# Smart Contract
# Security Audit Report

# Table Of Contents

# 1 Executive Summary

On 2023.04.21, the SlowMist security team received the Davos Money team's security audit application for Davos Money, 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

Davos is a first-rate cross-chain lending and borrowing platform that offers its users the opportunity to borrow the DUSD Stable Asset, initially using their staked MATIC assets as collateral.

## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | Potential out of gas in DGTReward claiming | Denial of Service Vulnerability | Low | Acknowledged |

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N2 | setSwapPool can be called by malicious actor if not called during deployment | Race Conditions Vulnerability | Low | Ignored |
| N3 | Risk of excessive authority | Authority Control Vulnerability Audit | Low | Acknowledged |
| N4 | Fixed decimal used in fetching oracle price | Unsafe External Call Audit | Medium | Confirmed |
| N5 | Unchecked transfer | Unsafe External Call Audit | Low | Fixed |
| N6 | Unclear logic in SwapPool | Design Logic Audit | Suggestion | Fixed |
| N7 | Fixed price used in DgtOracle | Design Logic Audit | Low | Ignored |
| N8 | Missing zero address check | Others | Suggestion | Fixed |
| N9 | msg.value inside loop | Denial of Service Vulnerability | Low | Fixed |
| N10 | Unprotected upgrade function | Authority Control Vulnerability Audit | Critical | Acknowledged |
| N11 | prefer parsePriceFeedUpdates to updatePriceFeeds | Others | Suggestion | Acknowledged |
| N12 | Use multisig wallet for feeReceiver | Others | Suggestion | Confirmed |
| N13 | Inflation attack in addLiquidity in SwapPool | Design Logic Audit | Critical | Fixed |

# 4 Code Overview

## 4.1 Contracts Description

**Audit version:**

https://github.com/davos-money/davos-contracts/tree/main

commit: e00e3afa169cef8ae2f6bc0b6a206b87698f36a4

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:

| DGTRewards | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| initPool | external | Can Modify State | auth |
| setDgtToken | external | Can Modify State | auth |
| setRewardsMaxLimit | external | Can Modify State | auth |
| setOracle | external | Can Modify State | auth |
| setRate | external | Can Modify State | auth |
| setMaxPools | external | Can Modify State | auth |
| dgtPrice | public | view | - |
| rewardsRate | public | view | - |
| distributionApy | public | view | - |
| pendingRewards | public | view | - |
| claimable | public | view | poolInit |

| DGTRewards | | | |
|---|---|---|---|
| unrealisedRewards | public | view | poolInit |
| drop | public | Can Modify State | - |
| claim | external | Can Modify State | - |
| cage | public | Can Modify State | auth |
| uncage | public | Can Modify State | auth |

| DGTToken | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| mint | external | Can Modify State | auth |
| burn | external | Can Modify State | - |
| pause | external | Can Modify State | auth |
| unpause | external | Can Modify State | auth |

| Interaction | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| enableWhitelist | external | Can Modify State | auth |
| disableWhitelist | external | Can Modify State | auth |
| setWhitelistOperator | external | Can Modify State | auth |
| addToWhitelist | external | Can Modify State | operatorOrWard |

| Interaction | | | |
|---|---|---|---|
| | | | |
| removeFromWhitelist | external | Can Modify State | operatorOrWard |
| initialize | external | Can Modify State | initializer |
| setCores | public | Can Modify State | auth |
| setDavosApprove | public | Can Modify State | auth |
| setCollateralType | external | Can Modify State | auth |
| setCollateralDuty | external | Can Modify State | auth |
| setDavosProvider | external | Can Modify State | auth |
| removeCollateralType | external | Can Modify State | auth |
| stringToBytes32 | external | pure | - |
| deposit | external | Can Modify State | whitelisted |
| borrow | external | Can Modify State | - |
| dropRewards | public | Can Modify State | - |
| payback | external | Can Modify State | - |
| withdraw | external | Can Modify State | - |
| drip | public | Can Modify State | - |
| poke | public | Can Modify State | - |
| setRewards | external | Can Modify State | auth |
| collateralPrice | public | view | - |
| davosPrice | external | view | - |
| collateralRate | external | view | - |
| depositTVL | external | view | - |
| collateralTVL | external | view | - |

| Interaction | | | |
|---|---|---|---|
| free | public | view | - |
| locked | public | view | - |
| borrowed | external | view | - |
| availableToBorrow | external | view | - |
| willBorrow | external | view | - |
| currentLiquidationPrice | external | view | - |
| estimatedLiquidationPrice | external | view | - |
| estimatedLiquidationPriceDAVOS | external | view | - |
| borrowApr | public | view | - |
| startAuction | external | Can Modify State | - |
| buyFromAuction | external | Can Modify State | - |
| getAuctionStatus | external | view | - |
| upchostClipper | external | Can Modify State | - |
| getAllActiveAuctionsForToken | external | view | - |
| resetAuction | external | Can Modify State | - |
| totalPegLiquidity | external | view | - |

