GUARDIAN
AUDITS

SMART CONTRACT SECURITY AUDIT OF

GMX

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand

Client Firm GMX

Final Report Date June 2nd, 2023

Audit Summary

GMX engaged Guardian to review the security of its decentralized synthetics perpetuals exchange. From the 23rd of May to the 2nd of June, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

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, Avalanche**

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

Table of Contents

Project Information

Project Overview 4

Audit Scope & Methodology 5

Smart Contract Risk Assessment

Inheritance Graph 11

Findings & Resolutions 12

Addendum

Disclaimer 33

About Guardian Audits 34

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics-updated/commits/internal-fixes
Commit(s)	1ff3bacc68a0718d6d561d7502157862c7f7bb68

Audit Summary

Delivery Date	June 2nd, 2023
Audit Methodology	Static Analysis, Manual Review, Written Tests

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
🔴 Critical	0	0	0	0	0	0
🟠 High	1	0	0	0	0	1
🟡 Medium	9	0	0	0	0	9
🟢 Low	9	0	0	0	0	9

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
ADLU	AdlUtils.sol	2c7321b50446e8bf7a79719fd98fde681ab4b280
BNK	Bank.sol	814ae68a75e21621c3d22c174eea28c45fb2118f
SBNK	StrictBank.sol	8235d39cfa13186c2b4a4fbadfab029bcd8f91d2
CBKU	CallbackUtils.sol	bc1d4d41503db65b399610e4c3bf3b9472f4b87b
DCBK	IDepositCallbackReceiver.sol	aed58b8d02e950c17f1e375d1ff4537e20d4d460
OCBK	IOrderCallbackReceiver.sol	b986dcf7d9deb75f6cbb6e630a3f7d2a27f75374
WCBK	IWithdrawalCallbackReceiver.sol	d85b4c126911ed219a4bb13349a35887e9b6db84
ARBS	ArbSys.sol	0d9703a3477e40ccce9b0b526c0c9f4310034496
CHAIN	Chain.sol	020d318af7d3d4ecba2cdf36669c582923611ae9
CON	Config.sol	93106e7ee27a8c29deb0a854f78c4a7a0a63b94b
TIME	Timelock.sol	17fc8ce10a5756e757a042c6385deda7daa60dcd
DATA	DataStore.sol	60cd23b1a42c8a8b663e3a5e2f756b841fff7200
KEY	Keys.sol	3024e52e1eafe990ae65dfc2124e9702d788f8b8
DEP	Deposit.sol	f15405bbbafc9a723e2a596193919975b25c35e8
DEPS	DepositStoreUtils.sol	aefb9405cbeeeaa26c31a66d124386b34aad1b12
DEPU	DepositUtils.sol	c871656056f44bc02dd8dcd7cb18e18f3bbd44aa
DEPV	DepositVault.sol	1d19ad5afc0baec27a608a2f53cbb5b6f48f8f26
EDU	ExecuteDepositUtils.sol	30fa8a15a1bdc8f4af2facdef152b98d08dea131
ADLH	AdlHandler.sol	655ac0ce383e6f12c30a3014a325ca3dd8ba5762
BOH	BaseOrderHandler.sol	9383f59bf6b5508bdb67b0201907a82a6719312c

