GUARDIAN AUDITS

# SMART CONTRACT SECURITY AUDIT OF

# GMX

# Summary

**Audit Firm:** Guardian Audits

**Client Firm:** GMX

**Final Report Date -** January 8th, 2023

## Audit Summary

After a line by line manual analysis and automated review, Guardian has concluded that:

- GMX's smart contracts have an **ACTIVE OWNERSHIP**
- Important privileged address jurisdictions:
  - Execute any order or liquidation at any arbitrary price
  - Defer execution of any arbitrary deposit, withdrawal, order, or liquidation
  - Select the order of execution for all incoming deposits, withdrawals, and orders
  - Power to shut off any feature – including withdrawals and closing positions
  - Change out the protocol-wide variables used in the data store
- GMX's privileged addresses have numerous "write" privileges. Centralization risk correlated to the active ownership is **HIGH**

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Comprehensive code coverage test suite: https://github.com/GuardianAudits/GMX_2

# Table of Contents

# Project Overview

## Project Summary

| | |
|---|---|
| Project Name | GMX |
| Language | Solidity |
| Codebase | https://github.com/gmx-io/gmx-synthetics/tree/v1.2 |
| Commit(s) | c70e419c5ff80cc14c75d622a365c7fe2a138420 |

## Audit Summary

| | |
|---|---|
| Delivery Date | January 8th, 2023 |
| Audit Methodology | Static Analysis, Manual Review, Test Suite, Contract Fuzzing |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Declined | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|---|
| Critical | 18 | 18 | 0 | 0 | 0 | 0 |
| High | 7 | 7 | 0 | 0 | 0 | 0 |
| Medium | 15 | 15 | 0 | 0 | 0 | 0 |
| Low | 45 | 45 | 0 | 0 | 0 | 0 |

# Audit Scope & Methodology

| ID\ | File | SHA-1 Checksum(s) |
|---|---|---|
| ADLU | AdlUtils.sol | 259c2ce714b95eeaa5ba89cd9963a6055be3fa14 |
| BNK | Bank.sol | 58a16bed897a7ca486a99b53fafb38c81bba206b |
| FRCV | FundReceiver.sol | ce5cdce23516a41d4b21a6c47d16caba12e7d226 |
| SBNK | StrictBank.sol | 388d4cc6e7b6595391dd9923bd3a814a25c68616 |
| CBKU | CallbackUtils.sol | e3e424cf7983800eefb4816fb83808f587042e8c |
| DCBK | IDepositCallbackReceiver.sol | aa5fdcf5bbce47804492084e538ee77a5c9a4163 |
| OCBK | IOrderCallbackReceiver.sol | afe940bd67bca7b0fc7887a5157b90bc2cbdbc7c |
| WCBK | IWithdrawalCallbackReceiver.sol | d81ec7a110e3304b5961c49e4b1c28d230096af0 |
| ARBS | ArbSys.sol | 335e28cbe4a888164d6a744d0f5fa0796e3c949f |
| CHAIN | Chain.sol | 3128b9b401cee662017b44005d56f619d16400d5 |
| DATA | DataStore.sol | ffe6854ac260fcf738b8d90cfb1b19a517736dfb |
| KEY | Keys.sol | 42ece21b7081bb62c6b0814cca49aa4fbddb7361 |
| DEP | Deposit.sol | 0a3289d25e87c6986a72f1fba794022d0ee8a2e9 |
| DEPS | DepositStore.sol | 4ec885ca50f3bd6768865bde4c4fc1e47631ff22 |
| EMIT | EventEmitter.sol | 0b45405cd433b5f86db20de97dccd345fa24b5f4 |
| DEPH | DepositHandler.sol | bfa4b6933fddd304ab01c118c14fd0105a075c00 |
| ORDH | OrderHandler.sol | 8be491e8b7c1868ee54f3d85a3a59aef53df1a97 |
| WTDH | WithdrawalHandler.sol | f2976f43235f14e492c69b4cab4fcb94a1e8dfc2 |
| FTU | FeatureUtils.sol | 25d640b35ab184e55379eca74f1bde9ed91c4272 |
| FR | FeeReceiver.sol | a0c70d46fd375e2c7d199637fe907ad6b313ce44 |

# Audit Scope & Methodology

| ID | File | SHA-1 Checksum(s) |
|---|---|---|
| GSU | GasUtils.sol | 7ec392fc74a560df194a7034df7b690de2571e9d |
| GOV | Governable.sol | 8e28e6309b41795fe83fdfa60ed8fe91866c67d1 |
| LIQU | LiquidationUtils.sol | 6ddaad9d7a10c86b3af9345d683d29f53ef81e8d |
| MKT | Market.sol | 28b8bba45e479891d7b0f5b8c7dfb311a2cea4d3 |
| MKTF | MarketFactory.sol | 2044d2f08f85c60ce58b60bba5b57d6a90cf45cd |
| MKTS | MarketStore.sol | 003cf7878b2059f8b8dde40c667fc91a5e4dd25d |
| MKTT | MarketToken.sol | a6a281e065bfa0508a6c6f4b011263f9b0b27970 |
| MKTU | MarketUtils.sol | 05892597b0ba9550a7e8cd2a92c0285b6f7ec7dc |
| NCU | NonceUtils.sol | f1fae627d2b7c712fa0469ae7891635ca1c064c9 |
| PFE | IPriceFeed.sol | a504f29cc5a8450950f3f2dbcfedcd9bef3394f9 |
| OCL | Oracle.sol | 6b31a560c2641882942f8a5ae36058bdea715adb |
| OCLM | OracleModule.sol | faf569451fcf67fa3db147cb2338825481269e58 |
| OCLS | OracleStore.sol | 9f7a648f98b236b3b87031eba446e617b7d0eb64 |
| OCLU | OracleUtils.sol | 847953586d2f31bdc878bdeb42656623c8631fc0 |
| DOU | DecreaseOrderUtils.sol | ddfbcbf3e09d629710a1502bb5f7237169ad5c55 |
| IOU | IncreaseOrderUtils.sol | 4bfd85e949e56b3594dfd3fea8090b675d865e31 |
| ORD | Order.sol | 861daff09458ff3fd03d2bbc7f68a4d19bffde44 |
| OBU | OrderBaseUtils.sol | f9d07ff449f2ee5731a21bc17064a97c3a72c710 |
| ORDS | OrderStore.sol | 96e1e5cea8007e7160afd85577f8d157813a0903 |
| FEU | FeeUtils.sol | 74477035cf27c2e20ec0b01439f56d04c0771707 |

# Audit Scope & Methodology

| ID | File | SHA-1 Checksum(s) |
|---|---|---|
| ORDU | OrderUtils.sol | c9c1cc6bb051037c21ea7a61f6a3a2f904d1dcfc |
| SWOU | SwapOrderUtils.sol | cd93d48f021cd97e867d35e6c0eec586bfca0039 |
| DPU | DecreasePositionUtils.sol | aeafcebca9a8eddd7c20637ab227d80423d6b5aa |
| IPU | IncreasePositionUtils.sol | 8ba39a298742db9952ef165261b622868fff4092 |
| POS | Position.sol | 238281c8911a90b4a68686cebe073031e0264f99 |
| POSS | PositionStore.sol | c9c4dee1900078cdfa49d39a2704f72d5318b859 |
| POSU | PositionUtils.sol | 72e63619ebb710f23a4012b8e8c2c69a04d84c71 |
| PRICE | Price.sol | 0fd280cd3ced350403f967b314206eda7ca908d0 |
| PPU | PositionPricingUtils.sol | d77acccc7fd2d77a1042bcc7a9b64be469008a26 |
| PRU | PricingUtils.sol | 34f00d0899bb67ac247a1badd6f0df563bb5ac88 |
| SPRU | SwapPricingUtils.sol | 050e9a8b852ee23f994e167799b3f0b8b968c4ed |
| READ | Reader.sol | 0b1cddc3ca983415c0605390e0260ece2ba20b80 |
| REFS | IReferralStorage.sol | 4bcdad78c2f99aba278e7bede61a746faa165d18 |
| REFT | ReferralTier.sol | 598f550410a80066564e0de088b0d3982b136ce2 |
| REFU | ReferralUtils.sol | 7aa05d81dc6a8c7342b5fae493db0c9b0fe60f64 |
| REFS | ReferralStorage.sol | 6221d4cd045cd26b8faf16c5b9168d38e1a77025 |
| ROLE | Role.sol | 8ed78872fa4481405ec9b7c2b13d72ff60bb13a1 |
| ROLEM | RoleModule.sol | 2c3d62e124ead38962153ee96af9668eb3d83106 |
| ROLES | RoleStore.sol | ffe6b2a39a1c6f6bf175f710faa5eabe59bf0404 |
| DEPU | DepositUtils.sol | ec33084815ed384d35ba6fd627de9782d5e73792 |