| LinearDecrease | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |

| LinearDecrease | | | |
|---|---|---|---|
| price | external | view | - |

| StairstepExponentialDecrease | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |
| price | external | view | - |

| ExponentialDecrease | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |
| price | external | view | - |

| Clipper | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| sales | external | view | - |

| Clipper | | | |
|---|---|---|---|
| file | external | Can Modify State | auth; lock |
| file | external | Can Modify State | auth; lock |
| kick | external | Can Modify State | auth; lock; isStopped |
| redo | external | Can Modify State | auth; lock; isStopped |
| take | external | Can Modify State | auth; lock; isStopped |
| count | external | view | - |
| list | external | view | - |
| getStatus | external | view | - |
| upchost | external | Can Modify State | - |
| yank | external | Can Modify State | auth; lock |

| Davos | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| transfer | external | Can Modify State | - |
| transferFrom | public | Can Modify State | - |
| mint | external | Can Modify State | auth |
| burn | external | Can Modify State | - |
| approve | external | Can Modify State | - |
| push | external | Can Modify State | - |
| pull | external | Can Modify State | - |

| Davos | | | |
|---|---|---|---|
| move | external | Can Modify State | - |
| permit | external | Can Modify State | - |
| increaseAllowance | public | Can Modify State | - |
| decreaseAllowance | public | Can Modify State | - |
| setSupplyCap | public | Can Modify State | auth |
| updateDomainSeparator | external | Can Modify State | auth |

| Dog | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| chop | external | view | - |
| bark | external | Can Modify State | auth |
| digs | external | Can Modify State | auth |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |

| Jar | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |

| Jar | | | |
|---|---|---|---|
| | | | |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| lastTimeRewardApplicable | public | view | - |
| tokensPerShare | public | view | - |
| earned | public | view | - |
| replenish | external | Can Modify State | authOrOperator; update |
| setSpread | external | Can Modify State | authOrOperator |
| setExitDelay | external | Can Modify State | authOrOperator |
| addOperator | external | Can Modify State | auth |
| removeOperator | external | Can Modify State | auth |
| extractDust | external | Can Modify State | auth |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |
| join | external | Can Modify State | update; nonReentrant |
| exit | external | Can Modify State | update; nonReentrant |
| redeemBatch | external | Can Modify State | nonReentrant |

| GemJoin | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |

| GemJoin | | | |
|---|---|---|---|
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |
| join | external | Can Modify State | auth |
| exit | external | Can Modify State | auth |

| DavosJoin | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |
| join | external | Can Modify State | auth |
| exit | external | Can Modify State | auth |

| Jug | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| init | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |

| Jug | | | |
|---|---|---|---|
| file | external | Can Modify State | auth |
| drip | external | Can Modify State | - |

| Spotter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| poke | external | Can Modify State | - |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |

| Vat | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| behalf | external | Can Modify State | auth |
| regard | external | Can Modify State | auth |
| hope | external | Can Modify State | - |
| nope | external | Can Modify State | - |
| initialize | external | Can Modify State | initializer |

| Vat | | | |
|---|---|---|---|
| | | | |
| init | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |
| slip | external | Can Modify State | auth |
| flux | external | Can Modify State | auth |
| move | external | Can Modify State | auth |
| frob | external | Can Modify State | auth |
| fork | external | Can Modify State | auth |
| grab | external | Can Modify State | auth |
| heal | external | Can Modify State | - |
| suck | external | Can Modify State | auth |
| fold | external | Can Modify State | auth |

| Vow | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| rely | external | Can Modify State | auth |
| deny | external | Can Modify State | auth |
| initialize | external | Can Modify State | initializer |
| file | external | Can Modify State | auth |
| file | external | Can Modify State | auth |
| heal | external | Can Modify State | - |

| Vow | | | |
|---|---|---|---|
| feed | external | Can Modify State | - |
| flap | external | Can Modify State | - |
| cage | external | Can Modify State | auth |
| uncage | external | Can Modify State | auth |

| AuctionProxy | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| startAuction | public | Can Modify State | - |
| resetAuction | public | Can Modify State | - |
| buyFromAuction | public | Can Modify State | - |
| getAllActiveAuctionsForClip | external | view | - |

| BaseStrategy | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| setStrategist | external | Can Modify State | onlyOwner |
| setFeeRecipient | external | Can Modify State | onlyOwner |
| pause | external | Can Modify State | onlyOwnerOrStrategist |
| unpause | external | Can Modify State | onlyOwnerOrStrategist |
| balanceOfWant | public | view | - |
| balanceOfPool | public | view | - |
| balanceOf | public | view | - |

| CeToken | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |

| CeToken | | | |
|---|---|---|---|
| initialize | external | Can Modify State | initializer |
| burn | external | Can Modify State | onlyMinter |
| mint | external | Can Modify State | onlyMinter |
| changeVault | external | Can Modify State | onlyOwner |
| getVaultAddress | external | view | - |

