Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.04.08, the SlowMist security team received the Trustin team's security audit application for Trustin, 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.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules 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.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using 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:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Trustin Protocol offers an automated marketplace for users to lend and borrow various cryptocurrencies. Implemented via smart contracts, it provides a transparent way for users to earn interest or obtain loans. The protocol utilizes a dynamic interest rate model that adjusts rates based on market supply and demand. Besides offering liquidity in multiple crypto assets.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N2	Decimal loss with an empty marke	Design Logic Audit	Medium	Acknowledged
N3	Receive can lock users' native tokens	Others	Low	Acknowledged
N4	Missing the event record	Others	Suggestion	Fixed
N5	External call reminder	Unsafe External Call Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/unfamous-labs/trustin-contracts-audit

commit: 65a8a6f4a76680a1ad4a436fbc68b8aca7ec6239

Fixed Version:

https://github.com/unfamous-labs/trustin-contracts-audit

commit: e143ea465b9bbc4e951a7c5e14957bd771cec527

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

Comptroller			
Function Name	Visibility	Mutability	Modifiers

Comptroller			
hasRole	Public	-	-
<Constructor>	Public	Can Modify State	-
getAssetsIn	External	-	-
checkMembership	External	-	-
enterMarkets	Public	Can Modify State	-
addToMarketInternal	Internal	Can Modify State	-
exitMarket	External	Can Modify State	-
mintAllowed	External	Can Modify State	-
mintVerify	External	Can Modify State	-
redeemAllowed	External	Can Modify State	-
redeemAllowedInternal	Internal	-	-
redeemVerify	External	-	-
borrowAllowed	External	Can Modify State	-
borrowVerify	External	Can Modify State	-
repayBorrowAllowed	External	Can Modify State	-
repayBorrowVerify	External	Can Modify State	-
liquidateBorrowAllowed	External	-	-
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	-

Comptroller			
getAccountLiquidity	Public	-	-
getAccountLiquidityInternal	Internal	-	-
getHypotheticalAccountLiquidity	Public	-	-
getHypotheticalAccountLiquidityInternal	Internal	-	-
liquidateCalculateSeizeTokens	External	-	-
_setPriceOracle	Public	Can Modify State	-
_setCloseFactor	External	Can Modify State	-
_setCollateralFactor	External	Can Modify State	-
_setLiquidationIncentive	External	Can Modify State	-
_supportMarket	External	Can Modify State	-
_addMarketInternal	Internal	Can Modify State	-
_initializeMarket	Internal	Can Modify State	-
_setMarketBorrowCaps	External	Can Modify State	-
_setBorrowCapGuardian	External	Can Modify State	-
_setPauseGuardian	Public	Can Modify State	-
_setMintPaused	Public	Can Modify State	-
_setBorrowPaused	Public	Can Modify State	-
_setCTokenPauseState	Public	Can Modify State	-
_setTransferPaused	Public	Can Modify State	-
_setSeizePaused	Public	Can Modify State	-
_setLiquidateWhitelistEnabled	Public	Can Modify State	-
_setLiquidateWhitelist	Public	Can Modify State	-

Comptroller			
_setMintBorrowPaused	Public	Can Modify State	-
_setManagerRole	Public	Can Modify State	-
_become	Public	Can Modify State	-
adminOrInitializing	Internal	-	-
setCompSpeedInternal	Internal	Can Modify State	-
updateCompSupplyIndex	Internal	Can Modify State	-
updateCompBorrowIndex	Internal	Can Modify State	-
distributeSupplierComp	Internal	Can Modify State	-
distributeBorrowerComp	Internal	Can Modify State	-
updateContributorRewards	Public	Can Modify State	-
claimZnt	Public	Can Modify State	-
claimZnt	Public	Can Modify State	-
claimZnt	Public	Can Modify State	-
grantCompInternal	Internal	Can Modify State	-
_grantComp	Public	Can Modify State	-
_setCompSpeeds	Public	Can Modify State	-
_setContributorCompSpeed	Public	Can Modify State	-
getAllMarkets	Public	-	-
isDeprecated	Public	-	-
getBlockNumber	Public	-	-
_setCompAddress	Public	Can Modify State	-
getCompAddress	Public	-	-