# Audit Scope & Methodology

| ID | File | SHA-1 Checksum(s) |
|---|---|---|
| ERTR | ExchangeRouter.sol | 55f6ce6c10cde68e3e54969fda6ade49abd8fbcb |
| RTR | Router.sol | 55d4a4cefce92d35d037c4a20c2d5f62769d27c6 |
| SWPH | SwapHandler.sol | 2db50d3a67757a36d48c8903fadc1949bf508f81 |
| SWPU | SwapUtils.sol | 27c8242bc54213bfa9d8552ef1e3fed52080c4b5 |
| TIME | Timelock.sol | aed7f668600539203956ae9a39483a4f39aa1b6c |
| IWNT | IWNT.sol | 58e6a0a78c1d29807b9cb99160993b2047dafb21 |
| TU | TokenUtils.sol | 0994d3ec7a61f91af8cf702f3a273fa212dba230 |
| ARR | Array.sol | ff174889def6318db04b3c52c9769c6332b17461 |
| BIT | Bits.sol | f8f43900d19de9e5ff5e4178cd7fe9697b96ac61 |
| CLC | Calc.sol | bcb8da7dd1ceebfbbc787a97a2408bb34b6298a1 |
| ENM | EnumerableValues.sol | 036510650590ca8ba3894a909b0673f90fe96904 |
| MC | Multicall3.sol | 0906e589d5d1d47c55b692ce85e4c03a65cc90a1 |
| NULL | Null.sol | 0c0d3ad3d7365ec71fde559329bb0f35e6bf8fa5 |
| PMC | PayableMulticall.sol | 3511a57241b1c309bff8ad2650fa21f28b0b7d35 |
| PREC | Precision.sol | 2a9f7ad8aa6ea120d2c060bfea3a94caeca80873 |
| WTD | Withdrawal.sol | 139cc06cdfdb2819f7696685ad3971168fe3cca1 |
| WTDS | WithdrawalStore.sol | 267d7890dc236cc3d4be1c8e32c4e99b696dee54 |
| WTDU | WithdrawalUtils.sol | 0e9dd4eb3025a327693df682d1859a4f58b11ef6 |

# Audit Scope & Methodology

## Vulnerability Classifications

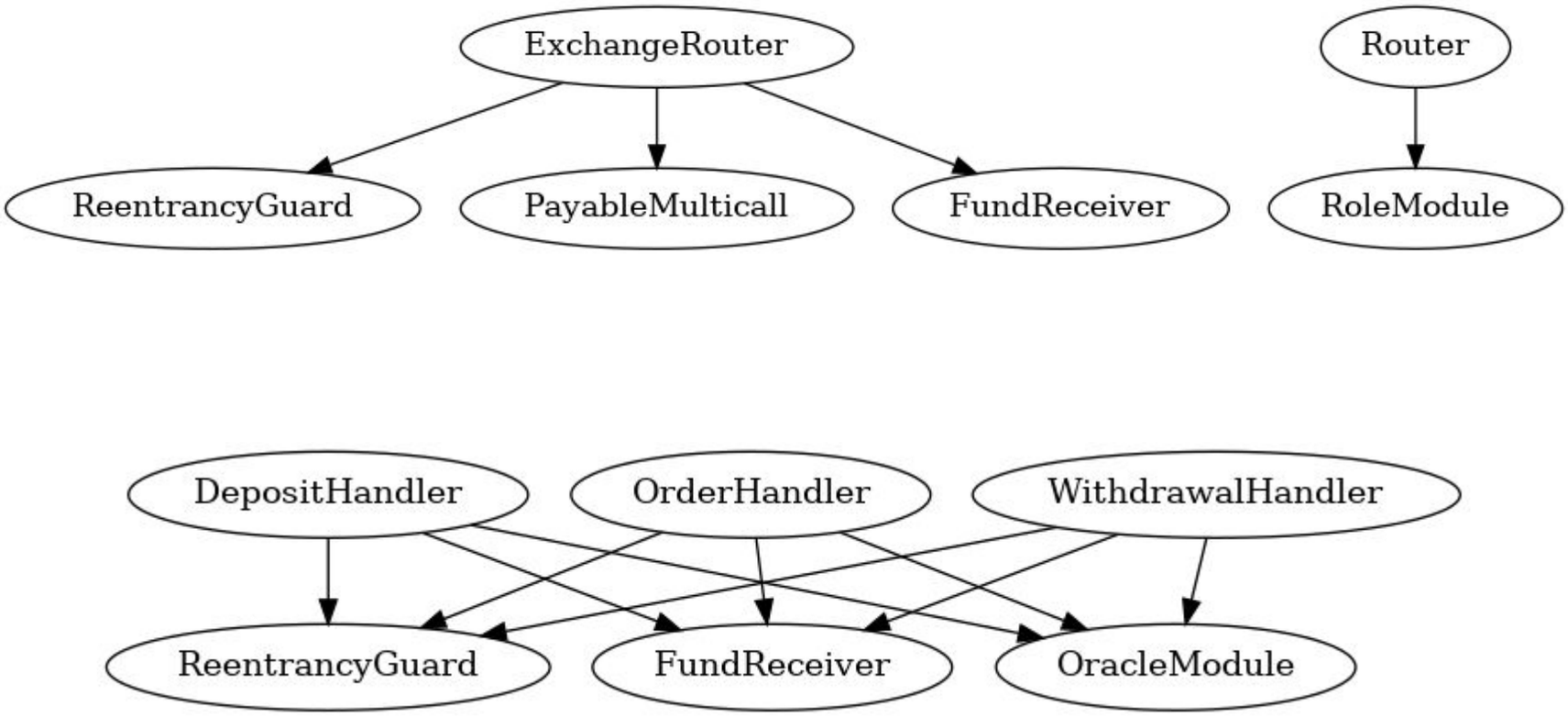
| Vulnerability Level | Classification |
| --- | --- |
| 🔴 Critical | Easily exploitable by anyone, causing loss/manipulation of assets or data. |
| 🟠 High | Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data. |
| 🟡 Medium | Inherent risk of future exploits that may or may not impact the smart contract execution. |
| 🟢 Low | Minor deviation from best practices. |

## Methodology

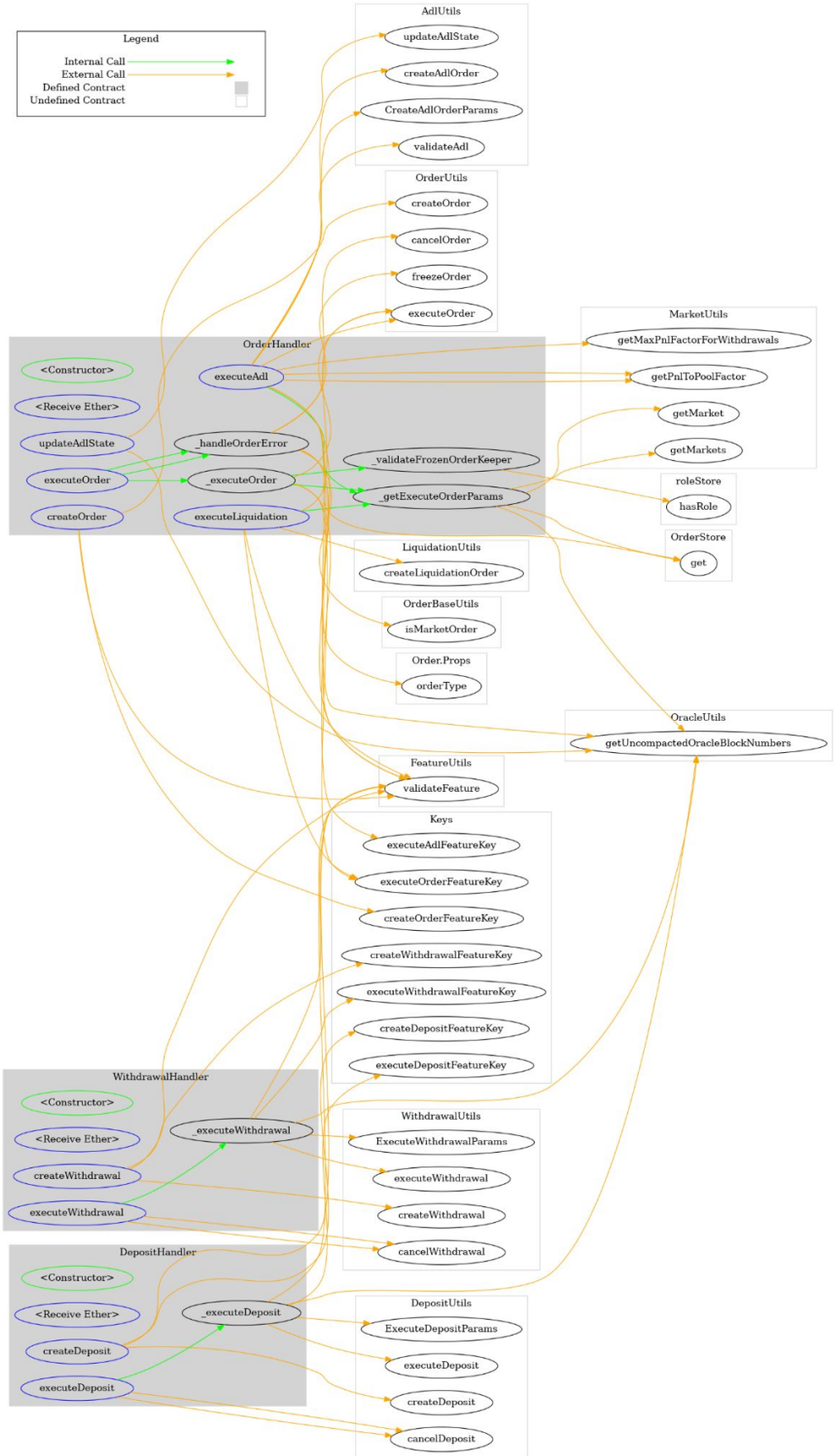
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.
- Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