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
DEPH	DepositHandler.sol	1c1298a0459edf0c267ca03b972b6cd7c25f6bfd
EU	ExchangeUtils.sol	92d707b3877211bac90d669f641c40f86a4c6858
LIQH	LiquidationHandler.sol	c5b3c7089b41b94e1d36f4e0492bf758e106547a
LIQU	LiquidationUtils.sol	b6b9912ac014f788f75ecb08949a9d4edcf00d4b
ORDH	OrderHandler.sol	b3f6beb9bbfc13e70cfa1670d65c80bc4b9748e4
WTDH	WithdrawalHandler.sol	9bd4e57ac77ea657488efd8bca655cf3ddc5a3ff
MKTS	MarketStoreUtils.sol	d0d6795a715d7cec428fe4333a37da5df650b3e7
FU	FeatureUtils.sol	c13c0754b300f9db33673ddda7d10443f5897d24
FEH	FeeHandler.sol	63bcddb25b38037cdb4ce99b9e9bf4936aa46bed
FEU	FeeUtils.sol	86ceab5e579c88f0cde1627f6e46d3dbdc4f5cb6
GSU	GasUtils.sol	0532683bdb842a447bfdbf15ba0564f4e06e0a75
LIQU	LiquidationUtils.sol	3181ad9d6d64a3bc5428e87464d103381c53ac1b
MKT	Market.sol	a66a2a9127674ffb23d74d7a252f046f98c2e182
MKTF	MarketFactory.sol	77deb5eb5fe4bae2de7471ae66c4f6cdef5a9a92
MKTSU	MarketStoreUtils.sol	d0d6795a715d7cec428fe4333a37da5df650b3e7
MKTT	MarketToken.sol	a55f9a9931d906583050b4f01b74b7adbe54cf1d
MKTU	MarketUtils.sol	71796927a98696b0a5f1a30d0fdaa216894e8e2b
NONCE	NonceUtils.sol	6ec2082417987d5c4e859adefb9b28efb1ed5c39
IPF	IPriceFeed.sol	431babdd9ab4ee30ae9eba84f469620a3d2951f3
OCL	Oracle.sol	d67693b91b26a84ce8e84fd375970ed07dfe840f

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
OCLM	OracleModule.sol	d20d435823840fd73e58b2b9f9270ef8663093dd
OCLS	OracleStore.sol	1e6a95ac567b91c345c1647a82e62d7fd00617e2
OCLU	OracleUtils.sol	96a8ae6230ceeabe0cc471487d1d2fd8a354511e
BOU	BaseOrderUtils.sol	3ab93a3ea0aa0f97e0a2e10c7cf26213fdbe79f5
DOU	DecreaseOrderUtils.sol	3f171729dc3055f1ae7867949aad477c8941d7d5
IOU	IncreaseOrderUtils.sol	1e69e800f02b8c034bbc83c2602ddb3194b58bc6
ORD	Order.sol	090ce71a5e61445b7288267bf157dc4927f4e6a1
ORDSU	OrderStoreUtils.sol	3dfdd3dcaf4b55eab0ac21639c096994062fa266
ORDU	OrderUtils.sol	0ffa9b101bfa4a4e6e92c6602b7b711ed9d4618d
ORDV	OrderVault.sol	65051e5535ff27531518d29b779792df02e191c7
DPCU	DecreasePositionCollateralUtils.sol	88dad5c90fea4845fb956cac03a1c08dd2bbc99c
DPU	DecreasePositionUtils.sol	78489065744b179300ba784141320452e62e7334
IPU	IncreasePositionUtils.sol	7ad663427c15de3ff78692867999ff10bb90f261
POS	Position.sol	d6dced94def32ea2786c29749a3bea2f4e9e4202
PSU	PositionStoreUtils.sol	2c3dd7a46f4541311eecbb80a30d6650c1362eb3
POSU	PositionUtils.sol	2c217328bc44bc2b9a867b1797c25eeb028a36eb
PRICE	Price.sol	c1f87807a20c43c1710d1e3c3e628e265cd5686e
PPU	PositionPricingUtils.sol	41c9eb4a6e49f22d376334df5dfa4dc2cfb1e901
PU	PricingUtils.sol	2f55f33f64f01e06b0aa5b928651ec4496ce4ba9
SWPPU	SwapPricingUtils.sol	73a5c8671e031ff38415c3316df676a84ab30524