CarefulMath			
Function Name	Visibility	Mutability	Modifiers
mulUInt	Internal	-	-
divUInt	Internal	-	-
subUInt	Internal	-	-
addUInt	Internal	-	-
addThenSubUInt	Internal	-	-

CDaiDelegate			
Function Name	Visibility	Mutability	Modifiers
_becomeImplementation	Public	Can Modify State	-
_becomeImplementation	Internal	Can Modify State	-
_resignImplementation	Public	Can Modify State	-
accrueInterest	Public	Can Modify State	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-
add	Internal	-	-
mul	Internal	-	-

CErc20Delegate			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_becomeImplementation	Public	Can Modify State	-

CErc20Delegate			
_resignImplementation	Public	Can Modify State	-

CErc20			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	-
mint	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	-
sweepToken	External	Can Modify State	-
_addReserves	External	Can Modify State	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-
_delegateCompLikeTo	External	Can Modify State	-

CToken			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	-
transferTokens	Internal	Can Modify State	-

CToken			
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	-	-
exchangeRateCurrent	Public	Can Modify State	nonReentrant
exchangeRateStored	Public	-	-
exchangeRateStoredInternal	Internal	-	-
getCash	External	-	-
accrueInterest	Public	Can Modify State	-
mintInternal	Internal	Can Modify State	nonReentrant
mintFresh	Internal	Can Modify State	-
redeemInternal	Internal	Can Modify State	nonReentrant

CToken			
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	-
_setPendingAdmin	External	Can Modify State	-
_acceptAdmin	External	Can Modify State	-
_setComptroller	Public	Can Modify State	-
_setReserveFactor	External	Can Modify State	nonReentrant
_setReserveFactorFresh	Internal	Can Modify State	-
_addReservesInternal	Internal	Can Modify State	nonReentrant
_addReservesFresh	Internal	Can Modify State	-
_reduceReserves	External	Can Modify State	nonReentrant
_reduceReservesFresh	Internal	Can Modify State	-
_setInterestRateModel	Public	Can Modify State	-
_setInterestRateModelFresh	Internal	Can Modify State	-

CToken			
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-

ComptrollerInterface			
Function Name	Visibility	Mutability	Modifiers
enterMarkets	External	Can Modify State	-
exitMarket	External	Can Modify State	-
mintAllowed	External	Can Modify State	-
mintVerify	External	Can Modify State	-
redeemAllowed	External	Can Modify State	-
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	-
liquidateCalculateSeizeTokens	External	-	-

ComptrollerInterface			
hasRole	External	-	-

CTokenInterface			
Function Name	Visibility	Mutability	Modifiers
transfer	External	Can Modify State	-
transferFrom	External	Can Modify State	-
approve	External	Can Modify State	-
allowance	External	-	-
balanceOf	External	-	-
balanceOfUnderlying	External	Can Modify State	-
getAccountSnapshot	External	-	-
borrowRatePerBlock	External	-	-
supplyRatePerBlock	External	-	-
totalBorrowsCurrent	External	Can Modify State	-
borrowBalanceCurrent	External	Can Modify State	-
borrowBalanceStored	External	-	-
exchangeRateCurrent	External	Can Modify State	-
exchangeRateStored	External	-	-
getCash	External	-	-
accrueInterest	External	Can Modify State	-
seize	External	Can Modify State	-
_setPendingAdmin	External	Can Modify State	-
_acceptAdmin	External	Can Modify State	-

CTokenInterface			
_setComptroller	External	Can Modify State	-
_setReserveFactor	External	Can Modify State	-
_reduceReserves	External	Can Modify State	-
_setInterestRateModel	External	Can Modify State	-