# Inheritance Graph



# Call Graph



# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| ORDH-1 | Risk Free Trades From Empty Positions | Protocol Manipulation | Critical | Unresolved |
| ORDU-1 | Cancelled Order In beforeOrderExecution Callback | Protocol Manipulation | Critical | Unresolved |
| GLOBAL-1 | shouldUnwrapNativeToken DoS | Denial-of-Service | Critical | Unresolved |
| ORDU-2 | Uncancellable/Unfreezable Order | Protocol Manipulation | Critical | Unresolved |
| DPU-1 | Open Interest Errantly Increased | Logical Error | Critical | Unresolved |
| DPU-2 | Incorrect Impact Calculation | Logical Error | Critical | Unresolved |
| DPU-3 | Incorrect Fee Decrement | Logical Error | Critical | Unresolved |
| DPU-4 | Wrong Token Amount Applied | Logical Error | Critical | Unresolved |
| OBU-1 | StopLossDecrease Orders Cannot Execute | Logical Error | Critical | Unresolved |
| SWPU-1 | Price Impact Not Transferred | Logical Error | Critical | Unresolved |
| DOU-1 | Swap From Arbitrary Market | Protocol Manipulation | Critical | Unresolved |
| GLOBAL-2 | Funding Fees Not Properly Incremented | Logical Error | Critical | Unresolved |
| MKTU-1 | Unliquidateable Short With Long Collateral | Protocol Manipulation | Critical | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| IOU-1 | Token Transferred To Wrong Market | Logical Error | Critical | Unresolved |
| MKTU-2 | Incorrect Funding Per Size Calculation | Logical Error | Critical | Unresolved |
| DPU-5 | Swapping Collateral to PnL Token Inflates Output | Logical Error | Critical | Unresolved |
| MKTU-3 | Incorrect Funding Fees Accounting | Logical Error | Critical | Unresolved |
| DPU-6 | No Market Validation When Swapping Collateral to PnL | Input Validation | Critical | Unresolved |
| DOU-2 | Decrease Order Gas Attack | Gas Vamp Attack | High | Unresolved |
| DOU-3 | Incorrect LimitDecrease Size Assignment | Logical Error | High | Unresolved |
| ORDH-2 | Frozen Order Execution Loop | Gas Vamp Attack | High | Unresolved |
| GLOBAL-3 | Lack of Open Interest Caps | Lack of Controls | High | Unresolved |
| OBU-2 | Cannot Only Increase Position Collateral | Logical Error | High | Unresolved |
| ORDU-3 | Cannot Decrease Position Collateral | Logical Error | High | Unresolved |
| MKTU-3 | Broken Swap | Logical Error | High | Unresolved |
| GLOBAL-4 | Centralization Risk | Centralization /Privilege | Medium | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| ERTR-1 | Any Address May Rescue Trapped ETH | Permissioning | Medium | Unresolved |
| GLOBAL-5 | Execution Fee DoS | Denial-of-Service | Medium | Unresolved |
| GLOBAL-6 | Missing Validation | Validation | Medium | Unresolved |
| ERTR-2 | Weak Referrals | Incentives | Medium | Unresolved |
| GLOBAL-7 | Cannot Cancel Deposits/Withdrawals | Locked Funds | Medium | Unresolved |
| ORDU-4 | Unnecessary Execution Fee | Logical Error | Medium | Unresolved |
| GLOBAL-8 | Unintelligible Revert Reasons | Incorrect Parsing | Medium | Unresolved |
| OBU-3 | Unexpected Frozen Order | Logical Error | Medium | Unresolved |
| GLOBAL-9 | Possible Loss of Funds | Validation | Medium | Unresolved |
| FR-1 | Lack of Access Control For Events | Access Control | Medium | Unresolved |
| MKTU-4 | Lack of Funding Fees When OI Is Stacked | Logical Error | Medium | Unresolved |
| OCL-1 | Chainlink Feed Validation | Pricefeed Validation | Medium | Unresolved |
| MKTF-1 | Malicious Backing Token | Protocol Manipulation | Medium | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| OCL-2 | Median Price Validation | Validation | Medium | Unresolved |
| ERTR-3 | Bespoke String | Prefer Constants | Low | Unresolved |
| RST-1 | Zero Address Checks | Validation | Low | Unresolved |
| RST-2 | Function Naming | Documentation | Low | Unresolved |
| ORDH-3 | Typo | Typo | Low | Unresolved |
| ORD-1 | Typo | Typo | Low | Unresolved |
| ORD-2 | Typo | Typo | Low | Unresolved |
| DOU-4 | Typo | Typo | Low | Unresolved |
| MKTU-5 | Typo | Typo | Low | Unresolved |
| MKTU-6 | Missing Event | Events | Low | Unresolved |
| MKTU-7 | Missing Event | Events | Low | Unresolved |
| ERTR-4 | Potential Social Engineering Attack | Social Engineering | Low | Unresolved |
| IPU-1 | Superfluous Code | Optimization | Low | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| GOV-1 | Pull Over Push Ownership | Ownership / Privilege | Low | Unresolved |
| SWOU-1 | Empty Market Validation | Validation | Low | Unresolved |
| DEPU-1 | Recomputed Values | Optimization | Low | Unresolved |
| GLOBAL-10 | Lack of Constants | Prefer Constants | Low | Unresolved |
| OBU-4 | Unused Variable | Optimization | Low | Unresolved |
| GSU-1 | Superfluous Code | Optimization | Low | Unresolved |
| KEY-1 | Poor Naming Choice | Documentation | Low | Unresolved |
| OCL-3 | Outdated Variable Name | Documentation | Low | Unresolved |
| ERTR-5 | Unnecessary Variable | Optimization | Low | Unresolved |
| DPU-7 | Duplicate Condition Checks | Optimization | Low | Unresolved |
| PPU-1 | Unnecessary Variable | Optimization | Low | Unresolved |
| IOU-2 | Earlier Market Validation | Optimization | Low | Unresolved |
| OCL-4 | Cached Variables | Optimization | Low | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| MKTU-8 | Unnecessary Variable | Optimization | Low | Unresolved |
| SWU-2 | Cached Variables | Optimization | Low | Unresolved |
| PPU-2 | Recomputed Value | Optimization | Low | Unresolved |
| OCL-5 | Cached Variables | Optimization | Low | Unresolved |
| OCL-6 | Cached Variables | Optimization | Low | Unresolved |
| OCL-7 | Custom Reverts | Optimization | Low | Unresolved |
| ORDH-4 | Unnecessary Variable | Optimization | Low | Unresolved |
| SWOU-2 | Inconsistent Use of Variable | Optimization | Low | Unresolved |
| WTDH-1 | Unnecessary Variable | Optimization | Low | Unresolved |
| DEPH-1 | Unnecessary Variable | Optimization | Low | Unresolved |
| DPU-8 | Superfluous Code | Optimization | Low | Unresolved |
| NCU-1 | Unnecessary Variable | Optimization | Low | Unresolved |
| DPU-9 | Unnecessary Variable | Optimization | Low | Unresolved |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| WH-2 | Unnecessary Variable | Optimization | Low | Unresolved |
| OCL-8 | Cached Variables | Optimization | Low | Unresolved |
| LIQU-1 | Unnecessary Variables | Optimization | Low | Unresolved |
| ORDH-5 | Unnecessary Centralization | Optimization | Low | Unresolved |
| GLOBAL-11 | Initialization of Default Values | Optimization | Low | Unresolved |
| GLOBAL-12 | Cached Array Length | Optimization | Low | Unresolved |
| GLOBAL-13 | Initialization of Default Values | Optimization | Low | Unresolved |

# ORDH-1 | Risk Free Trades From Empty Positions

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | ● Critical | OrderHandler.sol: 125 | Unresolved |

## Description - PoC

The custom handling for the `Keys.EMPTY_POSITION_ERROR_KEY` allows users to create `MarketDecrease` orders that continue to revert and be retried until the user creates a position. The `MarketDecrease` order would then be executed at the prices of the block in which the decrease order was created.