| CeVault | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| deposit | external | Can Modify State | nonReentrant |
| depositFor | external | Can Modify State | nonReentrant; onlyRouter |
| claimYieldsFor | external | Can Modify State | onlyRouter; nonReentrant |
| claimYields | external | Can Modify State | nonReentrant |
| withdraw | external | Can Modify State | nonReentrant |
| withdrawFor | external | Can Modify State | nonReentrant; onlyRouter |
| getTotalAmountInVault | external | view | - |
| getPrincipalOf | external | view | - |
| getYieldFor | external | view | - |
| getCeTokenBalanceOf | external | view | - |
| getDepositOf | external | view | - |
| getClaimedOf | external | view | - |
| changeRouter | external | Can Modify State | onlyOwner |
| getName | external | view | - |

| CeVault | | | |
|---|---|---|---|
| getCeToken | external | view | - |
| getAmaticcAddress | external | view | - |
| getRouter | external | view | - |

| CerosRouterLs | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| deposit | external | Can Modify State | nonReentrant; whenNotPaused |
| withdrawAMATICc | external | Can Modify State | nonReentrant; whenNotPaused |
| claim | external | Can Modify State | nonReentrant; whenNotPaused |
| claimProfit | external | Can Modify State | nonReentrant |
| withdrawFor | external | payable | nonReentrant; whenNotPaused; onlyOwnerOrStrategy |
| pause | external | Can Modify State | onlyOwner |
| unpause | external | Can Modify State | onlyOwner |
| changePriceGetter | external | Can Modify State | onlyOwner |
| changePairFee | external | Can Modify State | onlyOwner |
| changeStrategy | external | Can Modify State | onlyOwner |
| changePool | external | Can Modify State | onlyOwner |
| changeDex | external | Can Modify State | onlyOwner |
| changeCeVault | external | Can Modify State | onlyOwner |

| CerosRouterLs | | | |
|---|---|---|---|
| getAmountOut | public | view | - |
| getPendingWithdrawalOf | external | view | - |
| getYieldFor | external | view | - |

| CerosRouterLsEth | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| deposit | external | Can Modify State | nonReentrant; whenNotPaused |
| withdrawAMATICc | external | Can Modify State | nonReentrant; whenNotPaused |
| claim | external | Can Modify State | nonReentrant; whenNotPaused |
| claimProfit | external | Can Modify State | nonReentrant |
| withdrawFor | external | payable | nonReentrant; whenNotPaused; onlyOwnerOrStrategy |
| pause | external | Can Modify State | onlyOwner |
| unpause | external | Can Modify State | onlyOwner |
| changePriceGetter | external | Can Modify State | onlyOwner |
| changePairFee | external | Can Modify State | onlyOwner |
| changeStrategy | external | Can Modify State | onlyOwner |
| changePool | external | Can Modify State | onlyOwner |
| changeDex | external | Can Modify State | onlyOwner |

| CerosRouterLsEth | | | |
|---|---|---|---|
| changeCeVault | external | Can Modify State | onlyOwner |
| getAmountOut | public | view | - |
| getPendingWithdrawalOf | external | view | - |
| getYieldFor | external | view | - |

| CerosRouterSp | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | public | Can Modify State | initializer |
| deposit | external | payable | nonReentrant |
| depositWMatic | external | Can Modify State | nonReentrant |
| claim | external | Can Modify State | nonReentrant |
| claimProfit | external | Can Modify State | nonReentrant |
| getAmountOut | public | view | - |
| withdrawWithSlippage | external | Can Modify State | nonReentrant |
| getProfitFor | external | view | - |
| getYieldFor | external | view | - |
| changeVault | external | Can Modify State | onlyOwner |
| changeDex | external | Can Modify State | onlyOwner |
| changeSwapPool | external | Can Modify State | onlyOwner |
| changeProvider | external | Can Modify State | onlyOwner |
| changePairFee | external | Can Modify State | onlyOwner |
| changePriceGetter | external | Can Modify State | onlyOwner |
| getSwapPool | external | view | - |

| CerosRouterSp | | | |
|---|---|---|---|
| | | | |
| getCeToken | external | view | - |
| getWMaticAddress | external | view | - |
| getCertToken | external | view | - |
| getPoolAddress | external | view | - |
| getDexAddress | external | view | - |
| getVaultAddress | external | view | - |

| DavosProvider | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| provide | external | payable | whenNotPaused; nonReentrant |
| release | external | payable | whenNotPaused; nonReentrant |
| liquidation | external | Can Modify State | onlyOwnerOrInteraction; nonReentrant |
| daoBurn | external | Can Modify State | onlyOwnerOrInteraction; nonReentrant |
| daoMint | external | Can Modify State | onlyOwnerOrInteraction; nonReentrant |
| pause | external | Can Modify State | onlyOwner |
| unPause | external | Can Modify State | onlyOwner |
| changeCollateral | external | Can Modify State | onlyOwner |
| changeCollateralDerivative | external | Can Modify State | onlyOwner |
| changeMasterVault | external | Can Modify State | onlyOwner |