CErc20Interface			
Function Name	Visibility	Mutability	Modifiers
mint	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	-
sweepToken	External	Can Modify State	-
_addReserves	External	Can Modify State	-

CDelegatorInterface			
Function Name	Visibility	Mutability	Modifiers
_setImplementation	External	Can Modify State	-

CDelegateInterface			
Function Name	Visibility	Mutability	Modifiers
_becomeImplementation	External	Can Modify State	-

CDelegateInterface			
_resignImplementation	External	Can Modify State	-

InterestRateModel			
Function Name	Visibility	Mutability	Modifiers
getBorrowRate	External	-	-
getSupplyRate	External	-	-
getModelName	External	-	-

ComptrollerErrorReporter			
Function Name	Visibility	Mutability	Modifiers
fail	Internal	Can Modify State	-
failOpaque	Internal	Can Modify State	-

ExponentialNoError			
Function Name	Visibility	Mutability	Modifiers
truncate	Internal	-	-
mul_ScalarTruncate	Internal	-	-
mul_ScalarTruncateAddUInt	Internal	-	-
lessThanExp	Internal	-	-
lessThanOrEqualExp	Internal	-	-
greaterThanExp	Internal	-	-
isZeroExp	Internal	-	-
safe224	Internal	-	-
safe32	Internal	-	-

ExponentialNoError			
add_	Internal	-	-
add_	Internal	-	-
add_	Internal	-	-
sub_	Internal	-	-
sub_	Internal	-	-
sub_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
mul_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
div_	Internal	-	-
fraction	Internal	-	-

CErc20Delegator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_setImplementation	Public	Can Modify State	-
mint	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	-
transfer	External	Can Modify State	-
transferFrom	External	Can Modify State	-
approve	External	Can Modify State	-
allowance	External	-	-
balanceOf	External	-	-
balanceOfUnderlying	External	Can Modify State	-
getAccountSnapshot	External	-	-
borrowRatePerBlock	External	-	-
supplyRatePerBlock	External	-	-
totalBorrowsCurrent	External	Can Modify State	-
borrowBalanceCurrent	External	Can Modify State	-
borrowBalanceStored	Public	-	-

CErc20Delegator			
exchangeRateCurrent	Public	Can Modify State	-
exchangeRateStored	Public	-	-
getCash	External	-	-
accrueInterest	Public	Can Modify State	-
seize	External	Can Modify State	-
sweepToken	External	Can Modify State	-
_setPendingAdmin	External	Can Modify State	-
_setComptroller	Public	Can Modify State	-
_setReserveFactor	External	Can Modify State	-
_acceptAdmin	External	Can Modify State	-
_addReserves	External	Can Modify State	-
_reduceReserves	External	Can Modify State	-
_setInterestRateModel	Public	Can Modify State	-
delegateTo	Internal	Can Modify State	-
delegateToImplementation	Public	Can Modify State	-
delegateToViewImplementation	Public	-	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-

CErc20Immutable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

CEther			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
mint	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	-
_addReserves	External	Payable	-
<Receive Ether>	External	Payable	-
getCashPrior	Internal	-	-
doTransferIn	Internal	Can Modify State	-
doTransferOut	Internal	Can Modify State	-

PriceOracle			
Function Name	Visibility	Mutability	Modifiers
getUnderlyingPrice	External	-	-
assetPrices	External	-	-

Unitroller			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

Unitroller			
_setPendingImplementation	Public	Can Modify State	-
_acceptImplementation	Public	Can Modify State	-
_setPendingAdmin	Public	Can Modify State	-
_acceptAdmin	Public	Can Modify State	-
_setAdminDirect	Public	Can Modify State	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-

DAIInterestRateModelV3			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	JumpRateModelV2
updateJumpRateModel	External	Can Modify State	-
getSupplyRate	Public	-	-
dsrPerBlock	Public	-	-
poke	Public	Can Modify State	-