This way a user can submit a long `MarketDecrease` for an empty position, and wait until the price of the index token decreases before submitting a `MarketIncrease` order and realizing risk-free profits from the exchange.

## Recommendation

Do not revert and retry on `Keys.EMPTY_POSITION_ERROR_KEY`.

# ORDU-1 | Cancelled Order In beforeOrderExecution Callback

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | ● Critical | OrderUtils.sol: 122 | Unresolved |

## Description - PoC

In the `beforeOrderExecution` callback it is possible to cancel the order prior to processing which returns funds to the user and removes the order from the `orderStore`. However, the order will still execute and create a position with the initial collateral delta and USD size.

This results in a deficit in the `orderStore` balances which causes accounting issues across the entire exchange.

## Recommendation

Do not allow order cancellation to occur during the execution of that order, possibly by moving the `cancelOrder` function to the `orderHandler` and allowing `NonReentrant` modifiers to resolve this issue. Furthermore, ensure consistency between storage and cached parameters.

# GLOBAL-1 | shouldUnwrapNativeToken DoS

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Denial-of-Service | ● Critical | Global | Unresolved |

## Description - PoC

The `shouldUnwrapNativeToken` flag can be exploited by users to create positions that cannot be decreased by liquidations or ADL orders.

For both liquidation orders and ADL orders the `shouldUnwrapNativeToken` flag is set to `true`, however the position can be created by a contract that is unable to receive the native token, causing the order execution to revert.

## Recommendation

Refactor the `shouldUnwrapNativeToken` logic so that it cannot be used to determine whether or not transactions are able to succeed, and optionally set the `shouldUnwrapNativeToken` flag to `false` for liquidation and ADL orders.

## ORDU-2 | Uncancellable/Unfreezable Order

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | ● Critical | OrderUtils.sol: 198 | Unresolved |

### Description - PoC

In the process of cancelling or freezing an order, the `executionFee` is paid to the keeper and a native token refund is issued to the user. However, the address of the user can point to a contract that is unable to accept the native token, causing the cancellation or freezing to revert.

This way, the user may cause their order execution to revert and be retried until they wish their order to be executed – enabling risk free trades with knowledge of how the market moved after their order was created.

### Recommendation

Consider refunding the user in WNT rather than the native token directly to avoid transaction manipulation.

# DPU-1 | Open Interest Errantly Increased

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | DecreasePositionUtils.sol: 319 | Unresolved |

## Description - PoC

The call to `MarketUtils.applyDeltaToOpenInterestInTokens` applies the positive `sizeDeltaInTokens` to the open interest in tokens while the position is being decreased by that amount of tokens rather than increased.

This incorrectly represents the accounting of the decrease order and perturbs all open interest, pnl and reserves accounting for that market.

## Recommendation

Negate the `sizeDeltaInTokens`, as these tokens are being removed from the open interest.

# DPU-2 | Incorrect Impact Calculation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | DecreasePositionUtils.sol: 471 | Unresolved |

## Description - PoC

The calculation of the `priceImpactAmount` is computed with the `params.order.sizeDeltaUsd`, however the price impact that the user actually experiences is based on the `adjustedSizeDeltaUsd` which can be significantly smaller than the `params.order.sizeDeltaUsd`.

This way the impact that is applied to the accounting for the pool and the impact that actually takes place can be significantly different and cause the market to start double counting funds in both the impact pool and the `poolAmount`.

## Recommendation

Compute the `priceImpactAmount` using the `adjustedSizeDeltaUsd`.

# DPU-3 | Incorrect Fee Decrement

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | DecreasePositionUtils.sol: 557 | Unresolved |

## Description - PoC

In the condition where the values.outputAmount is intended to be subtracted from the fee amount, the values.outputAmount is set to 0 before being subtracted from the fee amount, causing the fee to be taken from both the outputAmount and the user's collateral.

In the case of large fees, this can lead to significant loss of assets for users interacting with the exchange, and can potentially cause unexpected accounting within a market.

## Recommendation

Set the values.outputAmount to 0 after subtracting it from the fees.totalNetCostAmount.

# DPU-4 | Wrong Token Amount Applied

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | DecreasePositionUtils.sol: 324, 501 | Unresolved |

## Description - PoC

The `pnlAmountForPool` is either the `collateralToken` if the user realized losses or the `pnlToken` if the user realized gains. However the `applyDeltaToPoolAmount` treats this amount as `collateralToken` no matter what, therefore perturbing the accounting of the pool by applying a `pnlToken` amount to a `collateralToken` amount.

## Recommendation

Be sure to `applyDeltaToPoolAmount` for the correct token that is being removed from the pool.

# OBU-1 | StopLossDecrease Orders Cannot Execute

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | OrderBaseUtils.sol: 242 | Unresolved |

## Description - PoC

`shouldValidateAscendingPrice` errantly applies to `StopLossDecrease` orders as well, stipulating that for long position `StopLossDecrease` orders, the price must be increasing over the range to be executed and the inverse for short position `StopLossDecrease` orders.

These price range requirements are unexpected for `StopLossDecrease` orders and leads to them not acting as stop losses for positions, which would cause tremendous loss for traders.

## Recommendation

Use a separate condition to validate the price range for `StopLossDecrease` orders.

## SWPU-1 | Price Impact Not Transferred

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | SwapUtils.sol: 207 | Unresolved |

### Description - PoC

When swapping, the current `MarketToken` transfers the `poolAmountOut` to the next pool in the swap route, but this value does not include any positive price impact that the user might have accrued. This way the positive impact amount is left, unaccounted for, in the current `MarketToken` contract and the following `MarketToken` in the route “thinks” it has received the positive impact amount but it hasn’t.

This causes a deficit in the accounting for the `MarketToken` which was told that it received the positive impact amount, but never did.

### Recommendation

Be sure to send the positive impact amount to the next `MarketToken` in the swap route.

# DOU-1 | Swap From Arbitrary Market

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | ● Critical | DecreaseOrderUtils.sol: 100 | Unresolved |

## Description - PoC

When swapping after decreasing an order, the first market in the `swapPath` is not required to be the market the position was in. This way a user can specify a market that accepts the `result.outputToken` and the `result.outputAmount` will be swapped from that market, but the user's profits are left in the original market.

This breaks the accounting system in the market that was provided in place of the position's market as the first market in the `swapPath`.

## Recommendation

Transfer the funds to the first market in the `swapPath`, or require that the first market be the position's market.

# GLOBAL-2 | Funding Fees Not Properly Incremented

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | Global | Unresolved |

## Description - PoC

When both increasing and decreasing a position, funding fees are incremented for a user when the `fees.funding.longTokenFundingFeeAmount` or `fees.funding.shortTokenFundingFeeAmount` are greater than 0.

However the funding fees are intended to be paid for by the user when they are positive and received by the user when they are negative. Currently, when funding fees are positive, the user both pays for and receives funding fees. But when they are negative, the funding fees are entirely ignored.

## Recommendation

Refactor the `incrementClaimableFundingAmount` logic when both increasing and decreasing a position so that funding fees are paid out when they are negative.

# MKTU-1 | Unliquidateable Short With Long Collateral

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | ● Critical | MarketUtils.sol: 973 | Unresolved |

## Description - PoC

It is possible for user to create unliquidateable short positions with long collateral because when computing the `openInterestWithPnL` the `Calc.sum` operation underflows and reverts when the PnL is negative with magnitude greater than the open interest of the pool.