| DavosProvider | | | |
|---|---|---|---|
| changeInteraction | external | Can Modify State | onlyOwner |
| changeUnderlying | external | Can Modify State | onlyOwner |
| changeNativeStatus | external | Can Modify State | onlyOwner |

| NonTransferableERC20 | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| name | public | view | - |
| symbol | public | view | - |
| decimals | public | view | - |
| totalSupply | public | view | - |
| balanceOf | public | view | - |
| transfer | public | Can Modify State | - |
| allowance | public | view | - |
| approve | public | Can Modify State | - |
| transferFrom | public | Can Modify State | - |

| dMATIC | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| burn | external | Can Modify State | onlyMinter |
| mint | external | Can Modify State | onlyMinter |
| changeMinter | external | Can Modify State | onlyOwner |
| getMinter | external | view | - |

| LP | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | external | Can Modify State | initializer |
| setSwapPool | external | Can Modify State | - |
| mint | external | Can Modify State | onlySwapPool |
| burn | external | Can Modify State | onlySwapPool |

| SwapPool | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| initialize | public | Can Modify State | initializer |
| addLiquidityEth | external | payable | onlyProvider; nonReentrant |
| addLiquidity | external | Can Modify State | onlyProvider; nonReentrant |
| swapEth | external | payable | onlyIntegrator; nonReentrant |
| swap | external | Can Modify State | onlyIntegrator; nonReentrant |
| getAmountOut | external | view | - |
| getAmountIn | external | view | - |
| withdrawOwnerFeeEth | external | Can Modify State | onlyOwner; nonReentrant |
| withdrawOwnerFee | external | Can Modify State | onlyOwner; nonReentrant |
| getRemainingManagerFee | external | view | - |
| withdrawManagerFee | external | Can Modify State | onlyManager; nonReentrant |
| withdrawManagerFeeEth | external | Can Modify State | onlyManager; nonReentrant |
| add | public | Can Modify State | - |

| SwapPool | | | |
|---|---|---|---|
| setFee | external | Can Modify State | onlyOwnerOrManager |
| setThreshold | external | Can Modify State | onlyManager |
| setMaticPool | external | Can Modify State | onlyOwner |
| enableIntegratorLock | external | Can Modify State | onlyOwnerOrManager |
| enableProviderLock | external | Can Modify State | onlyOwnerOrManager |
| excludeFromFee | external | Can Modify State | onlyOwnerOrManager |
| triggerRebalanceAnkr | external | Can Modify State | nonReentrant; onlyManager |
| triggerRebalanceAnkrWithPercent | external | Can Modify State | nonReentrant; onlyManager |
| approveToMaticPool | external | Can Modify State | - |
| skim | public | Can Modify State | - |
| remove | public | Can Modify State | nonReentrant |
| contains | external | view | - |
| length | external | view | - |
| at | external | view | - |
|  | external | payable | - |

## 4.3 Vulnerability Summary

### [N1] [Low] Potential out of gas in DGTReward claiming

Category: Denial of Service Vulnerability

Content

`claim` function in `DGTReward` loop through the `poolsList`. In some extreme case, when `poolsList` is big enough, the user may suffer from lock of fund due to out of gas. Note that `pendingRewards` also loop through `poolsList`

- contracts/DGTRewards.sol

```
function claim(uint256 amount) external {
    require(amount <= pendingRewards(msg.sender), "Rewards/not-enough-rewards");
    require(poolLimit >= amount, "Rewards/rewards-limit-exceeded");
    uint256 i = 0;
    while (i < poolsList.length) { // potential OOG
        drop(poolsList[i], msg.sender);
        i++;
    }
    claimedRewards[msg.sender] += amount;
    poolLimit -= amount;

    IERC20Upgradeable(dgtToken).safeTransfer(msg.sender, amount);
    emit Claimed(msg.sender, amount);
}
```

Solution

It is recommended to have `claim(uint256 pool, uint256 amount)` which only claim `poolsList[pool]`

Status

Acknowledged; Team: The DGTRewards.sol and related DGTContracts are work in progress. The current out of gas error will be true if there were many pools. But we will only have one pool for now.

### [N2] [Low] setSwapPool can be called by malicious actor if not called during deployment

Category: Race Conditions Vulnerability

Content

`setSwapPool` in `LP` is not called in initializer. This expose vulnerability when the `swapPool` is not set due to misconfiguration.

- contracts/swapPool/LP.sol

```
function setSwapPool(address _swapPool) external {
  require(swapPool == address(0), "swap pool can be set only once");
```

```
    swapPool = _swapPool;
  }
```

Solution

Either initialize swapPool in the initializer or set up a owner role to set this variable.

Status

Ignored; It was by design and is taken care of in the scripts. SwapPool has already been deployed. Setting swap pool can't be shifted to initializer because SwapPool also requires the address of the LP token.

### [N3] [Low] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

Table below list all critical function can be called by EOA.

- CerosRouterLs | Owner | withdrawFor, changePriceGetter, changePairFee, changeStrategy, changePool, changeDex, changeCeVault
- CerosRouterLsEth | Owner | withdrawFor, changePriceGetter, changePairFee, changeStrategy, changePool, changeDex, changeCeVault
- CerosRouterSp | Owner | changeVault, changeDex, changeSwapPool, changeProvider, changePairFee, changePriceGetter
- CeToken | Owner | changeVault
- CeVault | Owner | changeRouter
- DavosProvider | Owner | daoMint, daoBurn, liquidation, changeCollateral, changeCollateralDerivative, changeMasterVault, changeInteraction, changeUnderlying, changeNativeStatus
- dMATIC | Owner | changeMinter
- dMATIC | Minter | mint, burn
- MasterVault | Provider | depositUnderlying, withdrawUnderlying
- MasterVault | Manager | withdrawFromStrategy, withdrawAllFromStrategy, retireStrat, migrateStrategy

- MasterVault | Owner | withdrawUnderlying, withdrawFromStrategy, withdrawAllFromStrategy, retireStrat, migrateStrategy, addStrategy, changeStrategyAllocation, changeAllocateOnDeposit, setDepositFee, setWithdrawalFee, changeProvider, setWaitingPool,
- CerosYieldConverterStrategyLs | Owner | changeDestination
- CerosYieldConverterStrategySp | Owner | changeSwapPool, changeCeRouter
- SwapPool | Owner | setMaticPool

For example. In `DGTRewards`, owner can set the `oracle` to be any address. In turns, `oracle` can be used to manipulate the claimable rewards.

- contracts/DGTRewards.sol

```
function setOracle(address oracle_) external auth {
    require(oracle_ != address(0), "Reward/invalid-oracle");
    oracle = PipLike(oracle_);authority

    emit DgtOracleChanged(address(oracle));
}
```

Solution

It is recommended to use multisig, timelock and delegate the governance to DAO for this.

Status

Acknowledged; Multisig contracts will be assigned for all these contracts. For all those other contracts where an address has been initialized or to be initialized (like setSwapPool or setOracle) by another function (which requires the address to be 0) will be taken care of in the scripts. And for all those accesses where an access is given to another contract is by design (wards of MakerDAO).

### [N4] [Medium] Fixed decimal used in fetching oracle price

Category: Unsafe External Call Audit

Content

`peek` in `MaticOraclev2.sol` assume the decimal of the price returned by chainlink oracle is always 8, which is true for `MATIC/usd`. But it is more safe to fetch this decimal when initialize the contract.

- contracts/oracle/MaticOraclev2.sol

```
function peek() public view returns (bytes32, bool) {
    // Chainlink
    (
        /*uint80 roundID*/,
        int256 price,
        /*uint startedAt*/,
        uint256 timeStamp,
        /*uint80 answeredInRound*/
    ) = priceFeed.latestRoundData();

    if (block.timestamp - timeStamp <= threshold && price >= 0) {
        return (bytes32(uint256(price) * (10**10)), true); // @audit should not use
fixed decimal
    }

    // Pyth
    PythStructs.Price memory pythPrice = pyth.getPrice(pythPriceID);

    if (pythPrice.price < 0) {
        return (0, false);
    }
    return (bytes32(uint64(pythPrice.price) * (10**uint32(pythPrice.expo + 18))),
true);
}
```

Solution

Initialize the price decimal in intializer instead.

Status

Confirmed; Team will consider this in the near future for a more generic oracle.

## [N5] [Low] Unchecked transfer

Category: Unsafe External Call Audit

Content

The return value of an external transfer/transferFrom call is not checked

```
_aMATICc.transferFrom(msg.sender,address(this),amount)
(contracts/ceros/CeVault.sol#79)
_aMATICc.transfer(recipient,availableYields) (contracts/ceros/CeVault.sol#134)
_aMATICc.transfer(recipient,realAmount) (contracts/ceros/CeVault.sol#173)
```

```
s_maticToken.transferFrom(msg.sender,address(this),_amount)
(contracts/ceros/CerosRouterLs.sol#77)
s_aMATICc.transfer(_recipient,profit) (contracts/ceros/CerosRouterLs.sol#147)
s_maticToken.transferFrom(msg.sender,address(this),_amount)
(contracts/ceros/CerosRouterLsEth.sol#77)
s_aMATICc.transfer(_recipient,profit) (contracts/ceros/CerosRouterLsEth.sol#147)
IERC20(_wMaticAddress).transferFrom(msg.sender,address(this),amount)
(contracts/ceros/CerosRouterSp.sol#87)
_certToken.transfer(recipient,profit) (contracts/ceros/CerosRouterSp.sol#141)
davos.transfer(keeper,davosBal) (contracts/libraries/AuctionProxy.sol#39)
davos.transfer(keeper,davosBal) (contracts/libraries/AuctionProxy.sol#62)
davos.transferFrom(msg.sender,address(this),davosMaxAmount)
(contracts/libraries/AuctionProxy.sol#83)
davos.transfer(receiverAddress,davosBal) (contracts/libraries/AuctionProxy.sol#99)
davos.transfer(receiverAddress,davosBal) (contracts/libraries/AuctionProxy.sol#99)
IERC20Upgradeable(davos).transferFrom(msg.sender,address(this),wad)
(contracts/vow.sol#99)
```

Solution

use safeTransfer and safeTransferFrom instead

Status

Fixed; Team:

aMATICc is a trusted contract from our side.

DAVOS is a trusted contract from our side.

_certToken is a trusted contract from our side.

s_maticToken now uses safeTransferFrom.

_wMaticAddress now uses safeTransferFrom.

### [N6] [Suggestion] Unclear logic in SwapPool

Category: Design Logic Audit

Content

When first minting LP for liquidity provider, only `2 * amount0 / 1e8` was mint.

```
function _addLiquidity(
  uint256 amount0,
  uint256 amount1,
  bool useEth
) internal virtual {
  uint256 ratio = ICerosToken(cerosToken).ratio();
```

```
    uint256 value = (amount0 * ratio) / 1e18;
    if (amount1 < value) {
      amount0 = (amount1 * 1e18) / ratio;
    } else {
      amount1 = value;
    }
    if (useEth) {
      INativeERC20(nativeToken).deposit{ value: amount0 }();
      uint256 diff = msg.value - amount0;
      if (diff != 0) {
        _sendValue(msg.sender, diff);
      }
    } else {
      IERC20Upgradeable(nativeToken).safeTransferFrom(msg.sender, address(this),
amount0);
    }
    IERC20Upgradeable(cerosToken).safeTransferFrom(msg.sender, address(this), amount1);
    if (nativeTokenAmount == 0 && cerosTokenAmount == 0) {
      require(amount0 > 1e18, "cannot add first time less than 1 token");
      nativeTokenAmount = amount0;
      cerosTokenAmount = amount1;

      ILP(lpToken).mint(msg.sender, (2 * amount0) / 10**8);
    } else {
      uint256 allInNative = nativeTokenAmount + (cerosTokenAmount * 1e18) / ratio;
      uint256 mintAmount = (2 * amount0 * ILP(lpToken).totalSupply()) / allInNative;
      nativeTokenAmount += amount0;
      cerosTokenAmount += amount1;

      ILP(lpToken).mint(msg.sender, mintAmount);
    }
    emit LiquidityChange(msg.sender, amount0, amount1, nativeTokenAmount,
cerosTokenAmount, true);
  }
```

Solution

It seems like a lock of minimal liquidity. Waiting for discussion with team.

Status

Fixed; change 1e8 to 1e18

## [N7] [Low] Fixed price used in DgtOracle

Category: Design Logic Audit

Content

`contracts/oracle/DgtOracle.sol`

```
function changePriceToken(uint256 _price) external onlyOwner {

    price = _price;
    emit PriceChanged(price);
}
```

Solution

Waiting for disscusion with team.

Status

Ignored; Team: This is intentional since the price of DGT Token is not tied to liquidity at the start.

### [N8] [Suggestion] Missing zero address check

Category: Others

Content

```
rewards = rewards_ (contracts/DGTToken.sol#36)
whitelistOperator = usr (contracts/Interaction.sol#62)
dog = dog_ (contracts/Interaction.sol#107)
s_strategy = _strategy (contracts/ceros/CerosRouterLs.sol#203)
s_strategy = _strategy (contracts/ceros/CerosRouterLsEth.sol#202)
_wMaticAddress = wMaticToken (contracts/ceros/CerosRouterSp.sol#54)
vow = data (contracts/clip.sol#158)
vow = data (contracts/dog.sol#113)
DAVOS = _davosToken (contracts/jar.sol#98)
vow = data (contracts/jug.sol#108)
underlying = _underlyingToken (contracts/masterVault/WaitingPool.sol#60)
rewards = rewards_ (contracts/DGTToken.sol#36)
whitelistOperator = usr (contracts/Interaction.sol#62)
dog = dog_ (contracts/Interaction.sol#107)
s_strategy = _strategy (contracts/ceros/CerosRouterLs.sol#203)
s_strategy = _strategy (contracts/ceros/CerosRouterLsEth.sol#202)
_wMaticAddress = wMaticToken (contracts/ceros/CerosRouterSp.sol#54)
vow = data (contracts/clip.sol#158)
vow = data (contracts/dog.sol#113)
DAVOS = _davosToken (contracts/jar.sol#98)
vow = data (contracts/jug.sol#108)
underlying = _underlyingToken (contracts/masterVault/WaitingPool.sol#60)
tokenIn = _tokenIn (contracts/oracle/PriceOracle.sol#27)
tokenIn = _tokenIn (contracts/oracle/PriceOracleTestnet.sol#36)
factory = factory_ (contracts/oracle/SlidingWindowOracle.sol#56)
```

```
_swapPool = swapPool
(contracts/strategies/cerosYieldConverterStrategy/CerosYieldConverterStrategySp.sol#4
1)
swapPool = _swapPool (contracts/swapPool/LP.sol#26)
nativeToken = _nativeToken (contracts/swapPool/SwapPool.sol#144)
cerosToken = _cerosToken (contracts/swapPool/SwapPool.sol#145)
lpToken = _lpToken (contracts/swapPool/SwapPool.sol#146)
davosJoin = _davosJoin (contracts/vow.sol#60)
multisig = multisig_ (contracts/vow.sol#61)
multisig = data (contracts/vow.sol#78)
davosJoin = data (contracts/vow.sol#81)
davos = data (contracts/vow.sol#84)
```

Solution

Add check on zero address

Status

Fixed; Team: Majority of them are related to MakerDAO core contracts and the rest of them are related to those contracts which are taken care of in the deployment scripts.

### [N9] [Low] msg.value inside loop

Category: Denial of Service Vulnerability

Content

`msg.value` used inside loop in `MasterVault`.

- MasterVault.sol:170 (allocate)
- MasterVault.sol:277 (_withdrawFromActiveStrategies)

Consider `provide` in `DavosProvider`. When `f` in `masterVault.depositUnderlying{value: f}` larger than 0. This is possible when:

1. `isNative == true && msg.value > amount`
2. `isNative == false && msg.value > 0`

```
function provide(uint256 _amount) external payable override whenNotPaused
nonReentrant returns (uint256 value) {

    if(isNative) {
        require(msg.value >= _amount, "DavosProvider/native-less-than-amount");
        uint256 fees = msg.value - _amount;
```

```
            IWrapped(underlying).deposit{value: _amount}();
            value = masterVault.depositUnderlying{value: fees}(_amount);
        } else {
            underlying.safeTransferFrom(msg.sender, address(this), _amount);
            value = masterVault.depositUnderlying{value: msg.value}(_amount);
        }

        value = _provideCollateral(msg.sender, value);
        emit Deposit(msg.sender, value);
        return value;
    }
```

Inside `depositUnderlying`, if `allocateOnDeposit` is setted. `allocate` will be called.

```
function depositUnderlying(uint256 _amount) external payable override nonReentrant
whenNotPaused onlyOwnerOrProvider returns (uint256 shares) {

    require(_amount > 0, "MasterVault/invalid-amount");
    address src = msg.sender;

    SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable(asset()), src,
address(this), _amount);
    shares = _assessFee(_amount, depositFee);

    uint256 waitingPoolDebt = waitingPool.totalDebt();
    uint256 waitingPoolBalance =
IERC20Upgradeable(asset()).balanceOf(address(waitingPool));
    if(waitingPoolDebt > 0 && waitingPoolBalance < waitingPoolDebt) {
        uint256 waitingPoolDebtDiff = waitingPoolDebt - waitingPoolBalance;
        uint256 poolAmount = (waitingPoolDebtDiff < shares) ? waitingPoolDebtDiff :
shares;
        SafeERC20Upgradeable.safeTransfer(IERC20Upgradeable(asset()),
address(waitingPool), poolAmount);
    }

    _mint(src, shares);

    if(allocateOnDeposit == 1) allocate();

    emit Deposit(src, src, _amount, shares);
}
```

`_depositToStrategy` is called inside a loop.

```
function allocate() public {
    for(uint8 i = 0; i < strategies.length; i++) {
```

```
        if(strategyParams[strategies[i]].active) {
            StrategyParams memory strategy =  strategyParams[strategies[i]];
            uint256 allocation = strategy.allocation;
            if(allocation > 0) {
                uint256 totalAssetAndDebt = totalAssetInVault() + totalDebt;
                uint256 strategyRatio = (strategy.debt * 1e6) / totalAssetAndDebt;
                if(strategyRatio < allocation) {
                    uint256 depositAmount = ((totalAssetAndDebt * allocation) / 1e6)
- strategy.debt;
                    if(totalAssetInVault() > depositAmount && depositAmount > 0) {
                        _depositToStrategy(strategies[i], depositAmount);
                    }
                }
            }
        }
    }
}
```

Since `msg.value` remain unchanged inside loop. `IBaseStrategy(_strategy).deposit{value: msg.value}(capacity);` will revert when called second time.

Solution

Consider removing usage of `msg.value` or allocate to strategy seperately

Status

Fixed; Team: The msg.value was initially used for strategies involving fees in the form of native coin. Since, our MasterVault does auto allocate on deposit, this was problematic to predict which strategy is going to be used (For withdrawal for example, the strategy requiring the fees might not be used). Now, msg.value has been removed and currently no strategy requires fees in the form of native coin.

### [N10] [Critical] Unprotected upgrade function

Category: Authority Control Vulnerability Audit

Content

`upgradeToV2` in `contracts/oracle/MaticOraclev2.sol` is not protected. This expose vulnerability when misconfig.

```
function upgradeToV2(
    address _chainlinkAggregatorAddress,
    address _pythAddress,
    bytes32 _priceId,
```

```
    uint256 _threshold
) external {
    require(!isUpgradedForV2, "MaticOracleV2/already-upgraded");
    isUpgradedForV2 = true;
    priceFeed = AggregatorV3Interface(_chainlinkAggregatorAddress);
    pyth = IPyth(_pythAddress);
    pythPriceID = _priceId;
    threshold = _threshold;
}
```

**Solution**

Setup access control.

**Status**

Acknowledged; Team: I have confirmed this from the original dev and is not in use. Since both are related to MaticOracleV2, this oracle will be stashed.

### [N11] [Suggestion] prefer parsePriceFeedUpdates to updatePriceFeeds

**Category: Others**

**Content**

One difference between `updatePriceFeeds` and `parsePriceFeedUpdates` is:

1. `updatePriceFeeds` will not revert when passing history price.
2. `parsePriceFeedUpdates` will reject any price update outside of the time window provided in the following.

Though `pyth.updatePriceFeeds` is called in a seperated function. It can be a pitfall when used with other function and expecting it to revert when passing history price.

```
function updatePriceFeeds(bytes[] calldata priceUpdateData)//@audit use
parsePriceFeed
    external
    payable
{
    uint256 fee = pyth.getUpdateFee(priceUpdateData);
    pyth.updatePriceFeeds{value: fee}(priceUpdateData);
}
```

refs: https://docs.pyth.network/benchmarks#on-chain-contracts

**Solution**

**Status**

Acknowledged; Team: I have confirmed this from the original dev and is not in use. Since both are related to MaticOracleV2, this oracle will be stashed.

### [N12] [Suggestion] Use multisig wallet for feeReceiver

**Category: Others**

**Content**

`feeReceiver` in `contracts/masterVault/MasterVault.sol` is a EOA account and should be kept safe using multisig wallet.

**Solution**

Use multisig

**Status**

Confirmed; Team: It will be a Multisig contract.

### [N13] [Critical] Inflation attack in addLiquidity in SwapPool

**Category: Design Logic Audit**

**Content**

Consider this example: bob finds out alice is adding liquidity.

Pre-condition: no one add liquidity before.

Assume `raito = 1e18`.

1. Bob `_addLiquidity` with `amount0 = 1e8` and `amount1 = 1e8`. Bob gets `2 * 1e8 / 1e8 = 2` LP. `nativeTokenAmount = 1e8` and `cerosTokenAmount = 1e8`.
2. Bob send `4e18` native token to `SwapPool` and call `skim`. Now, `nativeTokenAmount = 1e8 + 4e18` and `cerosTokenAmount = 1e8`.
3. Alice `_addLiquidity` with `amount0 = 1e18` and `amount1 = 1e18`. Alice get `(2 * 1e18 * 2) / (1e8 + 4e18 + 1e8) = 0` LP

4. Bob remove liquidity and profit.

```
function _addLiquidity(
  uint256 amount0,
  uint256 amount1,
  bool useEth
) internal virtual {
  uint256 ratio = ICerosToken(cerosToken).ratio();
  uint256 value = (amount0 * ratio) / 1e18;
  if (amount1 < value) {
    amount0 = (amount1 * 1e18) / ratio;
  } else {
    amount1 = value;
  }
  if (useEth) {
    INativeERC20(nativeToken).deposit{ value: amount0 }();
    uint256 diff = msg.value - amount0;
    if (diff != 0) {
      _sendValue(msg.sender, diff);
    }
  } else {
    IERC20Upgradeable(nativeToken).safeTransferFrom(msg.sender, address(this),
amount0);
  }
  IERC20Upgradeable(cerosToken).safeTransferFrom(msg.sender, address(this), amount1);
  if (nativeTokenAmount == 0 && cerosTokenAmount == 0) {
    require(amount0 > 1e18, "cannot add first time less than 1 token");
    nativeTokenAmount = amount0;
    cerosTokenAmount = amount1;

    ILP(lpToken).mint(msg.sender, (2 * amount0) / 10**8);
  } else {
    uint256 allInNative = nativeTokenAmount + (cerosTokenAmount * 1e18) / ratio;
    uint256 mintAmount = (2 * amount0 * ILP(lpToken).totalSupply()) / allInNative;
    nativeTokenAmount += amount0;
    cerosTokenAmount += amount1;

    ILP(lpToken).mint(msg.sender, mintAmount);
  }
  emit LiquidityChange(msg.sender, amount0, amount1, nativeTokenAmount,
cerosTokenAmount, true);
}
```

### Solution

Introduce a offset for internal accounting.

refs: https://ethereum-magicians.org/t/address-eip-4626-inflation-attacks-with-virtual-shares-and-assets/12677/3

**Status**

Fixed; Team will deposit initial liquidity to the pool.

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002305110001 | SlowMist Security Team | 2023.04.21 - 2023.05.11 | High Risk |

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 critical risk, 6 low risk, 4 suggestion vulnerabilities. And 1 medium risk vulnerabilities were confirmed and being fixed; All other findings were fixed. 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