JumpRateModelV2			
Function Name	Visibility	Mutability	Modifiers
getBorrowRate	External	-	-
<Constructor>	Public	Can Modify State	BaseJumpRateModelV2
getModelName	External	-	-

BaseJumpRateModelV2			
Function Name	Visibility	Mutability	Modifiers

BaseJumpRateModelV2			
<Constructor>	Public	Can Modify State	-
updateJumpRateModel	External	Can Modify State	-
utilizationRate	Public	-	-
getBorrowRateInternal	Internal	-	-
getSupplyRate	Public	-	-
updateJumpRateModelInternal	Internal	Can Modify State	-

JumpRateModel			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
utilizationRate	Public	-	-
getBorrowRate	Public	-	-
getSupplyRate	Public	-	-
getModelName	External	-	-

LegacyInterestRateModel			
Function Name	Visibility	Mutability	Modifiers
getBorrowRate	External	-	-
getSupplyRate	External	-	-
getModelName	External	-	-

LegacyJumpRateModelV2			
Function Name	Visibility	Mutability	Modifiers
getBorrowRate	External	-	-

LegacyJumpRateModelV2			
<Constructor>	Public	Can Modify State	BaseJumpRateModelV2
getModelName	External	-	-

Maximillion			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
repayBehalf	Public	Payable	-
repayBehalfExplicit	Public	Payable	-

Reservoir			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
drip	Public	Can Modify State	-
add	Internal	-	-
sub	Internal	-	-
mul	Internal	-	-
min	Internal	-	-

SimplePriceOracle			
Function Name	Visibility	Mutability	Modifiers
_getUnderlyingAddress	Private	-	-
getUnderlyingPrice	Public	-	-
setUnderlyingPrice	Public	Can Modify State	-
setDirectPrice	Public	Can Modify State	-

SimplePriceOracle			
assetPrices	External	-	-
compareStrings	Internal	-	-

Timelock			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
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	-	-

WhitePaperInterestRateModel			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
utilizationRate	Public	-	-
getBorrowRate	Public	-	-
getSupplyRate	Public	-	-
getModelName	External	-	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

In the Comptroller, CToken, SimplePriceOracle, and Unitroller contracts, the admin role can modify key sensitive parameters such as the manager roles, the rate model, the market, the pause status, the whitelist, the price of the underlying asset, and the admin role, which will lead to the risk of over-privilege of the admin role.

Code location:

Comptroller.sol#847-946, 1018-1150, 1407-1450, 1478-1481

CToken.sol#864-971, 1046-1133

Unitroller.sol#39-141

SimplePriceOracle.sol#25-34

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. The authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N2] [Medium] Decimal loss with an empty marke

Category: Design Logic Audit

Content

If there are two markets, one of which was used by the UI, one of which was empty. Someone can mint collateral tokens in an empty market and redeem most minted tokens, then donate redeemed asset tokens to inflate the exchange rate through the getAccountSnapshot function. Next, borrow a different asset with the manipulated exchange rate, and redeem collateral to recover donation. However, the redeemUnderlying function may wrongly be

rounded down on the tokens to remove from a malicious caller, which causes the redemption of many tokens only to require little underlying assets. The last, liquidation borrower contract position with borrowed funds and redeem collateral tokens to reset the empty market.