When a pool reaches this state, it is impossible to increase or decrease any positions within the pool, as the `openInterestWithPnL` is calculated during both actions.

## Recommendation

Do not allow shorts to be able to lose more than the open interest in the pool or rectify these outstanding losses somehow.

# IOU-1 | Token Transferred To Wrong Market

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | IncreaseOrderUtils.sol: 21 | Unresolved |

## Description - PoC

When creating an increase order, the collateral token gets sent to the market parameter on the order. This becomes problematic when a `swapPath` is provided, and the `initialCollateralToken` is not a token found in the order's market.

Consider the following example:

1) Order with ETHUSD market, WBTC as initial collateral, and [BTCUSD, ETHUSD] `swapPath`.
2) WBTC gets transferred to the ETHUSD market rather than the BTCUSD market
3) BTCUSD market accounting is now off as the pool thinks there is more WBTC backing positions than there really is.

## Recommendation

Transfer the collateral token to the first market in the `swapPath` if a `swapPath` is provided.

# MKTU-2 | Incorrect Funding Per Size Calculation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | MarketUtils.sol: 654-655, 660-661 | Unresolved |

## Description

`shortCollateralFundingPerSizeForLongs` is being both added to and subtracted in the same branch which not only misrepresents the true value of `shortCollateralFundingPerSizeForLongs`, but also leaves the `longCollateralFundingPerSizeForShorts` uninitialized. This negatively impacts the purpose of the funding fee which is to incentivize long and short balance in the pool.

## Recommendation

Change line 655 to `cache.fps.longCollateralFundingPerSizeForShorts -= cache.fps.fundingAmountPerSizeForLongCollateralForShorts.toInt256();`

Change line 661 to `cache.fps.longCollateralFundingPerSizeForShorts += cache.fps.fundingAmountPerSizeForLongCollateralForShorts.toInt256();`

# DPU-5 | Swapping Collateral to PnL Token Inflates Output

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | DecreasePositionUtils.sol: 375 | Unresolved |

## Description - PoC

After the swap from collateral token to PnL token, `values.outputAmount` is incremented instead of replaced with the `swapOutputAmount`. This leads to the addition of two different tokens which can drastically increase the `values.outputAmount` depending on the PnL token's precision.

For example, if `values.outputAmount` represents $10,000 USDC, and then it swaps to WETH ($2000/ETH), `swapOutputAmount` will be 5 WETH which will get added to the $10,000 USDC. WETH, with 18 decimals of precision, will drastically inflate the user's output. While this is unlikely to succeed due to the large difference in precision for USDC and WETH, tokens with closer precisions are highly susceptible to this bug.

## Recommendation

Set `values.outputAmount` to 0 and `values.pnlAmountForUser` to `swapOutputAmount`

# MKTU-3 | Incorrect Funding Fees Accounting

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● Critical | MarketUtils.sol: 864 | Unresolved |

## Description - PoC1, PoC2

If there is no open interest on a trader's side and their collateral token, the funding amount per size stamped on their position is 0. This makes them immune from providing funding fees to the other side when necessary as the funding fee amount would simply be 0.

Furthermore, a separate trader whose claimable funding fees are incremented wouldn't be receiving their tokens from the immune trader (as the funding fee to pay is always 0), but would transfer tokens directly from the market causing a disruption to the market's accounting.

## Recommendation

Consider refactoring funding fee logic so that if a newly created position is stamped with a 0 funding amount per size on their collateral token, the trader is able to pay a funding fee once there is open interest on the other side.

# DPU-6 | No Market Validation When Swapping Collateral to PnL

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Input Validation | ● Critical | DecreasePositionUtils.sol: 359 | Unresolved |

## Description

Because the `swapPath` is simply the first element of the `swapPathMarket`, a malicious user could supply an arbitrary market which contains the collateral token. Such a market does not need to contain the PnL token, even though the purpose of the swap is to get the PnL token from collateral token. With the current logic, this can be used to inflate the `values.outputAmount` as per DPU-5.

If the logic was amended so that the `cache.outputToken` was the `pnlToken`, the amount of `pnlToken` to withdraw from the market can be inflated.

## Recommendation

Add validation such that the market in the `swapPathMarkets` actually contains both the collateral token and PnL token, and that the token received from the swap is indeed the PnL token.

# DOU-2 | Decrease Order Gas Attack

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Vamp Attack | High | DecreaseOrderUtils.sol: 63 | Unresolved |

## Description - PoC

When the `sizeDeltaUsd` of a `LimitDecrease` order exceeds the position `sizeInUsd`, the order lives on in the `orderStore` and the `executionFee` is set to 0. This can lead to a potential gas vamp attack on the keeper, where a user continually submits increase orders to create small positions to be decreased by a `LimitDecrease` order that remains in the `orderStore`.

Each execution of the `LimitDecrease` order can be arbitrarily expensive due to callbacks and the keeper would receive no remuneration for a potentially large amount of gas expenditure. An orchestrated attack like this could drain the keeper of its native tokens and shut down all execution on the exchange.

## Recommendation

Require an `executionFee` for additional executions of `LimitDecrease` orders.

# DOU-3 | Incorrect LimitDecrease Size Assignment

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | High | DecreaseOrderUtils.sol: 61 | Unresolved |

## Description - PoC

When the `sizeDeltaUsd` of a `LimitDecrease` order exceeds the position `sizeInUsd`, the order lives on in the `orderStore` but the `sizeDeltaUsd` of the order is set to the `result.adjustedSizeDeltaUsd`, which is the amount that the order was just able to decrease the position by, not the amount that the order has left to decrease.

## Recommendation

Assign the `sizeDeltaUsd` of the order to be the `order.sizeDeltaUsd() - result.adjustedSizeDeltaUsd`.

## ORDH-2 | Frozen Order Execution Loop

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Vamp Attack | High | OrderHandler.sol: 299 | Unresolved |

### Description - PoC

When the execute order feature is blocked, all non-market orders will become frozen rather than being cancelled.

This could potentially spur on an infinite loop of execution for the frozen order keeper without any remuneration for gas costs.

Additionally, loops of continually failing frozen orders like this could occur from a number of errors caused during order execution.

### Recommendation

Consider cancelling all orders or simply reverting when the execute order feature is blocked. Additionally take special care around the frozen order keeper logic to avoid costly infinite loops of frozen orders–and consider requiring an `executionFee` for additional executions of orders after they are frozen.

# GLOBAL-3 | Lack of Open Interest Caps

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Lack of Controls | ● High | Global | Unresolved |

## Description

There is a lack of general open interest caps per `MarketToken` aside from the reserves validation.

This enables users to open positions and increase the open interest right up to the reserve limit so that anyone attempting to withdraw from the `MarketToken` (or possibly swap with this `MarketToken`) will be unable to pass `validateReserves`. This way depositors funds can be held hostage – only being freed if others deposit (at which point someone may increase the open interest yet again and hold these new funds hostage as well).

Additionally, in the event of arbitrage attacks the exchange has no effective mechanism to limit the open interest per market to limit the attack size.

## Recommendation

Implement open interest caps that limit the total open interest a `MarketToken` can have for either direction.

# OBU-2 | Cannot Only Increase Position Collateral

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | ● High | OrderBaseUtils.sol: 334 | Unresolved |

## Description - PoC

The price calculation in `getExecutionPrice` divides by the `sizeDeltaUsd`, therefore reverting when the `sizeDeltaUsd` is 0.

This results in users not being able to increase their collateral without increasing the size of their position. If a user were rushing to increase their collateral to avoid liquidation, the transaction would revert and the user would likely not be able to figure out why before their position becomes liquidated.

## Recommendation

Refactor the `getExecutionPrice` logic to allow for a `sizeDeltaUsd` of 0.

# ORDU-3 | Cannot Decrease Position Collateral

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | High | OrderUtils.sol: 45 | Unresolved |

## Description - PoC

When creating an order, there is no condition to handle the initialCollateralDeltaAmount for decrease orders, therefore the initialCollateralDeltaAmount is always 0 for decrease orders and it is impossible for users to decrease their collateral amount without fully closing their position.

## Recommendation

Implement a condition for the initialCollateralDeltaAmount that allows users to decrease their collateral as expected.

# MKTU-3 | Broken Swap

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | High | MarketUtils.sol: 1272 | Unresolved |

## Description

In the `getMarkets` function, the loop continues in the case of the `NO_SWAP`, `SWAP_PNL_TOKEN_TO_COLLATERAL_TOKEN`, and `SWAP_COLLATERAL_TOKEN_TO_PNL_TOKEN` addresses. This means that the index of these markets is left as an uninitialized `Market.Props` struct.

