solution

Use self.msg.sender instead of self.tx.origin.

Status

Fixed on pull/29

[Suggestion] [N07] feeProvider calculateOriginationFee function never uses _user param

Content

score/feeProvider/feeProvider.py 24-26 lines

```
@external(readonly = True)
    def calculateOriginationFee(self, _user: Address, _amount: int) -> int:
        return exaMul(_amount, self.getLoanOriginationFeePercentage())
```

Solution

If a function never uses a param, remove it.

Status

Fixed on pull/28

[High] [N08] Anyone can call mintOnDeposit to mint tokens to any address for any amount

Content

score/oToken/oToken.py 296-303 lines

```
@external
def mintOnDeposit(self, _user: Address, _amount: int) -> None:
    cumulated = self._cumulateBalanceInternal(_user)


    balanceIncrease = cumulated['balanceIncrease']
    index = cumulated['index']
    self._mint(_user, _amount)
    self.MintOnDeposit(_user, _amount, balanceIncrease, index)
```

The mintOnDeposit it external, and no judgement to forbid the user to mint tokens.

Solution

Add deposit logical, or change the function to be internal.

Status

Fixed on pull/29

[High] [N09] Anyone can call burnOnLiquidation to burn any address balance

Content

score/oToken/oToken.py 305-317 lines,

```
@external
def burnOnLiquidation(self, _user: Address, _value: int) -> None:
    cumulated = self._cumulateBalanceInternal(_user)
    currentBalance = cumulated['principalBalance']
    balanceIncrease = cumulated['balanceIncrease']
    index = cumulated['index']
    self._burn(_user, _value)
    userIndexReset = False
    if currentBalance - _value == 0:
        userIndexReset = self._resetDataOnZeroBalanceInternal(_user)
    if userIndexReset:
        index = 0
    self.BurnOnLiquidation(_user, _value, balanceIncrease, index)
```

it's external, anyone can call, and then self._burn(_user, _value).

Solution

No Liquidation part? The function seems not completed.

Status

Fixed on pull/29

[Suggestion] [N10] some functions (_cumulateBalanceInternal...) like internal functions but external

Content

in score/oToken/oToken.py file,

we can see *_calculateCumulatedBalanceInternal, _cumulateBalanceInternal, these function names has "Internal" and prefix "_", when we use "Internal" and "_"* as prefix on a function name, we would think it is an internal function, but these functions are external.

Solution

they should be Internal? Or suggest that remove "Internal" and "_" from the function name.

Status

Fixed on pull/29

[Medium] [N11] anyone can call redeemUnderlying to let other user do redeem operation

Content

score/lendingPool/lendingPool.py 312-341 lines,

```
@external
def redeemUnderlying(self, _reserve: Address, _user: Address, _amount: int, _oTokenbalanceAfterRedeem: int,
                     _waitForUnstaking: bool = False):
    """
    redeems the underlying amount of assets requested by the _user.This method is called from the oToken contract
    :param _reserve:the address of the reserve
    :param _user:the address of the user requesting the redeem
    :param _amount:the amount to be deposited, should be -1 if the user wants to redeem everything
    :param _oTokenbalanceAfterRedeem:the remaining balance of _user after the redeem is successful
    :return:
    """


    core = self.create_interface_score(self._lendingPoolCoreAddress.get(), CoreInterface)
    if core.getReserveAvailableLiquidity(_reserve) < _amount:
        revert(f'There is not enough liquidity available to redeem')


    reward = self.create_interface_score(self._rewardAddress.get(), RewardInterface)
    reward.distribute()


    core.updateStateOnRedeem(_reserve, _user, _amount, _oTokenbalanceAfterRedeem == 0)
    if _waitForUnstaking:
        self._require(self.msg.sender == self._oIcxAddress.get(),
                      "Redeem with wait for unstaking failed: Invalid token")
        transferData = "{\"method\": \"unstake\"}".encode("utf-8")
        core.transferToUser(_reserve, self._stakingAddress.get(), _amount, transferData)
        self.RedeemUnderlying(_reserve, _user, _amount, self.block.timestamp)
        return


    core.transferToUser(_reserve, _user, _amount)
    self.RedeemUnderlying(_reserve, _user, _amount, self.block.timestamp)
```

it's external, anyone call it to let other user do redeem operation, even if they don't want to do this.

Solution

Remove _user param, use self.msg.sender instead.

Status

Fixed on pull/29

[Suggestion] [N13] Add readonly=True to isReserveBorrowingEnabled function

Content

score/lendingPoolCore/lendingPoolCore.py 378-380 lines,

```
@external
def isReserveBorrowingEnabled(self, _reserve: Address) -> bool:
    return self.getReserveData(_reserve)['borrowingEnabled']
```

it's a read function, but no readonly set.

Solution

@external(readonly=True)

Status

Fixed on pull/29

[Suggestion] [N14] Do not need to judge whether is owner while using @only_owner modifier

Content

score/lendingPoolDataProvider/lendingPoolDataProvider.py 154-160 lines,

```
@only_owner
@external
def setStaking(self, _address: Address) -> None:
    if self.msg.sender != self.owner:
        revert(f'Method can only be invoked by the owner')
```

```
    self._staking.set(_address)
```

score/lendingPoolDataProvider/lendingPoolDataProvider.py 174-179 lines,

```
@only_owner
 @external
 def setLiquidationManager(self, _address: Address) -> None:
     if self.msg.sender != self.owner:
         revert(f'Method can only be invoked by the owner')
     self._liquidationManager.set(_address)
```

The judgement is redundant while the function has @only_owner modifier

Solution

Remove the judgement.

Status

Fixed

[High] [N15] Anyone can call set_id on lendingPoolCore

Content

score/lendingPoolCore/lendingPoolCore.py 160-162 lines,

```
@external
def set_id(self, _value: str):
    self._id.set(_value)
```

After this set, it would make some functions not useful.

Solution

add @only_onwer modifier.

Status

Fixed on pull/32

5. Audit Result

5.1 Conclusion

Audit Result : Passed

Audit Number : 0X002103220001

Audit Date : March. 22, 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, 5 high-risk vulnerabilities, 1 medium-risk vulnerability and 5 enhancement suggestions were found during the audit, and all the findings were fixed, some ones were ignored after discussed that no impact.

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