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
IREFS	IReferralStorage.sol	2c0ee57b1c26ab40381e52535407c23e3443b5b8
REFT	ReferralTier.sol	8a34d5e24b6a317b063ebd59d85fa1fec9307ea7
REFU	ReferralUtils.sol	617bf4115a4d5a42f4fff58c37fd5651ad74af0b
ROL	Role.sol	86935a3af0c782e711076d1a2ad2222bda7185fa
ROLM	RoleModule.sol	b3e74811c0f6a46ff26da1474fd48969314d4938
ROLS	RoleStore.sol	dc8d62c33546e6a79368b2096e102c511eba108a
ERTR	ExchangeRouter.sol	51d97104f248e9c4d5b1cf2bb7a1de7ab384d8c1
RTR	Router.sol	0fde38bae3c62565cda7fec0ba521a46611d6e32
SWPH	SwapHandler.sol	9e3bb4bb999a70390ff2be5f447a7d4ffd5699c5
SWPU	SwapUtils.sol	f0a75866cc0a8191d1f4fc37d2b32c9fb64b9aa9
IWNT	IWNT.sol	f5776a90a5a9dcb5f89cd7f19341ae1d769a1bc5
ARR	Array.sol	475174aabc82306f52589c927641ce4c85f79e29
BM	BasicMulticall.sol	6ab8cb8c0369fc13d1caac3c7be4287d4cdbe8dc
BITS	Bits.sol	c7fa3c25af05c172cff6faccef14182665b875ba
CALC	Calc.sol	6ce439db40dd185a189d93b121441d8ee45717cb
CAST	Cast.sol	68780489ad9ee795bf3d0574e96b399d36504f58
ENV	EnumerableValues.sol	36354b53a39c4fb584313f8d3aac8e2b091d90a2
ERRU	ErrorUtils.sol	6e1290f8503c73a2a0f96f82d7d975aad22eb231
GREG	GlobalReentrancyGuard.sol	4f4a5deed4a1f00e7a349f87f5af802b85e8ba3b
PM	PayableMulticall.sol	d4748b4b4fa4715f63fac17d0f406627d64658da

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
PREC	Precision.sol	327da594b4f829b1dc21c548bf4e7a3c176aba79
RECU	ReceiverUtils.sol	601ac421b66d87c441960ec7b7e94aa5fbaeaa49
WTD	Withdrawal.sol	86a4ddc39df006b71c0ffc8366f401845488a9d1
WTDS	WithdrawalStoreUtils.sol	bf7b0e22c6c1975dd05f2663e161b67d482f9434
WTDU	WithdrawalUtils.sol	a63f7511edfa427108e6a0bfd9e79a606e929a4a
WTDV	WithdrawalVault.sol	5cc2b331b13f735dfebc983b9aec705692e0d2a2

Audit Scope & Methodology

Vulnerability Classifications

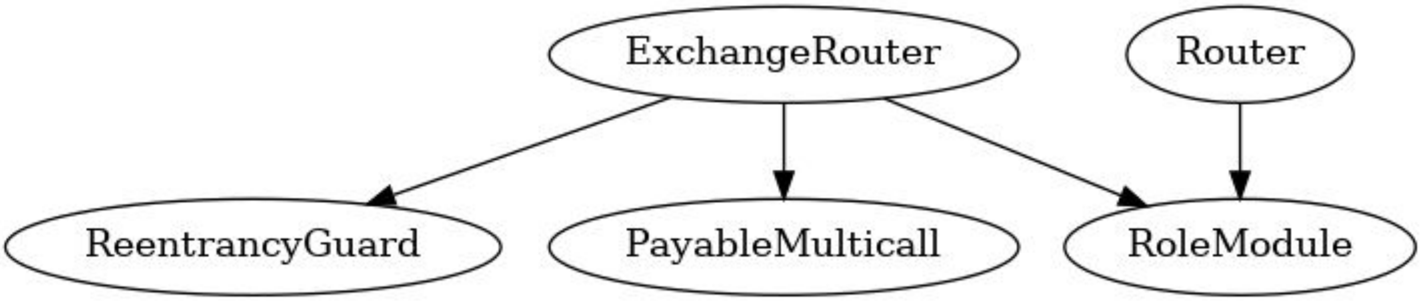
Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract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