Currently, when decreasing a position and swapping the collateral token to the PnL token, the first market in the `swapPathMarkets` is used. However according to the logic in `getMarkets` this market will be uninitialized. Therefore a decrease order using the `shouldSwapCollateralTokenToPnlToken` or the `shouldSwapPnlTokenToCollateralToken` feature will always revert.

In the worst case, a user may make a `StopLossDecrease` order with either of these address variables in the `swapPath` expecting a swap to take place when their stop loss is hit. However the stop loss would revert upon execution, potentially leading to unintended loss of user funds, because their position is never closed.

## Recommendation

Limit these address variables to only the first index of the `swapPath`, and use the second item in the `swapPathMarkets` to perform the corresponding swap.

# GLOBAL-4 | Centralization Risk

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization/Privilege | Medium | Global | Unresolved |

## Description

The exchange keeper and other permissioned addresses have the power to do nearly anything:

- Execute any order or liquidation at any arbitrary price
- Defer execution of any arbitrary deposit, withdrawal, order, or liquidation
- Select the order of execution for all incoming deposits, withdrawals, and orders
- Power to shut off any feature – including withdrawals and closing positions
- Change out the protocol-wide variables used in the `dataStore`

There are many assumptions made on the form of the price input from the keeper. Adding validation for the expected format and range of acceptable inputs would reduce the risk of high-cost mistakes, and assist in limiting the scope of the internal exploits available to the keeper.

The keeper must diligently execute orders, for example stop loss orders must be executed with the correct range of prices while staying within the `MAX_ORACLE_PRICE_AGE` limit.

## Recommendation

Treat the keeper's private key(s) with the utmost level of security and introduce as many safeguard checks as possible to limit the scope of the keepers potential attack vectors.

# ERTR-1 | Any Address May Rescue Trapped ETH

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Permissioning | Medium | ExchangeRouter.sol: 102 | Unresolved |

## Description

The `sendWnt` function can be called by anyone to collect any ether that may find itself in the `ExchangeRouter` contract.

Meanwhile, the `FundReceiver` contract that the `ExchangeRouter` extends stipulates that the controller is the only address that can recover these funds using the `recoverNativeToken` function.

## Recommendation

If it is not desired that any address be able to rescue trapped ether, consider refactoring the `sendWnt` logic to be able to safely use the actual amount of ether the user provided. Otherwise, no changes are necessary.

# GLOBAL-5 | Execution Fee DoS

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Denial-of-Service | Medium | Global | Unresolved |

## Description

There is a potential DoS with `createDeposit`, `createWithdrawal`, and `createOrder` as a malicious address may send a miniscule amount of WNT to the deposit/withdrawal/order store such that another user's WNT deposit no longer matches the `executionFee` in their deposit. Thus, the deposit execution reverts and is unable to be created.

## Recommendation

Consider removing the strict equality for the `executionFee` or handling the accounting in the `depositStore` such that another address cannot increase the deposit amount for a user.

# GLOBAL-6 | Missing Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Validation | Medium | Global | Unresolved |

## Description

When creating a deposit, there is no validation that the `longToken` or the `shortToken` are valid for the supplied market. If WBTC is accidentally used as the long token in an ETH/USDC market, the user would lose their WBTC in the `depositStore`.

Furthermore, there is no validation that either the long token amount or short token amount is non-zero. This check should be added to prevent the keeper from executing trivial deposits.

Additionally, when creating an order, there is no validation that the initial collateral token is valid for the provided market, which can lead to invalid orders stored in the `orderStore`.

## Recommendation

Implement the above mentioned validations.

## ERTR-2 | Weak Referrals

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Incentives | Medium | ExchangeRouter.sol: 164 | Unresolved |

### Description

A trader is allowed to specify a different referral code each time an order is created. Only one affiliate is permitted per trader account, so when an order is created with a different affiliate, the affiliate associated with the trader's account is updated.

This can lead to affiliates missing out on rewards if a trader decides to use another affiliate's referral code even if they were the one to bring the trader onto the platform.

### Recommendation

Consider whether this is desired behavior, if not refactor the referral logic to continue to reward referrers who first bring a trader to the platform.

# GLOBAL-7 | Cannot Cancel Deposits/Withdrawals

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Locked Funds | Medium | Global | Unresolved |

## Description

A user does not have the capability to cancel their deposit or withdrawal like with an order. As a result, if the keeper for some reason does not execute their deposit/withdrawal, their funds are locked.

## Recommendation

Allow users to cancel their deposits and withdrawals and recover their funds.

## ORDU-4 | Unnecessary Execution Fee

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | OrderUtils.sol: 198 | Unresolved |

### Description

A user can cancel an order on their own without the need of a keeper. Even if a user cancels an order on their own accord, the `executionFee` is paid to the keeper which doubles the amount of gas a user expends.

### Recommendation

Do not pay the `executionFee` to the keeper if the user is simply cancelling the order.

# GLOBAL-8 | Unintelligible Revert Reasons

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Incorrect Parsing | Medium | Global | Unresolved |

## Description

In every `catch (bytes memory _reason)` case, the reason is parsed with `string(abi.encode(_reason))` However, parsing the bytes reason like this results in unintelligible revert strings.

The first 4 bytes of the `_reason` bytes represent the selector for the error that caused the revert, and the rest represent the data that accompanies the error. There ought to be a way (perhaps off-chain) to map from these 4 bytes to the error type and decode the data.

Additionally, panic reverts will be caught in this case and should not be parsed.

## Recommendation

Consider an alternative approach to parsing the bytes revert reasons, and know that it is possible by chance for the 4 byte selector for two separate errors to be the same if they are not defined in the same contract.

# OBU-3 | Unexpected Frozen Order

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | OrderBaseUtils.sol: 249 | Unresolved |

## Description

When a limit order is executed with an invalid increasing/decreasing price range, the order reverts with a bespoke revert string–which results in the order getting frozen. But this is unexpected as all other order price-related errors simply revert and will be retried. This might lead to a poor execution of limit order types or a complete lack of execution of limit orders.

## Recommendation

Consider reverting in this case with a custom error that is handled similarly to the `UNACCEPTABLE_PRICE_ERROR_KEY`.

# GLOBAL-9 | Possible Loss of Funds

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Validation | Medium | Global | Unresolved |

## Description

When creating a decrease order or a swap order, there is no validation that the `receiver` address is not the zero address. In this case, when the order is executed and a swap takes place, the user's funds will be left in the `MarketToken`, otherwise if no swap is executed during the decrease, the funds would be sent to the zero address.

## Recommendation

Do not allow users to possibly lose funds this way and validate that the `receiver` is not the zero address for decrease and swap orders.

# FR-1 | Lack of Access Control For Events

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Access Control | Medium | FeeReceiver.sol: 22 | Unresolved |

## Description

Anyone can call `notifyFeeReceived` as there is a lack of access control, which leads to a depreciation of the authenticity of the event.

## Recommendation

Implement access control concerning who may call `notifyFeeReceived`.

# MKTU-4 | Lack of Funding Fees When OI Is Stacked

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | MarketUtils.sol: 619 | Unresolved |

## Description

When computing funding fees in the `getNextFundingAmountPerSize` function, the `cache.oi.longOpenInterest == 0 || cache.oi.shortOpenInterest == 0` condition stipulates that when all of the open interest is stacked on one side no funding fees are charged to the users who have positions on the stacked side. This means there is a lack of disincentive for the users with stacked positions to switch sides.

In such a case, the funding fees could be redirected to the `poolAmount` for the benefit of LPers.

Exchanges like Binance have a minimum funding fee that is held at all times to properly disincentivize such imbalances.

## Recommendation

Consider whether or not funding fees should be charged when there is no opposing side open interest. If funding fees should in fact be charged when only one side has all of the open interest, refactor the existing logic to charge funding fees and optionally send them to the pool or an arbitrary fee receiver.

# OCL-1 | Chainlink Feed Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Pricefeed Validation | Medium | Oracle.sol: 536 | Unresolved |

## Description

Extra validation checks should be added on the result from the Chainlink price feed to ensure non-stale data. The price from the data feed influences the execution of orders and liquidations so it is imperative the data is up to date and correct.