Reference:

https://www.comp.xyz/t/hundred-finance-exploit-and-compound-v2/4266

Code location:

contracts/CToken.sol#187-194,294-313,504

```
    function getAccountSnapshot(address account) override external view returns
(uint, uint, uint, uint) {
        return (
            NO_ERROR,
            accountTokens[account],
            borrowBalanceStoredInternal(account),
            exchangeRateStoredInternal()
        );
    }
    function exchangeRateStoredInternal() virtual internal view returns (uint) {
        uint _totalSupply = totalSupply;
        if (_totalSupply == 0) {
            /*
             * If there are no tokens minted:
             *  exchangeRate = initialExchangeRate
             */
            return initialExchangeRateMantissa;
        } else {
            /*
             * Otherwise:
             *  exchangeRate = (totalCash + totalBorrows - totalReserves) /
totalSupply
             */
            uint totalCash = getCashPrior();
            uint cashPlusBorrowsMinusReserves = totalCash + totalBorrows -
totalReserves;
            uint exchangeRate = cashPlusBorrowsMinusReserves * expScale /
_totalSupply;

            return exchangeRate;
        }
    }

    function redeemFresh(address payable redeemer, uint redeemTokensIn, uint
redeemAmountIn) internal {
```

```
        ...
        if (redeemTokensIn > 0) {
            /*
             * We calculate the exchange rate and the amount of underlying to be
redeemed:
             *  redeemTokens = redeemTokensIn
             *  redeemAmount = redeemTokensIn x exchangeRateCurrent
             */
            redeemTokens = redeemTokensIn;
            redeemAmount = mul_ScalarTruncate(exchangeRate, redeemTokensIn);
        } else {
            /*
             * We get the current exchange rate and calculate the amount to be
redeemed:
             *  redeemTokens = redeemAmountIn / exchangeRate
             *  redeemAmount = redeemAmountIn
             */
            redeemTokens = div_(redeemAmountIn, exchangeRate);
            redeemAmount = redeemAmountIn;
        }
```

Solution

The protocol could mint minimal balances to prevent an empty market scenario, making such attacks unfeasible. Or follow the steps in the reference.

Status

Acknowledged

[N3] [Low] Receive can lock users' native tokens

Category: Others

Content

There is a receive function in the CErc20Delegator, Timelock, and Unitroller contracts so that the contracts can receive native tokens. However, the receive function can lock users' native tokens when users transfer the native token in these contracts by mistake and there is no token processing logic.

Code location:

CErc20Delegator.sol#478

Timelock.sol#33

Unitroller.sol#162

```
receive() external payable {}
```

Solution

It's recommended to remove the receive() function if the contract only needs to receive ether from functions in the contract, and use the payable modifier in these functions instead.

Status

Acknowledged

[N4] [Suggestion] Missing the event record

Category: Others

Content

The admin role can modify the compAddress parameter, but there are no event logs in these functions.

Code location:

Comptroller.sol#1478-1481

```
function _setCompAddress(address compAddr) public {
    require(msg.sender == admin, "only admin allowed");
    compAddress = compAddr;
}
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Fixed

[N5] [Suggestion] External call reminder

Category: Unsafe External Call Audit

Content

In the Comptroller contract, users can call the claimZnt to claim the comp in markets, but the implementation address is seting by the admin and the import Zenith is not in the audit scope.

Code location:

Comptroller.sol#1335-1397

```
import "./znt/Zenith.sol";
    function claimZnt(address[] memory holders, CToken[] memory cTokens, bool
borrowers, bool suppliers) public {
        ...
        for (uint j = 0; j < holders.length; j++) {
            compAccrued[holders[j]] = grantCompInternal(holders[j],
compAccrued[holders[j]]);
        }
    }


    function grantCompInternal(address user, uint amount) internal returns (uint) {
        Zenith comp = Zenith(getCompAddress());
        uint compRemaining = comp.balanceOf(address(this));
        if (amount > 0 && amount <= compRemaining) {
            comp.transfer(user, amount);

            emit ClaimReward(user, amount);
            return 0;
        }
        return amount;
    }
```

Solution

It is recommended to clarify whether this external call contract is credible and check the validity of the incoming resolver address and data.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002404100003	SlowMist Security Team	2024.04.08 - 2024.04.10	Medium Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 medium risks, 1 low risk, and 2 suggestions. The code was not deployed to the mainnet.

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 based 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 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