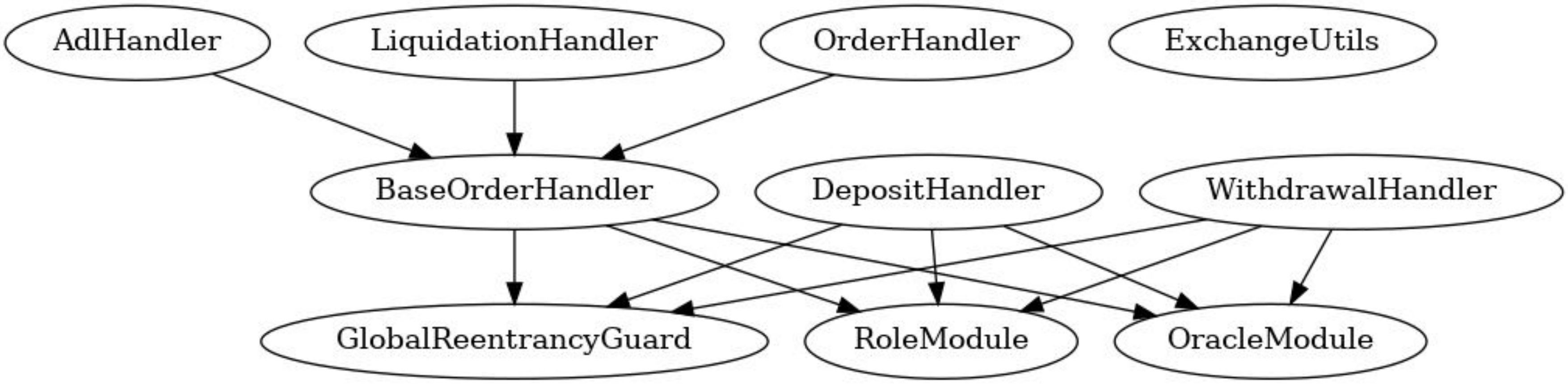
Methodology

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





Findings & Resolutions

ID	Title	Category	Severity	Status
DPCU-1	priceImpactDiffUsd Unclaimable For Adjusted PnL	Logical Error	● High	Resolved
DPCU-2	Fees May Be Errantly Credited To The Pool	Misaccounting	● Medium	Resolved
PU-1	Inaccurate Price Impact Formula	Logical Error	● Medium	Resolved
BOU-1	Incongruent Price Impact For Decrease Orders	Logical Error	● Medium	Resolved
MKTU-1	Wrong Impact Pool Maximization	Logical Error	● Medium	Resolved
ORDU-1	Read-only Reentrancy	Reentrancy	● Medium	Resolved
DPCU-3	Capped PnL Leads To Incongruent Accounting	Misaccounting	● Medium	Resolved
DPCU-4	adjustedPriceImpactDiffAmount Minimized	Logical Error	● Medium	Resolved
DPCU-5	Liquidation Reverts Due To Underflow	Underflow	● Medium	Resolved
DPCU-6	Position Price Impact Not Offset	Misaccounting	● Medium	Resolved
DPCU-7	Invalid priceImpactDiffUsd Emitted	Events	● Low	Resolved
DPCU-8	Event Emission For Insufficient Payment	Events	● Low	Resolved
POSU-1	User’s PnL Differs From Pool PnL	Warning	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
OCL-1	Inefficient Validation	Optimization	● Low	Resolved
OCL-2	Unnecessary Parameter	Superfluous Code	● Low	Resolved
ERR-1	Unnecessary Error	Superfluous Code	● Low	Resolved
MKTU-2	Unnecessary Cache Attributes	Superfluous Code	● Low	Resolved
CON-1	Duplicated Key In _initAllowedBaseKeys	Superfluous Code	● Low	Resolved
ADLH-1	Crowded Code Style	Formatting	● Low	Resolved

DPCU-1 | priceImpactDiffUsd Unclaimable For Adjusted PnL

Category	Severity	Location	Status
Logical Error	● High	DecreasePositionCollateralUtils.sol: 111, 157	Resolved

Description PoC

If the positionPnlUsd is positive but smaller than the priceImpactDiffUsd, the adjustedPositionPnlUsd is set to 0. However the condition for accounting for the pnlDiffAmount and making that amount claimable for the user is dependent on adjustedPositionPnlUsd > 0.

Therefore, cases where the priceImpactDiffUsd cannot be entirely fulfilled by the positionPnlUsd result in the user being unable to claim their pnl that was used to cover a portion of the priceImpactDiffUsd.

Recommendation

Change the condition to adjustedPositionPnlUsd ≥ 0 or make the incrementClaimableCollateralAmount call directly when the PnL is decreased by the priceImpactDiffUsd.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

DPCU-2 | Fees May Be Errantly Credited To The Pool

Category	Severity	Location	Status
Misaccounting	● Medium	DecreasePositionCollateralUtils.sol: 473	Resolved

Description

In the processForceClose function, the amountForPool is the remainingCollateral minus the fundingFees.

However it is possible in some cases that this amountForPool includes amounts that were meant to be subtracted from the collateral for other beneficiaries other than the pool. For example the feeReceiver, uiFeeReceiver, and affiliate.

This situation can arise when the pendingCollateralDeduction is only slightly larger than the remaining collateral, and the exact deduction that put the collateral deduction over the remaining collateral threshold is one of these fees that should not be distributed to the pool.

Recommendation

Consider decrementing these fees from the amountForPool and crediting as much as possible to the rightful receivers.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

PU-1 | Inaccurate Price Impact Formula

Category	Severity	Location	Status
Logical Error	● Medium	PricingUtils.sol: 120	Resolved

Description PoC

The comment in the applyImpactFactor function states the following:

We divide by 2 here to more easily translate liquidity into the appropriate impactFactor values. For example, if the impactExponentFactor is 2 and we want to have an impact of 0.1% for $2 million of difference we can set the impactFactor to be 0.1% / 2 million, in factor form that would be 0.001 / 2,000,000 * (10 ^ 30)

However this additional divisor of 2 is redundant, especially in the given example.

Consider the diffUsd of 2,000,000 and an impactExponentFactor of 2 (ignoring units):

exponentValue = 2,000,000 * 2,000,000; impactFactor = 0.001 / 2,000,000

exponentValue * impactFactor = 2,000,000 * 2,000,000 * .001 / 2,000,000 / 2 = 2,000,000 * .001 / 2

Without the extra division by 2 we already have the result we're looking for, 2,000,000 * .001 = 2,000 since 2,000,000 and 1/2,000,000 cancelled the additional x2 introduced.

This directly contradicts the example given in the applyImpactFactor function.

Recommendation

Remove the redundant 1/2.

Resolution

GMX Team: The recommendation was implemented.

BOU-1 | Incongruent Price Impact For Decrease Orders

Category	Severity	Location	Status
Logical Error	● Medium	BaseOrderUtils.sol: 350	Resolved

Description PoC

The formula used to compute the executionPrice in the BaseOrderUtils.getExecutionPrice function does not agree with the priceImpactAmount calculation in the PositionPricingUtils.getPriceImpactAmount function during decrease orders.

When calculating the executionPrice, the priceImpactUsd is applied in a fraction with the sizeDeltaUsd. This agrees with the getPriceImpactAmount calculations for increase orders, as the sizeDeltaUsd and the executionPrice determine the trader's sizeInTokens and therefore their immediate PnL.

However, when closing a position, the trader realizes PnL based on the sizeInTokens and executionPrice, not the sizeDeltaUsd and executionPrice. Therefore the effect that price impact has on the trader's PnL is not accurately reflected by the calculation for the executionPrice. Ultimately because of this, the executionPrice and resulting trader's PnL do not agree with the priceImpactAmount generated by the PositionPricingUtils.getPriceImpactAmount function.

Recommendation

Consider applying the priceImpactUsd directly to the trader's PnL rather than manipulating the executionPrice for decrease orders.

Resolution

GMX Team: The executionPrice logic was refactored.

MKTU-1 | Wrong Impact Pool Maximization

Category	Severity	Location	Status
Logical Error	● Medium	MarketUtils.sol: 369	Resolved

Description

The Impact pool pricing should use `!maximize` for the index token since `impactPoolUsd` is being deducted, however this calculation uses `maximize`.

Recommendation

Change `maximize` to `!maximize` for the index token valuation.

Resolution

GMX Team: The recommendation was implemented.

ORDU-1 | Read-only Reentrancy

Category	Severity	Location	Status
Reentrancy	● Medium	OrderUtils.sol: 244	Resolved

Description

In the `OrderUtils.cancelOrder` function, the `orderVault.transferOut` is executed before the order is removed from the `dataStore`.

Recommendation

Move the removal of the order from the `dataStore` to before the `orderVault.transferOut` call.

Resolution

GMX Team: The recommendation was implemented.

DPCU-3 | Capped PnL Leads To Incongruent Accounting

Category	Severity	Location	Status
Misaccounting	● Medium	DecreasePositionCollateralUtils.sol: 89	Resolved

Description

In the event that a trader's PnL is capped, the PnL they experience from price impact may not be accurately represented by the change in balance of the position impact pool, therefore perturbing the pool value.

For example: A trader is positively impacted but their PnL is capped. The capping of their PnL essentially changes their `executionPrice` and negates the positive impact they received.

However this positive impact is still removed from the position impact pool to offset the immediate gain in PnL the trader would have realized from the impact.

Therefore the trader does not actually experience the PnL boost from the price impact amount, but that amount is still credited towards the pool value with the removal from the position impact pool.

Recommendation

Consider computing what ought to be removed from the position impact pool after the trader's PnL is capped.

Resolution

GMX Team: The logic in `DecreasePositionCollateralUtils.sol` was refactored.

DPCU-4 | adjustedPriceImpactDiffAmount Minimized

Category	Severity	Location	Status
Logical Error	● Medium	DecreasePositionCollateralUtils.sol: 117	Resolved

Description

While converting the adjustedPriceImpactDiffUsd to a collateral token amount, the collateralTokenPrice.max is used. However the collateralTokenPrice.max will result in a smaller adjustedPriceImpactDiffAmount.

In scenarios where the max price has a nontrivial difference with the min price, e.g. a depeg event, this can lead to users paying significantly less for this capped price impact amount than they ought to.

Recommendation

Use the collateralTokenPrice.min when translating the adjustedPriceImpactDiffUsd to a adjustedPriceImpactDiffAmount.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

DPCU-5 | Liquidation Reverts Due To Underflow

Category	Severity	Location	Status
Underflow	● Medium	DecreasePositionCollateralUtils.sol: 456	Resolved

Description

Although rare, there are cases where the `pendingCollateralDeduction` is smaller than the `fees.funding.fundingFeeAmount`, resulting in a revert upon the `cache.remainingCostAmount` calculation in the `processForceClose` function.

Consider the following:

- Position with 1 token of collateral
- Fees of total 11 tokens
- Funding fees of 3 tokens
- Profit of 9 tokens

In this case, the profit is used to cover 9 tokens of the `fees.collateralCostAmount`, so the remaining `fees.collateralCostAmount` is 2 tokens. Therefore the `values.pendingCollateralDeduction` will be larger than the `values.remainingCollateralAmount` and the execution will enter the `processForceClose` function.

However when the `remainingCostAmount` is computed, the funding fees (3 tokens) will be subtracted from the pending deduction (2 tokens) and revert.

Recommendation

Although this scenario will be rare, the percentage of funding fees that may be covered by position profit should be accounted for to avoid an underflow.

Resolution

GMX Team: The logic in `DecreasePositionCollateralUtils.sol` was refactored.

DPCU-6 | Position Price Impact Not Offset

Category	Severity	Location	Status
Mis-accounting	● Medium	DecreasePositionCollateralUtils.sol: 294	Resolved

Description

During the force closure of a position during a liquidation or ADL order, the accounting for the position impact pool with applyDeltaToPositionImpactPool is skipped. However the effects of the price impact were already felt on the position's resulting PnL.

This results in scenarios where a user is significantly negatively/positively impacted during a liquidation/ADL and this amount is not reflected by the position impact pool and so the pool value is asymmetrically effected.

For instance, a user's PnL is positively impacted by $100 during a force close liquidation. This positive impact is translated to a decrease of the pool value by $100.

The positive impact is not offset by a decrease in the position impact pool, and therefore the pool realizes immediate losses from PI.

Vice-versa for the pool realizing immediate gains on negative price impact, although when a position is negatively impacted, it contributes to the collateral + pnl not being sufficient and therefore the necessary accounting becomes less straightforward. In these scenarios, the position impact pool ought to only be increased by the amount that the position actually experienced, as it wasn't able to cover it's entire losses/negative impact — effectively exactly offsetting whatever amount was "payable" (or actually was able to take effect) of the negative impact.

Recommendation

This is somewhat non-trivial to address in the negative impact case as mentioned above, however for the positive impact case, the full impact amount should be applied to the position impact pool, as this full amount is experienced by the trader.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

DPCU-7 | Invalid priceImpactDiffUsd Emitted

Category	Severity	Location	Status
Events	● Low	DecreasePositionCollateralUtils.sol: 556	Resolved

Description

During the processForceClose function, the priceImpactDiffUsd is assigned to 0 in the returned values, however there may have been a nonzero priceImpactDiffUsd that was applied to the adjustedPositionPnlUsd.

At present, this priceImpactDiffUsd would be misrepresented as 0 in the emitPositionDecrease function call.

Recommendation

Compute the amount of priceImpactDiffUsd that was applied to the adjustedPositionPnlUsd and return that as a part of the DecreasePositionCollateralValues in the processForceClose function.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

DPCU-8 | Event Emission For Insufficient Payment

Category	Severity	Location	Status
Events	● Low	DecreasePositionCollateralUtils.sol: 419	Resolved

Description

In the event that a position is force closed and the secondaryOutput or remainingCollateralAmount is insufficient to cover the position costs, it may be helpful to emit an event indicating the remainingCostAmount that was left uncovered.

Recommendation

Emit an event at the end of the processForceClose function if the remainingCostAmount is greater than 0. Similar to the logic for emitInsufficientFundingFeePayment.

Resolution

GMX Team: The logic in DecreasePositionCollateralUtils.sol was refactored.

POSU-1 | User's PnL Differs From Pool PnL

Category	Severity	Location	Status
Warning	● Low	PositionUtils.sol: 176-177	Resolved

Description

The PnL of a user's position is based upon their executionPrice, which deviates from the index price due to price impact. If we consider the scenario of a single trader in the market, when the user decreases their position, the position's PnL will not be equal to pool's PnL obtained from MarketUtils.getPnl.

As a result, this may lead to the pool's PnL less likely to be capped when +PI is experienced since the pool's PnL will be smaller. Similarly, this may lead to the pool's PnL more likely to be capped when -PI is experienced since the pool's PnL will be larger.

Recommendation

Be aware of this edge case, it may not be necessary to address it directly with a code change but is worth considering when configuring the PnL caps as well as other relevant variables.

Resolution

GMX Team: The executionPrice logic was refactored.

OCL-1 | Inefficient Validation

Category	Severity	Location	Status
Optimization	● Low	Oracle.sol: 468	Resolved

Description

The check `if (!primaryPrices[reportInfo.token].isEmpty())` can be performed at the top of the `for` loop to save gas as it unnecessary to perform all the price processing for this check.

Recommendation

Implement the above recommendation.

Resolution

GMX Team: The recommendation was implemented.

OCL-2 | Unnecessary Parameter

Category	Severity	Location	Status
Superfluous Code	● Low	Oracle.sol: 585	Resolved

Description

All calls to emitOraclePriceUpdated have parameter isPrimary as true. Thus, the parameter may be removed.

Recommendation

Implement the above recommendation.

Resolution

GMX Team: The recommendation was implemented.

ERR-1 | Unnecessary Error

Category	Severity	Location	Status
Superfluous Code	● Low	Errors.sol: 202	Resolved

Description

The InvalidPoolAdjustment error could be removed as it is never used.

Recommendation

Implement the above recommendation.

Resolution

GMX Team: The recommendation was implemented.

MKTU-2 | Unnecessary Cache Attributes

Category	Severity	Location	Status
Superfluous Code	● Low	MarketUtils.sol: 2406-2407	Resolved

Description

The cache.collateralForLongs and cache.collateralForShorts amounts have been removed from the aggregate minTokenBalance logic and are now validated individually.

However these values are still added in the result for the getExpectedMinTokenBalance function and reside in the GetExpectedMinTokenBalanceCache.

Recommendation

Remove the cache.collateralForLongs and cache.collateralForShorts values from the summation in the getExpectedMinTokenBalance function as well as the GetExpectedMinTokenBalanceCache struct.

Resolution

GMX Team: The recommendation was implemented.

CON-1 | Duplicated Key In _initAllowedBaseKeys

Category	Severity	Location	Status
Superfluous Code	● Low	Config.sol: 199	Resolved

Description

The `MAX_POSITION_IMPACT_FACTOR_FOR_LIQUIDATIONS` key is duplicated in the `_initAllowedBaseKeys` function.

Recommendation

Remove one of the duplicated `allowedBaseKeys[Keys.MAX_POSITION_IMPACT_FACTOR_FOR_LIQUIDATIONS] = true;` lines.

Resolution

GMX Team: The duplicated key was removed.

ADLH-1 | Crowded Code Style

Category	Severity	Location	Status
Formatting	● Low	AdlHandler.sol: 149	Resolved

Description

The msg.sender parameter is crowded on the same line as the oracleParams.

Recommendation

Put the msg.sender parameter on it’s own line.

Resolution

GMX Team: The recommendation was removed.

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