## Recommendation

Add the following require statements to validate the price feed:

- `require(answeredInRound >= roundID, "Chainlink:: Stale price")`
- `require(timestamp > 0, "Chainlink:: Round not complete")`

# MKTF-1 | Malicious Backing Token

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | Medium | MarketFactory.sol: 31 | Unresolved |

## Description

In the future, GMX hopes to enable permissionless `MarketToken` creation. However, a `MarketToken` can use an arbitrary malicious backing token that performs mischievous actions such as cancelling orders on the exchange during execution when the token is transferred.

Additionally, it is possible for contracts of the existing tokens on the exchange to be upgraded with new logic that is able to maliciously exploit the exchange.

## Recommendation

Be wary of malicious tokens and ensure there is no way to re-enter into the application during token transfers.

# OCL-2 | Median Price Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Validation | Medium | Oracle.sol: 491, 492 | Unresolved |

## Description

There should be validation that the `medianMaxPrice` is greater than the `medianMinPrice`. Otherwise, many critical exchange values such as `poolValue`, `pnlToPoolFactor`, and many others could be adversely affected.

## Recommendation

Implement the validation that `medianMaxPrice` is greater than the `medianMinPrice`.

# ERTR-3 | Bespoke String

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Prefer Constants | ● Low | ExchangeRouter.sol: 252 | Unresolved |

## Description

When cancelling an order, a bespoke `“USER_INITIATED_CANCEL”` string is used as the reason. This reason string might be better suited as a constant variable that can be reused when checking to see if the cancel was user generated.

## Recommendation

Consider making the `“USER_INITIATED_CANCEL”` string a constant variable.

## RST-1 | Zero Address Checks

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Validation | ● Low | ReferralStorage.sol: 127 | Unresolved |

### Description

Currently, there is no zero address check for the `_newAccount`. Users may accidentally burn their code ownership if the zero address is provided.

### Recommendation

Consider adding a zero address check for the `_newAccount`.

## RST-2 | Function Naming

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Documentation | Low | IReferralStorage.sol | Unresolved |

### Description

The `codeOwners`, `referrerDiscountShares`, and `referrerTiers` functions should be made singular as they all return a single value rather than multiple.

Additionally, the `tiers` function may be better named as `tierValues` as this more clearly represents what the function returns.

### Recommendation

Consider renaming the `codeOwners`, `referrerDiscountShares`, `referrerTiers`, and `tiers` functions.

## ORDH-3 | Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Typo | Low | OrderHandler.sol: 139 | Unresolved |

### Description

The comment “the account of the position to liquidation” should read “the account of the position to liquidate”.

### Recommendation

Update the comment.

## ORD-1 | Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Typo | Low | Order.sol: 30 | Unresolved |

### Description

In the comment, “current” is misspelled as “curent”.

### Recommendation

Update the comment.

## ORD-2 | Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Typo | Low | Order.sol: 65 | Unresolved |

### Description

In the comment, “the initial token sent it for the swap” should read “the initial token sent in for the swap”.

### Recommendation

Update the comment.

## DOU-4 | Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Typo | Low | DecreaseOrderUtils.sol: 10 | Unresolved |

### Description

In the comment, “Library” is misspelled as “Libary”.

### Recommendation

Update the comment.

## MKTU-5 | Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Typo | Low | MarketUtils.sol: 1004 | Unresolved |

### Description

In the comment, “for withdrawals a market” should read as “for withdrawals for a market”

### Recommendation

Update the comment.

# MKTU-6 | Missing Event

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Events | Low | MarketUtils.sol: 570 | Unresolved |

## Description

There is no corresponding event emitted when the `fundingAmountPerSize` is set in `updateFundingAmountPerSize`.

## Recommendation

Consider emitting an event when updating the `fundingAmountPerSize`.

# MKTU-7 | Missing Event

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Events | Low | MarketUtils.sol: 680 | Unresolved |

## Description

There is no corresponding event emitted when the `cumulativeBorrowingFactor` is set in `updateCumulativeBorrowingFactor`.

## Recommendation

Consider emitting an event when updating the `cumulativeBorrowingFactor`.

# ERTR-4 | Potential Social Engineering Attack

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Social Engineering | Low | ExchangeRouter.sol: 266, 295 | Unresolved |

## Description

Since the `claimFundingFees` and `claimAffiliateReward` functions accept an arbitrary `receiver` address, they are somewhat prone to frontend code injection or related social engineering attacks.

## Recommendation

Consider sending the rewards directly to the `msg.sender` for increased security. If the `receiver` address is kept, be sure to validate that it is not the zero address so users are not able to accidentally burn their fees.

# IPU-1 | Superfluous Code

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | IncreasePositionUtils.sol: 93 | Unresolved |

## Description

At the beginning of the `increasePosition` function the `account`, `market`, `collateralToken`, and `isLong` are set on the position from the params. But the position must already have these values as it is created with them and keyed off of them.

## Recommendation

Remove these lines, otherwise document why they are necessary.

# GOV-1 | Pull Over Push Ownership

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Ownership / Privilege | ● Low | Governable.sol: 27 | Unresolved |

## Description

The ownership transfer process should have a push and pull step rather than executing in a single transaction with `setGov`.

This way the protocol may avoid catastrophic errors such as setting the wrong governance address.

## Recommendation

Implement a two step transfer process for the `gov` role where the new `gov` address must explicitly accept its new role.

# SWOU-1 | Empty Market Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Validation | Low | SwapOrderUtils.sol: 19 | Unresolved |

## Description

Swap orders are required to be executed with a zero address market, this should be validated when first creating a swap order to save the keeper from executing orders that will possibly trivially fail.

## Recommendation

Consider adding validation that the market address is 0 for all swap orders in the `createOrder` function.

# DEPU-1 | Recomputed Values

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | DepositUtils.sol: 190, 203 | Unresolved |

## Description

The `longTokenUsd` and `shortTokenUsd` values are computed on line 190, but those same values are recomputed on line 203.

## Recommendation

Instead of recomputing the `longTokenUsd` and `shortTokenUsd`, use those variables in place of the recomputation.

# GLOBAL-10 | Lack of Constants

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Prefer Constants | Low | Global | Unresolved |

## Description

Every time the `emitSwapFeesCollected` function is called, the `action` bytes are recomputed. These common `action` bytes would be better suited as “action keys” in the Keys.sol contract.

## Recommendation

Create keys for the `action` bytes.

# OBU-4 | Unused Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | OrderBaseUtils.sol: 88 | Unresolved |

## Description

In the `ExecuteOrderParams` struct, the `positionKey` is never set or used.

## Recommendation

Remove the extraneous `positionKey`.

## GSU-1 | Superfluous Code

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | GasUtils.sol: 158-177 | Unresolved |

### Description

The `estimateExecuteIncreaseOrderGasLimit`, `estimateExecuteDecreaseOrderGasLimit`, and `estimateExecuteSwapOrderGasLimit` all have the same exact logic, the only difference being the order type gas limit key.

A common function that abstracts over the order type gas limit key could de-duplicate these three estimation functions.

### Recommendation

Implement a single estimation function that can be used with any order type gas limit key.

# KEY-1 | Poor Naming Choice

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Documentation | ● Low | Keys.sol: 118 | Unresolved |

## Description

The `maxPnlFactorForWithdrawals` variable is poorly named. It serves as a lower bound for the `pnlToPoolFactor` after an ADL has been executed, but the naming as a max and relation to withdrawals are not immediately obvious without additional documentation.

## Recommendation

Rename the variable or provide more documentation expounding upon the naming.

# OCL-3 | Outdated Variable Name

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Documentation | Low | Oracle.sol: 100 | Unresolved |

## Description

The name of the `uint256` emitted with the `MaxPriceAgeExceeded` error is `blockNumber`, however this `uint256` now represents a timestamp rather than a block number.

## Recommendation

Update the variable name appropriately.

# ERTR-5 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | ● Low | ExchangeRouter.sol | Unresolved |

## Description

Throughout the `ExchangeRouter` contract, an `account` variable is used to store the `msg.sender`, but this variable is often only used once. Instead of storing this variable on the stack, reference `msg.sender` directly to save gas.

## Recommendation

Remove the sub-optimal declarations of the `account` address variable.

# DPU-7 | Duplicate Condition Checks

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | DecreasePositionUtils.sol: 168, 169 | Unresolved |

## Description

The `isLong` boolean is checked twice with two ternary operators, but it would save gas to only check `isLong` once with an `if` condition.

## Recommendation

Replace the two ternaries with a single `if` condition.

# PPU-1 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | PositionPricingUtils.sol: 174 | Unresolved |

## Description

The `openInterestParams` variable is declared and only used once, rather than declaring it use the return value of `getNextOpenInterest` directly in the call to `_getPriceImpactUsd`

## Recommendation

Remove the declaration of the `openInterestParams` variable.

# IOU-2 | Earlier Market Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | IncreaseOrderUtils.sol: 27 | Unresolved |

## Description

The market can be checked if it is empty before transferring the collateral token to save gas upon failure.

## Recommendation

Move validation to the top of the function.

# OCL-4 | Cached Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Oracle.sol: 428, 432, 443 | Unresolved |

## Description

The `Chain.currentBlockNumber()` and `Chain.currentTimestamp()` values can be stored outside the for-loop to save gas.

## Recommendation

Implement the suggestion above.

# MKTU-8 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | MarketUtils.sol: 318 | Unresolved |

## Description

Gas can be saved by not declaring the `pnl` variable before returning it.

## Recommendation

Directly return the value.

## SWU-2 | Cached Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | SwapUtils.sol: 94, 95, 102 | Unresolved |

### Description

The `nextIndex`, `receiver`, and `_params` variables can be declared outside the for-loop and re-assigned upon each iteration to save gas.

### Recommendation

Implement the suggestion above.

# PPU-2 | Recomputed Value

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | PositionPricingUtils.sol: 321 | Unresolved |

## Description

The `fees.totalNetCost` includes the `fees.positionFeeAmountForPool` and `fees.borrowingFeeAmount` in it's summation calculation, but these two numbers were already summed for the `fees.feesForPool` variable. The `fees.feesForPool` can be used to sum the `totalNetCost` and save some gas.

## Recommendation

Implement the suggestion above.

# OCL-5 | Cached Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Oracle.sol: 243, 249 | Unresolved |

## Description

The `signerIndex` and `signerIndexBit` variables can be declared outside the for-loop and re-assigned upon each iteration to save gas.

## Recommendation

Implement the suggestion above.

# OCL-6 | Cached Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Oracle.sol: 491, 492 | Unresolved |

## Description

The `medianMinPrice` and `medianMaxPrice` variables can be declared outside the for-loop and re-assigned upon each iteration to save gas.

## Recommendation

Implement the suggestion above.

# OCL-7 | Custom Reverts

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Oracle.sol: 461 | Unresolved |

## Description

The check `require(priceFeedAddress != address(0), "Oracle: invalid price feed")` could use a custom revert to save gas.

## Recommendation

Implement the suggestion above.

## ORDH-4 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | OrderHandler.sol: 360 | Unresolved |

### Description

The `isMarketOrder` variable is only used once. Rather than allocating stack variable space, the `if` condition can simply use `OrderBaseUtils.isMarketOrder(order.orderType())` to reduce gas expenditure.

### Recommendation

Implement the suggestion above.

# SWOU-2 | Inconsistent Use of Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | SwapOrderUtils.sol: 29 | Unresolved |

## Description

The order is stored as a variable here, but it is often simply accessed through the params. Either undeclare it to save gas or use it to access every order attribute rather than repeatedly using the params.

## Recommendation

Implement the suggestion above.

# WTDH-1 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | WithdrawalHandler.sol: 125 | Unresolved |

## Description

The `ExecuteWithdrawalParams` to execute a withdrawal can be created directly in the `executeWithdrawal` function call to save gas.

## Recommendation

Implement the suggestion above.

# DEPH-1 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | DepositHandler.sol: 128 | Unresolved |

## Description

The `ExecuteDepositParams` to execute a deposit can be created directly in the `executeDeposit` function call to save gas.

## Recommendation

Implement the suggestion above.

## DPU-8 | Superfluous Code

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | DecreasePositionUtils.sol: 439 | Unresolved |

### Description

The `values.remainingCollateralAmount -= params.order.initialCollateralDeltaAmount().toInt256()` calculation can occur on the line above where `values.remainingCollateralAmount` is declared.

### Recommendation

Implement the suggestion above.

# NCU-1 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | NonceUtils.sol: 33 | Unresolved |

## Description

Instead of declaring the `key` variable, simply return the hash to save on gas expenditure.

## Recommendation

Implement the suggestion above.

# DPU-9 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | DecreasePositionUtils.sol: 380 | Unresolved |

## Description

Instead of declaring the `reason` variable, pass the result directly to the event.

## Recommendation

Implement the suggestion above.

## WH-2 | Unnecessary Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | WithdrawalHandler.sol: 92 | Unresolved |

### Description

Instead of declaring the `reason` variable, pass the result directly to `cancelWithdrawal` .

### Recommendation

Implement the suggestion above.

# OCL-8 | Cached Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Oracle.sol: 523 | Unresolved |

## Description

The `token`, `priceFeed`, `_price`, `price`, `precision`, `stablePrice`, and `priceProps` variables can be declared outside the for-loop and re-assigned upon each iteration to save gas.

## Recommendation

Implement the suggestion above.

# LIQU-1 | Unnecessary Variables

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | LiquidationUtils.sol: 36, 45, 56 | Unresolved |

## Description

The `address`, `numbers`, `flags`, and `order` variables can be created directly when calling the `orderStore.set` function to save gas.

## Recommendation

Considering the trade-off in readability, Implement the suggestion above.

## ORDH-5 | Unnecessary Centralization

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | OrderHandler.sol: 178 | Unresolved |

### Description

There is no in-code requirement that the keeper must `updateAdlState` to turn ADL mode off. This requires users to trust that the keeper will turn ADL off when necessary, but this could be avoided by adding a time range to ADL mode or validating the `maxPnlFactor` upon ADL execution.

### Recommendation

Consider adding a time range to ADL mode or validating the `maxPnlFactor` upon ADL execution.

# GLOBAL-11 | Initialization of Default Values

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Low | Global | Unresolved |

## Description

Throughout the codebase the index for many for-loops in initialized to 0, the default value for `uint`. Avoid the unnecessary initialization and allow the default values to be implicitly assigned to these `uint` variables:

- MarketUtils.sol::1272
- Oracle.sol::242
- Oracle.sol::287
- Oracle.sol::424
- Oracle.sol::454
- Oracle.sol::472
- Oracle.sol::523
- OracleUtils.sol::121
- Reader.sol::59
- ExchangeRouter.sol::273
- ExchangeRouter.sol::302
- SwapUtils.sol::92
- Timelock.sol::44
- Timelock.sol::57
- Array.sol::37
- Array.sol::54
- PayableMulticall.sol::20

## Recommendation

Implement the suggestion above.

# GLOBAL-12 | Cached Array Length

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | ● Low | Global | Unresolved |

## Description

Throughout the codebase there are many for-loops that compute the length of an array upon each iteration. Caching the length of these arrays will decrease gas costs throughout the application:

- MarketUtils.sol::1272
- Oracle.sol::242
- Oracle.sol::424
- Oracle.sol::454
- Oracle.sol::472
- Oracle.sol::523
- Reader.sol::59
- ExchangeRouter.sol::273
- ExchangeRouter.sol::302
- SwapUtils.sol::92
- Timelock.sol::44
- Timelock.sol::57
- PayableMulticall.sol::20

## Recommendation

Implement the suggestion above.

# GLOBAL-13 | Initialization of Default Values

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | ● Low | Global | Unresolved |

## Description

Throughout the codebase `uint` variables are checked to be `> 0` however `!= 0` is a more efficient comparison to check if `uint` values are positive. Switching to `!= 0` will decrease gas costs throughout the application:

- DepositUtils.sol::215
- DepositUtils.sol::233
- DepositUtils.sol::295
- DepositUtils.sol::304
- Oracle.sol::551
- DecreaseOrderUtils.sol::74
- OrderUtils.sol::182
- DecreasePositionUtils.sol::302
- DecreasePositionUtils.sol::345
- DecreasePositionUtils.sol::386
- DecreasePositionUtils.sol::552
- IncreasePositionUtils.sol::243
- IncreasePositionUtils.sol::292
- TokenUtils.sol::186
- WithdrawalUtils.sol::233
- WithdrawalUtils.sol::255

## Recommendation

Implement the suggestion above.

# Auditor’s Verdict

After a line by line manual analysis and automated review, Guardian has concluded that:

- GMX’s smart contracts have an **ACTIVE OWNERSHIP**
- Important privileged address jurisdictions:
  - Execute any order or liquidation at any arbitrary price
  - Defer execution of any arbitrary deposit, withdrawal, order, or liquidation
  - Select the order of execution for all incoming deposits, withdrawals, and orders
  - Power to shut off any feature – including withdrawals and closing positions
  - Change out the protocol-wide variables used in the data store
- GMX’s privileged addresses have numerous “write” privileges. Centralization risk correlated to the active ownership is **HIGH**

# Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

# About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits