GMX · SECURITY REVIEW

GMX

GMX V2.2.1 Review.

A security review of GMX V2.2.1 Review.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
GMX	GMX V2.2.1 Review	Owen Thurm, Andreas Zottl, Nicholas Chew, Ferran Celades	July 17 – October 23, 2026	November 26, 2026

Published at guardian.security/portfolio/gmx/v2-2-1-review-2026-11-26

§ 00

Contents.

139 pages: overview, confidence ranking, scope, 102 findings, disclaimer, and about Guardian.

§ 01 **Overview** P. 03

§ 02 **Confidence Ranking** P. 04

§ 03 **Scope & Classifications** P. 05

Files in Scope P. 06

§ 04 **Findings** P. 12

Main Review · 69 findings P. 17

Remediation Review 1 · 18 findings P. 100

Remediation Review 2 · 10 findings P. 120

Remediation Review 3 · 4 findings P. 131

Remediation Review 4 · 1 finding P. 137

§ 05 **Disclaimer** P. 138

§ 06 **About Guardian** P. 139

§ 01

Overview.

Guardian reviewed GMX V2.2.1 Review across 6 review rounds. The final report includes 8 High, 37 Medium, 49 Low, 8 Informational findings.

AUDIT FIRM
Guardian

CLIENT
GMX

PROJECT
GMX V2.2.1 Review

PREPARED BY
Owen Thurm, Andreas Zottl, Nicholas Chew, Ferran Celades

AUDIT WINDOW
July 17 – October 23, 2026

FINAL REPORT
November 26, 2026

REPOSITORY
https://github.com/gmx-io/gmx-synthetics-private

METHODOLOGY
Smart contract · Manual · Main Review · Remediation Review 1 · Remediation Review 2 · Remediation Review 3 · Remediation Review 4 · Remediation Review 5

MAIN REVIEW COMMIT
ffacc466bfee

REMEDIATION REVIEW 1 COMMIT
ad2eaa17afff

REMEDIATION REVIEW 2 COMMIT
a0afb112f915

REMEDIATION REVIEW 3 COMMIT
7fbd664f3cbf

1.1 Confidence Ranking

4 / 5

High Confidence

Guardian assigns a Confidence Ranking of 4 out of 5, combining finding severity, remediation outcome, and the reviewer's assessment of latent risk. The full scale is on the next page.

1	2	3	4	5
VERY LOW CONFIDENCE	LOW CONFIDENCE	MODERATE CONFIDENCE	**HIGH CONFIDENCE**	VERY HIGH CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED
Critical	**0**	0	0	0	0
High	**8**	0	1	0	7
Medium	**37**	0	15	0	22
Low	**49**	0	35	0	14
Informational	**8**	0	6	0	2
All severities	**102**	**0**	**57**	**0**	**45**

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence ASSIGNED	Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity findings.
3 Moderate Confidence	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed GMX V2.2.1 Review in `https://github.com/gmx-io/gmx-synthetics-private`. The scope includes 41,432 source lines across 58 files.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH	Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.
IMPACT · MEDIUM	Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.
IMPACT · LOW	Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.
LIKELIHOOD · HIGH	Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.
LIKELIHOOD · MEDIUM	Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.
LIKELIHOOD · LOW	Unlikely to ever occur in production.

Files in Scope 1/6.

FILE	NSLOC	TOTAL
MAIN REVIEW		
contracts/config/Config.sol	528	691
contracts/config/ConfigSyncer.sol	120	174
contracts/config/ConfigUtils.sol	338	437
contracts/config/RiskOracleConfig.sol	209	278
contracts/data/Keys.sol	1,531	2,504
contracts/data/Keys2.sol	61	138
contracts/deposit/DepositUtils.sol	165	220
contracts/deposit/ExecuteDepositUtils.sol	424	593
contracts/error/Errors.sol	389	547
contracts/exchange/GlvWithdrawalHandler.sol	124	159
contracts/fee/FeeDistributor.sol	440	587
contracts/fee/FeeDistributorUtils.sol	77	98
contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol	222	259
contracts/glv/glvDeposit/GlvDepositUtils.sol	248	288
contracts/glv/glvShift/GlvShiftUtils.sol	227	276
contracts/glv/GlvUtils.sol	265	347
contracts/market/MarketUtils.sol	1,847	3,133
contracts/multichain/IMultichainProvider.sol	25	33
contracts/multichain/IMultichainStakingRouter.sol	4	8
contracts/multichain/LayerZeroProvider.sol	227	370
contracts/multichain/LayerZeroProviderUtils.sol	342	387
contracts/multichain/MultichainReader.sol	189	238
contracts/multichain/MultichainStakingRouter.sol	77	94
contracts/multichain/MultichainStakingUtils.sol	123	157
contracts/multichain/MultichainTransferRouter.sol	164	230
contracts/order/ExecuteOrderUtils.sol	101	141
contracts/position/DecreasePositionCollateralUtils.sol	547	735
contracts/position/DecreasePositionUtils.sol	250	378
contracts/position/Position.sol	116	206
contracts/position/PositionEventUtils.sol	241	287
contracts/position/PositionStoreUtils.sol	203	268
contracts/position/PositionUtils.sol	503	788
contracts/reader/GlvReader.sol	145	175
contracts/reader/Reader.sol	191	233
contracts/reader/ReaderDepositUtils.sol	148	176

Files in Scope 2/6.

FILE	NSLOC	TOTAL
contracts/reader/ReaderPositionUtils.sol	193	248
contracts/reader/ReaderPricingUtils.sol	207	279
contracts/reader/ReaderUtils.sol	194	245
contracts/router/relay/BaseGelatoRelayRouter.sol	325	454
contracts/router/relay/EIP6492Deployer.sol	7	15
contracts/router/relay/IRelayUtils.sol	82	137
contracts/router/relay/MultichainRelayUtils.sol	349	381
contracts/router/relay/RelayUtils.sol	379	429
contracts/router/relay/SignatureUtils.sol	76	177
contracts/router/relay/StakingUtils.sol	83	99
contracts/router/relay/SubaccountRouterUtils.sol	65	86
contracts/staking/GmxAccountWalletFactory.sol	45	58
contracts/swap/SwapHandler.sol	11	26
contracts/swap/SwapUtils.sol	297	441
REMEDIATION REVIEW 1		
contracts/config/Config.sol	340	463
contracts/config/ConfigKeys.sol	207	261
contracts/config/ConfigUtils.sol	376	483
contracts/config/FundingConfigUtils.sol	26	32
contracts/config/RiskOracleConfig.sol	205	274
contracts/data/Keys.sol	1,544	2,522
contracts/data/Keys2.sol	84	190
contracts/deposit/DepositUtils.sol	170	226
contracts/deposit/ExecuteDepositUtils.sol	428	598
contracts/error/Errors.sol	453	612
contracts/exchange/GlvWithdrawalHandler.sol	127	163
contracts/fee/FeeDistributor.sol	582	804
contracts/fee/FeeDistributorUtils.sol	129	167
contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol	224	263
contracts/glv/glvDeposit/GlvDepositUtils.sol	250	290
contracts/glv/glvShift/GlvShiftUtils.sol	229	279
contracts/glv/GlvUtils.sol	265	347
contracts/market/MarketUtils.sol	1,909	3,215
contracts/multichain/IMultichainProvider.sol	14	22
contracts/multichain/LayerZeroProvider.sol	273	437
contracts/multichain/LayerZeroProviderUtils.sol	205	240

Files in Scope 3/6.

FILE	NSLOC	TOTAL
contracts/multichain/MultichainReader.sol	196	249
contracts/multichain/MultichainTransferRouter.sol	176	248
contracts/order/ExecuteOrderUtils.sol	108	148
contracts/position/DecreasePositionCollateralUtils.sol	541	726
contracts/position/DecreasePositionUtils.sol	273	407
contracts/position/Position.sol	151	262
contracts/position/PositionEventUtils.sol	250	296
contracts/position/PositionStoreUtils.sol	258	338
contracts/position/PositionUtils.sol	575	891
contracts/reader/GlvReader.sol	172	204
contracts/reader/Reader.sol	204	248
contracts/reader/ReaderDepositUtils.sol	148	176
contracts/reader/ReaderPositionUtils.sol	199	254
contracts/reader/ReaderPricingUtils.sol	207	279
contracts/reader/ReaderUtils.sol	202	253
contracts/router/relay/BaseGelatoRelayRouter.sol	321	444
contracts/router/relay/EIP6492Deployer.sol	45	99
contracts/router/relay/IRelayUtils.sol	75	130
contracts/router/relay/MultichainRelayUtils.sol	350	382
contracts/router/relay/RelayUtils.sol	411	476
contracts/router/relay/SignatureUtils.sol	79	181
contracts/router/relay/SubaccountRouterUtils.sol	65	86
contracts/swap/SwapHandler.sol	17	33
contracts/swap/SwapUtils.sol	310	476
REMEDIATION REVIEW 2		
contracts/config/Config.sol	340	463
contracts/config/ConfigKeys.sol	223	278
contracts/config/ConfigSyncer.sol	116	169
contracts/config/ConfigUtils.sol	376	483
contracts/config/FundingConfigUtils.sol	26	32
contracts/config/RiskOracleConfig.sol	205	274
contracts/data/Keys.sol	1,580	2,584
contracts/data/Keys2.sol	91	193
contracts/deposit/DepositUtils.sol	170	226
contracts/deposit/ExecuteDepositUtils.sol	428	598
contracts/error/Errors.sol	449	612

Files in Scope 4 / 6.

FILE	NSLOC	TOTAL
contracts/exchange/GlvWithdrawalHandler.sol	127	163
contracts/fee/FeeDistributor.sol	567	785
contracts/fee/FeeDistributorUtils.sol	114	144
contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol	224	263
contracts/glv/glvDeposit/GlvDepositUtils.sol	250	290
contracts/glv/glvShift/GlvShiftUtils.sol	229	279
contracts/glv/GlvUtils.sol	265	347
contracts/market/MarketUtils.sol	1,909	3,215
contracts/multichain/IMultichainProvider.sol	27	35
contracts/multichain/IMultichainStakingRouter.sol	4	8
contracts/multichain/LayerZeroProvider.sol	244	391
contracts/multichain/LayerZeroProviderUtils.sol	370	420
contracts/multichain/MultichainReader.sol	196	249
contracts/multichain/MultichainStakingRouter.sol	91	110
contracts/multichain/MultichainStakingUtils.sol	182	278
contracts/multichain/MultichainTransferRouter.sol	176	248
contracts/order/ExecuteOrderUtils.sol	108	148
contracts/position/DecreasePositionCollateralUtils.sol	541	726
contracts/position/DecreasePositionUtils.sol	265	396
contracts/position/Position.sol	144	250
contracts/position/PositionEventUtils.sol	249	295
contracts/position/PositionStoreUtils.sol	247	324
contracts/position/PositionUtils.sol	546	852
contracts/reader/GlvReader.sol	172	204
contracts/reader/Reader.sol	204	248
contracts/reader/ReaderDepositUtils.sol	148	176
contracts/reader/ReaderPositionUtils.sol	199	254
contracts/reader/ReaderPricingUtils.sol	207	279
contracts/reader/ReaderUtils.sol	202	253
contracts/router/relay/BaseGelatoRelayRouter.sol	321	444
contracts/router/relay/EIP6492Deployer.sol	45	99
contracts/router/relay/IRelayUtils.sol	83	139
contracts/router/relay/MultichainRelayUtils.sol	350	382
contracts/router/relay/RelayUtils.sol	411	476
contracts/router/relay/SignatureUtils.sol	79	181
contracts/router/relay/StakingUtils.sol	94	112

Files in Scope 5/6.

FILE	NSLOC	TOTAL
contracts/router/relay/SubaccountRouterUtils.sol	65	86
contracts/staking/GmxAccountWalletFactory.sol	88	144
contracts/swap/SwapHandler.sol	17	33
contracts/swap/SwapUtils.sol	310	476
REMEDIATION REVIEW 3		
contracts/config/Config.sol	2	463
contracts/config/ConfigKeys.sol	101	261
contracts/config/ConfigUtils.sol	1	483
contracts/config/FundingConfigUtils.sol	0	32
contracts/config/RiskOracleConfig.sol	0	274
contracts/data/Keys.sol	19	2,522
contracts/data/Keys2.sol	4	190
contracts/deposit/DepositUtils.sol	0	226
contracts/deposit/ExecuteDepositUtils.sol	0	598
contracts/donation/GmDepositProxy.sol	251	325
contracts/error/Errors.sol	13	613
contracts/exchange/GlvWithdrawalHandler.sol	0	163
contracts/fee/FeeDistributor.sol	19	804
contracts/fee/FeeDistributorUtils.sol	15	167
contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol	0	263
contracts/glv/glvDeposit/GlvDepositUtils.sol	0	290
contracts/glv/glvShift/GlvShiftUtils.sol	0	279
contracts/glv/GlvUtils.sol	0	347
contracts/incentive/EsGmxIssuer.sol	2	188
contracts/incentive/IEsGmxIssuer.sol	2	6
contracts/incentive/IRatioVester.sol	1	6
contracts/incentive/RatioVester.sol	7	548
contracts/market/MarketUtils.sol	0	3,215
contracts/multichain/IMultichainProvider.sol	1	22
contracts/multichain/LayerZeroProvider.sol	41	437
contracts/multichain/LayerZeroProviderUtils.sol	32	240
contracts/multichain/MultichainReader.sol	0	249
contracts/multichain/MultichainTransferRouter.sol	0	248
contracts/order/ExecuteOrderUtils.sol	0	148
contracts/position/DecreasePositionCollateralUtils.sol	0	726
contracts/position/DecreasePositionUtils.sol	8	407

Files in Scope 6 / 6.

FILE	NSLOC	TOTAL
contracts/position/Position.sol	7	262
contracts/position/PositionEventUtils.sol	2	296
contracts/position/PositionStoreUtils.sol	11	338
contracts/position/PositionUtils.sol	59	891
contracts/reader/GlvReader.sol	0	204
contracts/reader/Reader.sol	0	248
contracts/reader/ReaderDepositUtils.sol	0	176
contracts/reader/ReaderPositionUtils.sol	0	254
contracts/reader/ReaderPricingUtils.sol	0	279
contracts/reader/ReaderUtils.sol	0	253
contracts/router/relay/BaseGelatoRelayRouter.sol	0	444
contracts/router/relay/EIP6492Deployer.sol	0	99
contracts/router/relay/IRelayUtils.sol	0	130
contracts/router/relay/MultichainRelayUtils.sol	0	382
contracts/router/relay/RelayUtils.sol	0	476
contracts/router/relay/SignatureUtils.sol	0	181
contracts/router/relay/SubaccountRouterUtils.sol	0	86
contracts/swap/SwapHandler.sol	0	33
contracts/swap/SwapUtils.sol	0	476
REMEDIATION REVIEW 4		
contracts/config/TimelockConfig.sol	37	641
contracts/data/Keys2.sol	5	202
contracts/error/Errors.sol	2	615
contracts/fee/FeeDistributor.sol	42	719
contracts/fee/FeeDistributorUtils.sol	103	271
contracts/multichain/IMultichainProvider.sol	7	22
contracts/multichain/LayerZeroProvider.sol	32	495
contracts/multichain/MultichainEventUtils.sol	35	150
REMEDIATION REVIEW 5		
contracts/fee/FeeDistributor.sol	0	718

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
H-01	Pending GLV deposits bypass delisting	Validation	HIGH	RESOLVED	p. 17
H-02	Swaps bypass trader P&L payout cap	Business logic errors	HIGH	RESOLVED	p. 18
H-03	Shared deployer enables wallet squatting	Authentication & Session flaws	HIGH	RESOLVED	p. 19
H-04	Swaps can stepwise reprice capped GM	Business logic errors	HIGH	RESOLVED	p. 20
H-05	Stake migration duplicates weekly rewards	Business logic errors	HIGH	ACKNOWLEDGED	p. 21
H-06	Unsynchronized snapshots stall rewards	Business logic errors	HIGH	RESOLVED	p. 22
H-07	Late buybacks misallocate rewards	Business logic errors	HIGH	RESOLVED	p. 23
H-08	Staking transfer approval misses rewards	Business logic errors	HIGH	RESOLVED	p. 24
M-01	Staking hashes omit the signing account	Signatures	MEDIUM	RESOLVED	p. 25
M-02	Account wallets cannot cancel vesting	Business logic errors	MEDIUM	RESOLVED	p. 27
M-03	Dust deposit causes extreme borrowing fees	Math	MEDIUM	RESOLVED	p. 29
M-04	Bridge-fee swap bypasses atomic size cap	Validation	MEDIUM	RESOLVED	p. 31
M-05	Bootstrap timelock roles remain active	Access Control	MEDIUM	ACKNOWLEDGED	p. 32
M-06	Stripped EIP-6492 signatures change account	Signatures	MEDIUM	RESOLVED	p. 34
M-07	Self-swaps bypass USD deposit caps	Validation	MEDIUM	RESOLVED	p. 35
M-08	Funding ramp config skips settlement	Configuration	MEDIUM	RESOLVED	p. 36
M-09	Kink borrowing exceeds configured maximum	Math	MEDIUM	RESOLVED	p. 37
M-10	Empty-side PnL hides distressed GM	Validation	MEDIUM	ACKNOWLEDGED	p. 38
M-11	Atomic withdrawals accept stale recovery prices	Oracle	MEDIUM	ACKNOWLEDGED	p. 39
M-12	Partial closes repeatedly refresh P&L cap	Business logic errors	MEDIUM	RESOLVED	p. 40

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
M-13	ECDSA-first bypasses EIP-7702 revocation	Signatures	MEDIUM	ACKNOWLEDGED	p. 41
M-14	Liquidation impact caps net incorrectly	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 42
M-15	Impact pool understates borrowing usage	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 43
M-16	Spread updates reprice cached reports	Oracle	MEDIUM	ACKNOWLEDGED	p. 44
M-17	Rate cuts let GLV overredeem assets	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 45
M-18	Aged positions drain impact inventory	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 46
M-19	Adjacent distributions strand GMX emissions	Business logic errors	MEDIUM	RESOLVED	p. 47
M-20	Cross-group shifts bypass virtual impact	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 48
M-21	Signed orders do not bind UI fee rate	Signatures	MEDIUM	ACKNOWLEDGED	p. 49
M-22	Weekly emissions bypass RewardDistributor	Business logic errors	MEDIUM	RESOLVED	p. 50
M-23	Mixed-time reports bypass bridge-fee cap	Oracle	MEDIUM	RESOLVED	p. 51
M-24	Payout caps can underfund impact reserve	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 53
M-25	Late GMX claims divert remote rewards	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 54
M-26	Funding mode switch reverses stale accrual	Configuration	MEDIUM	RESOLVED	p. 55
L-01	GLV swap flag does not block execution	Business logic errors	LOW	RESOLVED	p. 56
L-02	Fee swaps lack a dedicated pause	Validation	LOW	RESOLVED	p. 58
L-03	Bridge-fee swaps lack slippage protection	Validation	LOW	RESOLVED	p. 60
L-04	Zero-pool state blocks collateral top-ups	Business logic errors	LOW	RESOLVED	p. 62
L-05	Zero-pool market blocks healthy GLV shifts	DoS	LOW	RESOLVED	p. 64
L-06	Recovery executions lack usable reader quotes	Business logic errors	LOW	RESOLVED	p. 66
L-07	Zero-pool periods waive borrowing fees	Business logic errors	LOW	ACKNOWLEDGED	p. 68
L-08	Subaccount approvals replay across wallets	Signatures	LOW	RESOLVED	p. 70
L-09	Collateral caps block risk reductions	DoS	LOW	RESOLVED	p. 71
L-10	Staking signatures permit sibling-wallet reuse	Signatures	LOW	RESOLVED	p. 72

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
L-11	Data Stream depth is not bound to notional	Oracle	LOW	ACKNOWLEDGED	p. 74
L-12	Atomic withdrawal arbitrages oracle providers	Oracle	LOW	ACKNOWLEDGED	p. 75
L-13	Mixed fee vault migrations stall epochs	Business logic errors	LOW	ACKNOWLEDGED	p. 76
L-14	Direct GLV price bypasses negative-market check	Oracle	LOW	ACKNOWLEDGED	p. 77
L-15	Negative GLV removal rewards late deposits	Business logic errors	LOW	ACKNOWLEDGED	p. 78
L-16	Swaps bypass open-interest reserve buffer	Validation	LOW	ACKNOWLEDGED	p. 79
L-17	Disabled markets accrue trapped borrowing debt	Business logic errors	LOW	ACKNOWLEDGED	p. 80
L-18	Depeg midpoint overpays positive swap impact	Math	LOW	ACKNOWLEDGED	p. 81
L-19	Smaller-side exemption forgives stale borrowing	Configuration	LOW	ACKNOWLEDGED	p. 82
L-20	Gross swap impact ignores fee-net pool credit	Math	LOW	ACKNOWLEDGED	p. 83
L-21	GLV bypasses reserve withdrawal lock	Business logic errors	LOW	ACKNOWLEDGED	p. 84
L-22	Impact-pool withdrawals enable sandwiches	MEV	LOW	ACKNOWLEDGED	p. 85
L-23	Dust improvement discounts full deposit	Business logic errors	LOW	ACKNOWLEDGED	p. 86
L-24	Paused source strands peer fee allocation	Configuration	LOW	ACKNOWLEDGED	p. 87
L-25	Funding debts overstate loss offsets	Business logic errors	LOW	ACKNOWLEDGED	p. 88
L-26	PnL-cap changes reprice GLV shares	Configuration	LOW	ACKNOWLEDGED	p. 89
L-27	Cross-token lending can halt liquidations	DoS	LOW	ACKNOWLEDGED	p. 90
L-28	Stable price bands discount funding	Oracle	LOW	ACKNOWLEDGED	p. 91
L-29	Relay approvals bypass disabled integrations	Access Control	LOW	ACKNOWLEDGED	p. 92
L-30	ADL ignores collectible borrowing fees	Business logic errors	LOW	ACKNOWLEDGED	p. 93
L-31	Bridge signatures lack fee cap	Validation	LOW	ACKNOWLEDGED	p. 94
I-01	CREATE2 wallets are build-coupled	Upgradeability	INFORMATIONAL	RESOLVED	p. 95
I-02	Min-size update promotes old order	Unexpected Behavior	INFORMATIONAL	ACKNOWLEDGED	p. 97
I-03	Transient OI bypasses collateral buffer	Unexpected Behavior	INFORMATIONAL	ACKNOWLEDGED	p. 98
I-04	Delegated resize preserves non-owner payout	Access Control	INFORMATIONAL	ACKNOWLEDGED	p. 99

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
REMEDIATION REVIEW 1					
M-01	Swap restoration backcharges borrowing fees	Logical Error	MEDIUM	RESOLVED	p. 100
M-02	USD restore inflates reconstructed PnL cap	Logical Error	MEDIUM	RESOLVED	p. 101
M-03	Historical reads survive remote reset	Business logic errors	MEDIUM	RESOLVED	p. 103
M-04	Late completion consumes the next epoch	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 104
L-01	Old snapshots are overwritten too early	Warning	LOW	ACKNOWLEDGED	p. 105
L-02	Swap fix depends on PnL cap ordering	Configuration	LOW	RESOLVED	p. 106
L-03	Funding bounds remain cadence-dependent	Math	LOW	ACKNOWLEDGED	p. 107
L-04	Staking roles absent from role policy	Configuration	LOW	ACKNOWLEDGED	p. 108
L-05	Stale delegation overrides newer choice	Signatures	LOW	ACKNOWLEDGED	p. 109
L-06	Zero stake blocks treasury fee mode	Math	LOW	ACKNOWLEDGED	p. 110
L-07	Failed withdrawal can close later vesting	Signatures	LOW	ACKNOWLEDGED	p. 111
L-08	Fee sweep misclassifies buyback GMX	Business logic errors	LOW	ACKNOWLEDGED	p. 112
L-09	Bridge fee swaps are underpriced	Business logic errors	LOW	RESOLVED	p. 113
L-10	Legacy bridge payloads cannot be decoded	Business logic errors	LOW	ACKNOWLEDGED	p. 114
L-11	Delegated updates divert execution fee escrow	Business logic errors	LOW	ACKNOWLEDGED	p. 115
L-12	Market switchover can report false success	Configuration	LOW	ACKNOWLEDGED	p. 117
L-13	Config updater can undo oracle controls	Configuration	LOW	ACKNOWLEDGED	p. 118
I-01	Read delay consumes bridge grace period	Business logic errors	INFORMATIONAL	ACKNOWLEDGED	p. 119
REMEDIATION REVIEW 2					
M-01	Historical reads survive remote reset	Business logic errors	MEDIUM	RESOLVED	p. 120
M-02	Live cap reprices forgone PnL history	Logical Error	MEDIUM	RESOLVED	p. 121
M-03	V1 migration strands staked GLP	Business logic errors	MEDIUM	RESOLVED	p. 122
M-04	Compose reentrancy consumes victim actions	Frontrunning	MEDIUM	RESOLVED	p. 124
M-05	Migration disables multichain staking wallets	Business logic errors	MEDIUM	ACKNOWLEDGED	p. 125

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
M-06	Vault handler bypasses esGMX transfer limits	Logical Error	MEDIUM	RESOLVED	p. 126
L-01	Active wallets reject V1 position imports	Compatibility	LOW	RESOLVED	p. 127
L-02	Staked GLP rewards can block V1 receivers	DoS	LOW	ACKNOWLEDGED	p. 128
L-03	sGLP sweep traps stake on the vault	Unexpected Behavior	LOW	ACKNOWLEDGED	p. 129
I-01	Unused SafeERC20 import and declaration	Superfluous Code	INFORMATIONAL	ACKNOWLEDGED	p. 130
REMEDIATION REVIEW 3					
M-01	Donations checkpoint stale borrowing rates	Configuration	MEDIUM	RESOLVED	p. 131
L-01	OFT dust can block a fee distribution	Business logic errors	LOW	RESOLVED	p. 133
L-02	Refunded compose can consume later bridge funds	Logical Error	LOW	RESOLVED	p. 134
I-01	Verify capped PnL history for positions retained across the executor upgrade	Upgradeability	INFORMATIONAL	ACKNOWLEDGED	p. 136
REMEDIATION REVIEW 4					
I-01	Unused Cast.sol import in FeeDistributor	Superfluous Code	INFORMATIONAL	RESOLVED	p. 137

HIGH **H-01** RESOLVED

Pending GLV deposits bypass delisting

CATEGORY	Validation
LOCATION	contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol:63-114,177-216; contracts/glv/GlvUtils.sol:91-122,299-315,383-408
REVIEW	Main Review

DESCRIPTION

GLV membership is checked at request creation but not execution. A pending GM deposit remains in GlvVault and is therefore absent from the GlvToken balance checked during removal. After the market is disabled and removed, execution still accepts the globally enabled GM market, transfers its GM into GlvToken, values it for share minting, and passes the zero-cap early return. Subsequent GLV NAV excludes that removed market because valuation iterates only the supported-market set.
Impact: A pre-delisting depositor can mint a dominant share of GLV supply against GM that is immediately excluded from GLV NAV. Those shares can redeem supported healthy constituents, spreading a quarantined market's loss into the remaining GLV and draining an arbitrarily large fraction of its healthy backing as the stale deposit is scaled.

RECOMMENDATION

At execution, revalidate that the request market remains a supported and enabled constituent with GlvUtils.validateGlvMarket(dataStore, glv, market, true) before transferring GM or minting shares. As defense in depth, account for pending deposits during removal or explicitly cancel them.

PROOF OF CONCEPT https://sentry.guardianaudits.com/shared/findings/PHccPGOBkG87?findingId=019f888e-1775-7731-8fab-b496ea8ac73e

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/commit/815843bf1f84ebf646bc9ce99e5c9c4b27f800af

HIGH **H-02** RESOLVED

Swaps bypass trader P&L payout cap

CATEGORY	Business logic errors
LOCATION	`contracts/swap/SwapUtils.sol:358-408; contracts/market/MarketUtils.sol:3392-3473; trader P&L cap calculation paths`
REVIEW	Main Review

DESCRIPTION

The trader P&L cap is intended to limit withdrawals from a market side whose PnL-to-pool factor is breached. validateMaxPnlForSwap nevertheless permits a swap that merely improves the breached factor, even when the factor remains above its configured limit. A trader with capped positive PnL can swap assets into the breached side, increase its pool-value denominator, and then close the profitable position under a larger cap.

This perturbs the cap mechanism itself: temporary swap liquidity changes the amount of accrued trader PnL that may be released, even though the market remains in the breached state that the cap is intended to protect. The resulting excess payout is taken from LP backing. Whether the swap capital is later reversed is not necessary to the core violation; it only demonstrates that the liquidity was not durable recovery capital.

RECOMMENDATION

Do not permit swaps to release additional trader PnL while the relevant PnL-to-pool factor remains breached. Require post-swap compliance before the increased denominator can support a larger trader payout, or exclude swap-induced pool balance changes from the trader-PnL cap until they are durable under a defined lock or recovery condition. Add tests showing that a swap which improves but does not clear a breached factor cannot increase a profitable trader's capped payout.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/197`

HIGH **H-03** RESOLVED

Shared deployer enables wallet squatting

CATEGORY	Authentication & Session flaws
LOCATION	`contracts/router/relay/EIP6492Deployer.sol:10-14; contracts/router/relay/SignatureUtils.sol:158-187; contracts/router/relay/BaseGelatoRelayRouter.sol:426-447`
REVIEW	Main Review

DESCRIPTION

EIP6492Deployer is permissionless and executes arbitrary factory calls from one shared caller address. Caller-namespaced CREATE3 factories can derive a counterfactual wallet address from that shared deployer and a salt while allowing the wallet bytecode to vary. An attacker who learns a victim's intended salt can call EIP6492Deployer first with attacker-controlled ERC-1271 wallet code and permanently occupy the victim's predicted address.

When the legitimate relay later validates the counterfactual account, SignatureUtils observes code at the expected address and validates through the attacker-controlled ERC-1271 implementation. The signed wrapper hash constrains the legitimate relay request but does not prevent the independent, permissionless pre-deployment. This can let the attacker approve forged relay actions for the squatted identity and take prefunded multichain-vault balances.

RECOMMENDATION

Add the role-store-backed onlyController check to EIP6492Deployer.deploy so only trusted controller and relay-validation contracts can invoke factory calls. Relay validation already invokes the deployer from the router context after the EIP-6492 wrapper hash has bound the factory calldata to the signed request, while the factory call continues to execute from the unprivileged deployer address. Ensure every deployer call path is controller-gated, avoid granting roles to the deployer itself, and add a regression test proving an external caller cannot predeploy at a victim's predicted address.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/234`

HIGH **H-04** RESOLVED

Swaps can stepwise reprice capped GM

CATEGORY	Business logic errors
LOCATION	`contracts/swap/SwapUtils.sol:360-408; contracts/market/MarketUtils.sol:350-408, 3388-3425, 3431-3473`
REVIEW	Main Review

DESCRIPTION

When a directional trader-PnL cap is binding, a public swap can increase the relevant directional pool balance without adding equivalent net market value. This lowers that side's PnL-to-pool factor below the cap, causing previously capped trader profit to become recognized as a payable liability. The GM market's pool value, and therefore its GM token price, can decrease stepwise as a direct consequence of the swap. This is distinct from a trader simply bypassing its own PnL cap. A party can intentionally trigger the GM repricing and create an externally observable collateral-value shock. Lending protocols that accept the GM token as collateral can face induced liquidations or undercollateralization around the discontinuous price move. The actor can also trigger the repricing and then acquire or deposit GM at the lower valuation, capturing a pricing transition created by a permissionless swap.

RECOMMENDATION

Do not allow a swap to move a market from a breached PnL-cap state to an unbreached state while immediately repricing capped trader liabilities. Preserve the pre-swap capped-liability treatment through a settlement or cooldown window, or require a controlled recapitalization/ADL process before uncapping. Document the GM price discontinuity and avoid treating the GM token as manipulation-resistant collateral until this behavior is mitigated.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/247`

HIGH **H-05** ACKNOWLEDGED

Stake migration duplicates weekly rewards

CATEGORY	Business logic errors
LOCATION	`contracts/fee/FeeDistributor.sol: initiateDistribute(), processLzReceive(), distribute(); contracts/multichain/MultichainStakingRouter.sol: stakeGmx(), unstakeGmx()`
REVIEW	Main Review

DESCRIPTION

Release 2.2.1 combines fixed per-chain fee allocations with live reward-tracker balances and adds permissionless multichain staking. `initiateDistribute` snapshots each chain's tracker supply, and the distribution later assigns each chain a fixed GMX allocation from those snapshots. However, `distribute` starts each tracker's reward stream against its current supply.

A staker can leave a small amount staked on the source chain after both snapshots, move the remaining GMX through the new multichain staking flow, and stake it on the destination chain before the weekly streams begin. The source dust receives a share of the source chain's fixed allocation while the migrated principal receives a share of the destination allocation. The same time-weighted stake can therefore earn from both allocations in one epoch, diluting honest stakers. The window is created by the asynchronous LZ read / bridging lifecycle and there is no FeeDistributor state check in stakeGmx or unstakeGmx.

The snapshot-versus-live accounting existed before, but this PR introduces the normal cross-chain staking path that makes the migration executable by users.

RECOMMENDATION

Bind eligibility to the same epoch snapshot used to calculate allocations. For example, record per-account source and destination stake at the read epoch and distribute only against that checkpointed balance, or make a stake moved after the snapshot ineligible for both source and destination streams for that epoch. At minimum, freeze cross-chain staking changes between the snapshots and the start of all weekly reward streams.

RESOLUTION

Status: Acknowledged.

HIGH **H-06** RESOLVED

Unsynchronized snapshots stall rewards

CATEGORY	Business logic errors
LOCATION	`contracts/fee/FeeDistributor.sol:85-152; contracts/multichain/MultichainReader.sol:207-231`
REVIEW	Main Review

DESCRIPTION

The weekly cross-chain GMX distribution has no globally enforced snapshot boundary. Each chain independently calls initiateDistribute in a separate transaction, stores its local fee balance and tracker supply, and sends LayerZero read requests using that transaction's block timestamp. The MultichainReader uses the request timestamp for its remote reads, but each chain's request is created at a different time.

A user can perform ordinary permissionless actions—such as moving GMX stake between chains or creating fee proceeds—between those initiation transactions. The resulting distributors then finalize different views of the same epoch: one may use the pre-move allocation while another uses the post-move allocation. Each chain independently calculates its required local GMX pot and whether it has surplus to bridge from those inconsistent inputs.

Those funding plans can be mutually incompatible. In the failure case, multiple chains can each wait for inbound GMX while no chain classifies itself as the necessary surplus sender, preventing the weekly reward streams from starting. This is a race condition in normal user interaction, not a need for malicious keeper timing: coordinated keeper calls only make the snapshot times close and do not guarantee identical state.

RECOMMENDATION

Use an enforceable shared epoch snapshot rather than independent per-chain initiation timestamps. For example, commit a single epoch reference and snapshot stake and fee state against it, or lock/meter stake and fee changes across the distribution snapshot window. Validate that the aggregate cross-chain funding plan is conserved before accepting it, and provide a deterministic reconciliation path for any failed epoch.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/248`

HIGH **H-07** RESOLVED

Late buybacks misallocate rewards

CATEGORY	Business logic errors
LOCATION	`contracts/fee/FeeDistributor.sol:106-112, 163-225, 258-277; contracts/fee/FeeHandler.sol:181-207`
REVIEW	Main Review

DESCRIPTION

The weekly distribution snapshots each chain's withdrawable GMX and FeeDistributorVault balance in initiateDistribute. However, a permissionless FeeHandler buyback immediately increments the withdrawable GMX amount after that snapshot. When the LayerZero read response later reaches processLzReceive, the distributor calls withdrawFees(GMX), which pulls the then-current withdrawable GMX rather than the snapshotted amount.

Consequently, GMX from a buyback performed during the read window is swept into the current distribution cycle even though the cross-chain allocation was calculated from the earlier global snapshot. On a deficit chain, bridgedGmxReceived subsequently uses the live FeeDistributorVault balance to determine whether funding is sufficient, reclassifying the late proceeds as local current-epoch rewards instead of carrying them forward or allocating them across all chains.

A local staker can time permissionless buybacks after initiation and before the read response, acquire fee-token inventory, and recover an outsized portion of the GMX through that chain's reward stream. The mismatch is repeatable while fee inventory is available and misallocates reward proceeds away from the global staker set.

RECOMMENDATION

Bind the current epoch to the amount recorded at initiation. Withdraw only the snapshotted GMX during processLzReceive and record any newly withdrawable proceeds as next-epoch carry. Alternatively, cap each chain's current reward stream at its authenticated allocation and exclude post-snapshot balances from its funding check. Ensure all post-snapshot buyback proceeds enter the next shared distribution round.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/243`

HIGH **H-08** RESOLVED

Staking transfer approval misses rewards

CATEGORY	Business logic errors
LOCATION	`contracts/multichain/MultichainStakingUtils.sol:169-178; legacy RewardRouterV2.acceptTransfer`
REVIEW	Main Review

DESCRIPTION

In strict-transfer mode, MultichainStakingUtils.signalStakingTransfer approves the recipient account wallet for only the sender wallet's current fee-tracker staked amount, then signals the legacy V1 transfer. The later acceptance flow in RewardRouterV2 compounds the sender before it determines and unwinds the stake being migrated.

Compounding claims accrued GMX fee rewards and esGMX rewards, restakes them through the tracker stack, and increases the sender's fee-tracker stake after the approval amount was fixed. The transfer path can therefore require a larger allowance than the allowance established at signal time. Acceptance reverts on the shortfall, leaving the transfer pending and preventing the account from completing its multichain staking transfer.

This condition can arise from ordinary reward accrual and does not require a privileged action or compromised party. The release adds the account-wallet signal/accept flow that establishes the exact approval before delegating to the legacy router.

RECOMMENDATION

Do not rely on a pre-compounding exact allowance. Refresh the approval after any compounding that can change the tracker balance, or approve a safely bounded maximum for the destination wallet and revoke/reset it after successful acceptance. Prefer an atomic account-wallet flow that signals and accepts with the required allowance derived from the post-compound balance.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/238`

MEDIUM **M-01** RESOLVED

Staking hashes omit the signing account

CATEGORY	Signatures
LOCATION	`contracts/router/relay/StakingUtils.sol:8-129; contracts/multichain/MultichainStakingRouter.sol:25-143; contracts/router/relay/BaseGelatoRelayRouter.sol:426-466`
REVIEW	Main Review

DESCRIPTION

All eleven EIP-712 struct hashes introduced for MultichainStakingRouter omit the `account` whose signature is validated. The router receives `account` as a separate argument, computes a staking struct hash using only the action parameters and RelayParams, and then passes the unbound hash and account separately to `_validateCall`.

Because the replay-protection mapping is global by digest, two accounts using identical action parameters and RelayParams produce the same digest. An attacker can copy a victim's pending staking payload, replace only the separately supplied account and signature with their own, and execute the same action against the attacker's account. The attacker's valid call marks the shared digest as used. The victim's original transaction then reverts with `InvalidUserDigest` and must be re-signed with a new nonce.

`eip6492SignatureWrapperHash` does not provide account binding. EOAs normally share the zero value, and an attacker can also copy a victim's nonzero wrapper hash while supplying an ordinary EOA signature, for which SignatureUtils ignores the wrapper hash.

This affects stakeGmx, unstakeGmx, stakeEsGmx, unstakeEsGmx, handleStakingRewards, compoundStakingRewards, vestEsGmx, delegateGovGmx, signalStakingTransfer, acceptStakingTransfer, and withdrawFromWallet. It recreates the account-unbound digest collision fixed for the other gasless action families in commit 50d1c018a.

RECOMMENDATION

Add `address account` to every staking EIP-712 type definition and include it in every corresponding `abi.encode` struct hash. Pass `account` from MultichainStakingRouter into each StakingUtils hash helper. Update the off-chain typed-data definitions/signing helpers accordingly.

Add regression tests proving that otherwise-identical staking calls for two accounts produce different struct hashes and digests, and that one account cannot consume another account's digest.

M-01 Staking hashes omit the signing account CONTINUED

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/183

MEDIUM **M-02** RESOLVED

Account wallets cannot cancel vesting

CATEGORY	Business logic errors
LOCATION	`contracts/multichain/MultichainStakingUtils.sol:137-150,190-200; contracts/multichain/MultichainStakingRouter.sol:90-98,134-143; contracts/staking/GmxAccountWallet.sol:11-31; contracts/v1/IVesterV1.sol:5-8`
REVIEW	Main Review

DESCRIPTION

`MultichainStakingUtils.vestEsGmx` transfers the user's esGMX to their deterministic `GmxAccountWallet` and makes that wallet call `gmxVester.deposit(amount)`. The V1 vester therefore records the contract wallet, rather than the signing EOA, as the vesting account.

However, `GmxAccountWallet.execute` is restricted to the global CONTROLLER role, and neither `MultichainStakingRouter`, its interface, the EIP-712 staking actions, nor LayerZero dispatch exposes an operation that makes the wallet call `Vester.withdraw()`. `withdrawFromWallet` is not a substitute: it only invokes ERC20 `transfer`, while production Vester receipt tokens are non-transferable. Relay external calls originate from unprivileged `ExternalHandler` and cannot call the controller-gated wallet.

Users can claim GMX as it vests through `handleRewards`, but cannot pause vesting, recover unvested esGMX, or release the paired staking tokens. Even after the 365-day vest completes and GMX is claimed, the pair amount remains reserved until `withdraw()` executes. Recovery requires governance intervention to introduce or authorize a controller call path. The router is currently skipped on mainnet, so impact begins when this feature is enabled against the live V1 contracts.

RECOMMENDATION

Add a signed `withdrawVesting` / `cancelVestEsGmx` action across the full staking relay stack. It should make the user's account wallet call `IVester(gmxVester).withdraw()`, then measure and sweep any GMX and esGMX balance increases from the wallet into the user's multichain balance. The returned paired tracker token should remain in the wallet because it represents the released staking position.

Add `withdraw()` to `IVesterV1`, define and account-bind the new EIP-712 action, expose it through `IMultichainStakingRouter` and `MultichainStakingRouter`, add LayerZero dispatch and off-chain signing helpers.

RESOLUTION

Status: Resolved.

M-02 Account wallets cannot cancel vesting CONTINUED

https://github.com/gmx-io/gmx-synthetics-private/pull/188

MEDIUM M-03 RESOLVED

Dust deposit causes extreme borrowing fees

CATEGORY	Math
LOCATION	`contracts/market/MarketUtils.sol:620-632,2945-3028,3031-3069,3148-3176; contracts/deposit/ExecuteDepositUtils.sol:126-139,499-524`
REVIEW	Main Review

DESCRIPTION

When a market side has nonzero reserved USD but `poolUsd == 0`, recovery paths may pass `allowZeroPoolBorrowingFactor = true`, causing `getBorrowingFactorPerSecond` to return zero. A market deposit is then permitted to add liquidity without requiring the post-deposit state to satisfy `validateReserve` or `validateOpenInterestReserve`.

This creates a dangerous discontinuity. At exactly zero pool USD the rate is zero, but after a dust deposit makes `poolUsd` nonzero, normal utilization math resumes. The legacy model computes a value proportional to `reservedUsd^exponent / poolUsd`; the kink model likewise derives an uncapped `usageFactor` from `reservedUsd / (poolUsd * reserveFactor)` and extrapolates above 100% utilization. The configured borrowing coefficients are range-limited, but the final runtime rate is not capped.

A reproduced example sets \$5,000 of reserved USD on a depleted side and executes a valid deposit of one WNT wei. The borrowing rate changes from zero to `2e37`. With 1e30 factor precision, this equals 20 million times position notional per second. An attacker can permissionlessly create this dust deposit; a deposit keeper must execute the otherwise-valid request.

Beginning with the next second/block, remaining positions can be charged extreme borrowing fees on the next state update. Collection is bounded by available position assets: ordinary decreases may revert with `InsufficientFundsToPayForCosts`, while a full liquidation/ADL may consume available assets and discard the unpaid remainder. However, `getTotalPendingBorrowingFees` values aggregate fees without capping them by collectible collateral, so GM/GLV valuation may include economically uncollectible fees. Extreme rates can also cause arithmetic reverts and recreate the recovery deadlock this change intends to solve.

The issue applies when the depleted side is subject to borrowing; configurations that skip borrowing for the smaller side may return zero earlier.

RECOMMENDATION

M-03 Dust deposit causes extreme borrowing fees CONTINUED

Introduce an explicit per-market, per-side `MAX_BORROWING_FACTOR_PER_SECOND` and clamp the final computed borrowing rate after both the legacy and kink calculations. Constrain this setting with a protocol-wide upper bound in `ConfigUtils`, analogous to `MAX_FUNDING_FACTOR_PER_SECOND`.

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/189

MEDIUM **M-04** RESOLVED

Bridge-fee swap bypasses atomic size cap

CATEGORY	Validation
LOCATION	contracts/multichain/MultichainTransferRouter.sol:63-79,186-248; contracts/router/relay/BaseGelatoRelayRouter.sol:321-328
REVIEW	Main Review

DESCRIPTION

A signed cross-chain bridgeOut may specify an arbitrary non-WNT bridgeFee.feeAmount and route it through SwapHandler using AtomicSwap pricing. The entire output is first credited to the user's MultichainVault balance, after which LayerZeroProvider deducts only the actual messaging fee. Unlike the equivalent relay-fee conversion, this path does not enforce MAX_RELAY_FEE_SWAP_USD, and feeAmount is not bounded by or tied to the provider's live fee quote. The PoC executes the public signed path and converts \$1 million despite a configured \$100 cap while only 0.001 WNT is consumed as the actual bridge fee.

Impact: Any multichain user can turn a minimal bridge-out into an arbitrarily large atomic swap, bounded only by their credit and pool liquidity. This defeats the protocol's explicit size control for limiting oracle-mispricing extraction and materially increases LP exposure during honest oracle lag or adverse atomic-swap configuration. The PoC proves the cap bypass and general-purpose conversion, but not High-severity profit under sound prices.

RECOMMENDATION

Apply MAX_RELAY_FEE_SWAP_USD to bridge-fee conversions through shared validation. Additionally cap signed feeAmount or resulting WNT to a conservative bound derived from the provider's quoted messaging fee, and consider a dedicated emergency feature flag for these internal atomic swaps.

PROOF OF CONCEPT https://sentry.guardianaudits.com/shared/findings/PHccPGOBkG87?findingId=019f8b7a-a518-761b-a333-80fc172c86c6

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/commit/64bf5f1eaa965caf89b151e446e6e34728c38eff + https://github.com/gmx-io/gmx-synthetics-private/commit/f65caafbce2a081b487b48ec07fe3369b3f88c1d

MEDIUM M-05 ACKNOWLEDGED

Bootstrap timelock roles remain active

CATEGORY	Access Control
LOCATION	contracts/gov/GovTimelockController.sol:7-17; contracts/gov/ProtocolGovernor.sol
REVIEW	Main Review

DESCRIPTION

The Arbitrum GovTimelockController linked from ProtocolGovernor retains PROPOSER_ROLE, CANCELLER_ROLE, and EXECUTOR_ROLE for the bootstrap deployer after finalization. Only TIMELOCK_ADMIN_ROLE is revoked. The retained address is an EOA and the live Arbitrum role configuration confirms these operational roles remain active.
A holder of this key can queue and execute timelock operations after the configured delay without going through ProtocolGovernor's token-holder proposal and vote path, and can cancel legitimate queued governance operations. Whether this is exploitable depends on whether the bootstrap key is intentionally retained as a permanent governance superuser; absent that intent, it bypasses the designed token-governance controls.

RECOMMENDATION

During finalization, revoke the bootstrap deployer's PROPOSER_ROLE, CANCELLER_ROLE, and EXECUTOR_ROLE in addition to TIMELOCK_ADMIN_ROLE. Add deployment assertions and monitoring that enumerate all timelock role members, and document any deliberately retained emergency authority.

RESOLUTION

Status: Acknowledged.

Guardian: The Team 2 M-01 remediation revokes the bootstrap deployer's proposer, canceller, and executor roles during a fresh timelock finalization, but only while that deployer still has `TIMELOCK_ADMIN_ROLE`:

```
if (await govTimelockController.hasRole(TIMELOCK_ADMIN_ROLE, deployer)) {
  await revokeRoleIfPresent(PROPOSER_ROLE, deployer);
  await revokeRoleIfPresent(CANCELLER_ROLE, deployer);
  await revokeRoleIfPresent(EXECUTOR_ROLE, deployer);
  await revokeRoleIfPresent(TIMELOCK_ADMIN_ROLE, deployer);
} else {
  console.info("skipping govTimelockController role config");
}
```

M-05 Bootstrap timelock roles remain active CONTINUED

The previous finalization already removed the bootstrap EOA's admin role without removing its operational roles. The active Arbitrum timelock therefore takes the skip branch: the bootstrap EOA has no admin role but still has proposer, canceller, and executor. The timelock and governor deployments use fixed deployment IDs and are unchanged by this remediation, so deploying the PR does not replace the active governance pair.
The bootstrap EOA can consequently continue to schedule and execute operations after the timelock delay without a token holder vote, and can cancel operations submitted through the governor.

Recommendation
Use the current governor to execute one timelock operation that revokes the bootstrap EOA's proposer, canceller, and executor roles on the active timelock. Retain the updated deployment script for new deployments and confirm the final membership with the existing role validator.

`https://github.com/gmx-io/gmx-synthetics-private/pull/235 + https://github.com/gmx-io/gmx-synthetics-private/pull/244`

MEDIUM **M-06** RESOLVED

Stripped EIP-6492 signatures change account

CATEGORY	Signatures
LOCATION	`contracts/router/relay/SignatureUtils.sol:67-123; contracts/router/relay/StakingUtils.sol:95-130; contracts/multichain/MultichainStakingRouter.sol:112-121`
REVIEW	Main Review

DESCRIPTION

Staking action digests do not bind the supplied account address. For counterfactual smart-contract wallets, the user's EOA signature is wrapped using EIP-6492 so the wallet can be deployed and validate it through ERC-1271. SignatureUtils only enforces the signed wrapper commitment when the submitted signature retains the EIP-6492 marker. A relayer can strip the wrapper, supply the owner EOA as the separately supplied account, and pass the same inner ECDSA signature through the raw ECDSA branch.
A relayer receives the signed payload before broadcast and can submit the altered call without a mempool race. The first successful call consumes the global digest. This can redirect authorized multichain payments and staking actions from the intended counterfactual wallet to the owner's EOA, including signal of an amount-unbounded transferable staking position.

RECOMMENDATION

Require an EIP-6492-formatted signature whenever a nonzero wrapper hash is committed, and reject nonzero wrapper hashes in raw ECDSA or ERC-1271 validation. Bind the account address into every staking action typehash and validate the signature before account-relative token or external-call processing. Add regression tests for wrapper stripping and account substitution.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/201`

MEDIUM M-07 RESOLVED

Self-swaps bypass USD deposit caps

CATEGORY	Validation
LOCATION	contracts/deposit/ExecuteDepositUtils.sol:505-524,538-573; contracts/swap/SwapUtils.sol:342-384; contracts/market/MarketUtils.sol:2010-2022
REVIEW	Main Review

DESCRIPTION

A GM deposit may include its destination market in the swap path. The swap adds the original input token to that same market and returns the other pool token, which is then used as the deposit leg input. The deposit flow applies MAX_POOL_USD_FOR_DEPOSIT only to its tokenIn, i.e. the swap output, while the self-swap has already increased the original input token's pool balance.
Swaps enforce the separate raw MAX_POOL_AMOUNT limit for the original token but do not enforce the deposit-specific USD cap. A user can therefore pass the raw token limit while exceeding the governance-configured USD exposure cap for the net-added token. During a depeg, this admits additional toxic inventory and lets the depositor mint GM against it, socializing losses to incumbent liquidity providers.

RECOMMENDATION

Reject the destination market from its own deposit-leg swap path, or snapshot all destination-market pool balances before the composite action and enforce MAX_POOL_USD_FOR_DEPOSIT for every positive net pool delta after the swap and deposit legs. Add regression tests for both token directions, including nonzero fees and price impact.

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/198

MEDIUM **M-08** RESOLVED

Funding ramp config skips settlement

CATEGORY	Configuration
LOCATION	`contracts/config/ConfigSyncer.sol; contracts/market/MarketUtils.sol; funding checkpoint handler paths`
REVIEW	Main Review

DESCRIPTION

ConfigSyncer permits activation of MIN_FUNDING_INCREASE_RATE_PER_SECOND as a raw configuration write without an oracle-priced funding checkpoint. The limited configuration keeper cannot atomically settle funding through the same path: ConfigSyncer has no oracle input or multicall, while settlement-aware configuration paths require different authority or do not accept this key.

If funding is stale when the parameter is activated, the next checkpoint applies the newly configured minimum funding ramp over the entire elapsed interval. This can retroactively change the funding direction and charge the newly designated paying side for time before the configuration update, causing unexpected funding debits and potentially liquidations.

RECOMMENDATION

Require an atomic, oracle-priced funding checkpoint before activating MIN_FUNDING_INCREASE_RATE_PER_SECOND. Alternatively remove this key from ConfigSyncer and route it through a settlement-aware configuration path that enforces checkpointing. Add regression tests for stale funding before activation and verify the new rate applies only after the change.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/205`

MEDIUM M-09 RESOLVED

Kink borrowing exceeds configured maximum

CATEGORY	Math
LOCATION	contracts/position/DecreasePositionCollateralUtils.sol:121-139; contracts/position/DecreasePositionUtils.sol:248-343; contracts/market/MarketUtils.sol:620-631,2061-2082,2945-3069
REVIEW	Main Review

DESCRIPTION

The kink borrowing model is configured to reach its above-optimal borrowing factor at 100% usage. A profitable position decrease pays PnL from pool liquidity but does not apply the open-interest reserve validation that bounds usage during increases. If the decrease leaves reserved open interest above the reduced liquidity denominator, usage exceeds 100%.
The kink formula does not clamp usage at 100%; it linearly extrapolates beyond the configured maximum. This can impose borrowing charges far above the intended maximum on unrelated positions and liquidate otherwise healthy traders at unchanged oracle prices. The issue is triggered by a profitable decrease reducing liquidity, not by an untrusted configuration update.

RECOMMENDATION

Clamp kink usage at 100% so the configured above-optimal borrowing factor remains a true maximum. Also preserve the model's valid reserve domain after profitable decreases by constraining, deferring, or otherwise handling PnL payments that would leave reserved open interest above the supported denominator. Add tests that compare accrued borrowing against the configured maximum after a profitable decrease.

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/206

MEDIUM M-10

ACKNOWLEDGED

Empty-side PnL hides distressed GM

CATEGORY	Validation
LOCATION	`contracts/market/MarketUtils.sol:getPnlToPoolFactor(),getCappedPnl(); GLV admission and valuation paths`
REVIEW	Main Review

DESCRIPTION

When a directional pool side is exactly empty, getPnlToPoolFactor returns zero before accounting for outstanding positive PnL. This makes a distressed market appear safe to market-token deposit and GLV admission checks, while valuation also omits the liability.

An attacker can introduce personally held distressed GM into a GLV that previously had no exposure to that market, receive GLV value that does not reflect the hidden liability, and redeem healthy constituent assets. A small recapitalizing deposit can observe the same pre-addition zero result and unfreeze valuation, enabling the extraction. This transfers the distressed market's loss to existing GLV holders.

RECOMMENDATION

Treat an empty pool side with outstanding positive PnL as saturated or unsafe, not as a zero factor. Read PnL before the zero-denominator shortcut, reject distressed GM admission into GLVs, and ensure recapitalizing deposits cannot rely on the pre-addition zero sentinel. Preserve positive liabilities in GM and GLV valuation before healthy constituent withdrawals.

RESOLUTION

Status: Acknowledged.

MEDIUM M-11

ACKNOWLEDGED

Atomic withdrawals accept stale recovery prices

CATEGORY	Oracle
LOCATION	`contracts/oracle/ChainlinkPriceFeedUtils.sol:17-47; contracts/oracle/ChainlinkPriceFeedProvider.sol:38-72; contracts/oracle/Oracle.sol:113-123,252-329; contracts/exchange/WithdrawalHandler.sol:154-195`
REVIEW	Main Review

DESCRIPTION

After Arbitrum sequencer recovery, ChainlinkPriceFeedUtils permits the underlying Chainlink round under the per-token heartbeat. The provider then replaces the round's updatedAt with block.timestamp. Atomic price validation therefore sees the provider price as fresh even when the actual Chainlink observation predates the recovery and may be much older than the atomic-action age limit.
An attacker can execute an atomic withdrawal using a stale, favorable pre-recovery price before a new Chainlink round is published. This overvalues the withdrawing GM position and shifts the resulting loss to remaining LPs. The issue applies after the configured sequencer grace period because the underlying round timestamp is no longer preserved for the atomic freshness check.

RECOMMENDATION

Preserve the underlying Chainlink round updatedAt for atomic validation. Require an atomic price round to be newer than the sequencer recovery start and within a purpose-specific maximum age. Restrict atomic-provider authorization per token to feeds that can satisfy this tighter freshness requirement, or use a conservative non-atomic path for lower-frequency feeds.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-12** RESOLVED

Partial closes repeatedly refresh P&L cap

CATEGORY	Business logic errors
LOCATION	`contracts/position/DecreasePositionUtils.sol; trader P&L cap and capped-PnL settlement paths`
REVIEW	Main Review

DESCRIPTION

A profitable decrease caps current aggregate PnL against the current directional liquidity, pays the requested capped share, and removes the closed slice's full proportional raw size and token liability. A trader can partition a profitable close into many smaller decreases at the same price. Each slice recalculates the cap against the changed state, allowing a larger cumulative payout than an equivalent one-shot close.

This turns a configured trader P&L cap into a per-slice limit rather than a position-level limit. A trader dominating positive directional PnL can progressively extract more LP liquidity than the cap permits atomically, reducing GM backing and impairing later withdrawals, swaps, reserves, and payouts.

RECOMMENDATION

Make capped PnL settlement cumulative and invariant to position-close partitioning. Track realized capped payouts or retain an equivalent pre-payout baseline so subsequent decreases can receive only unused allowance. Add fixed-price tests comparing one-shot and multi-slice closes for equal total payout.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/204`

MEDIUM M-13

ACKNOWLEDGED

ECDSA-first bypasses EIP-7702 revocation

CATEGORY	Signatures
LOCATION	`contracts/router/relay/SignatureUtils.sol:67-123; relay validation call paths`
REVIEW	Main Review

DESCRIPTION

For non-EIP-6492 raw signatures, SignatureUtils attempts ECDSA recovery before checking whether the expected signer currently has contract code. An EIP-7702 authority's old native ECDSA signature can therefore remain accepted after the account is recovered or rotates its ERC-1271 policy, even when the active account implementation would reject that signature.

A stale or phished relay signature can execute during its validity window against assets that arrive after the recovery or policy update. The issue defeats the account's intended signature revocation semantics because ECDSA validation takes precedence over the current code-bearing account policy.

RECOMMENDATION

When expectedSigner has code, validate raw signatures exclusively through ERC-1271 or the account's equivalent active signature semantics. Use direct ECDSA recovery only for code-less signers, while preserving the distinct EIP-6492 path for counterfactual accounts. Add tests showing that a revoked native signature cannot bypass a code-bearing account's current policy.

RESOLUTION

Status: Acknowledged.

MEDIUM M-14

ACKNOWLEDGED

Liquidation impact caps net incorrectly

CATEGORY	Business logic errors
LOCATION	`contracts/position/PositionUtils.sol:438-490; contracts/position/DecreasePositionCollateralUtils.sol:141-186`
REVIEW	Main Review

DESCRIPTION

Liquidation health accounting applies positive and temporary negative exit-impact caps to fresh impact before adding the position's proportional pending impact. The actual decrease settlement path instead combines fresh and pending impact first, then caps the resulting total.
This mismatch can misstate remaining collateral in either direction. A capped fresh positive credit followed by pending negative impact can make a healthy position appear liquidatable. Conversely, a fresh negative impact capped before a positive pending credit can make liquidation health appear non-negative while the actual decrease settlement still has a negative capped impact. Liquidation eligibility can therefore diverge from the impact that would actually be realized when the position is decreased.

RECOMMENDATION

Use one shared helper for decrease settlement and liquidation health: combine fresh impact with proportional pending impact first, then apply the appropriate positive, negative, and impact-pool caps to the combined total. Add tests for both sign combinations, ensuring liquidation eligibility matches simulated decrease settlement.

RESOLUTION

Status: Acknowledged.

MEDIUM M-15

ACKNOWLEDGED

Impact pool understates borrowing usage

CATEGORY	Business logic errors
LOCATION	contracts/market/MarketUtils.sol:283-303, 410-420, 2945-3028; contracts/position/DecreasePositionCollateralUtils.sol:436-467
REVIEW	Main Review

DESCRIPTION

Negative position impact increases the market's gross poolAmount and the position impact pool by the same token amount. Market valuation through getPoolValueInfo deducts the position impact pool, so GM NAV does not increase. However, getBorrowingFactorPerSecond uses getPoolUsdWithoutPnl, whose denominator is based on gross poolAmount without the corresponding deduction.
As a result, the borrowing model can report lower utilization and charge lower borrowing rates even though the market's GM NAV is unchanged. The effect can move utilization below a configured rate kink and reduce recurring borrowing-fee revenue.

RECOMMENDATION

Define whether position-impact-pool tokens are intended to provide borrowing capacity. If they are economically reserved, deduct the applicable position impact pool from the borrowing liquidity denominator using the same token and price treatment as market valuation. Add rate tests around the kink for changes that move tokens into the impact pool without changing GM NAV.

RESOLUTION

Status: Acknowledged.

MEDIUM M-16

ACKNOWLEDGED

Spread updates reprice cached reports

CATEGORY	Oracle
LOCATION	`contracts/config/Config.sol:362-392; contracts/oracle/ChainlinkDataStreamProvider.sol:61-130; contracts/fee/FeeHandler.sol:105-131, 262-291`
REVIEW	Main Review

DESCRIPTION

ChainlinkDataStreamProvider applies the current DATA_STREAM_SPREAD_REDUCTION_FACTOR when a verified report is executed, without binding that factor to the report's observationsTimestamp. A still-valid report observed under an earlier spread policy can therefore be repriced immediately after a governance spread-reduction update.

FeeHandler.buyback is permissionless and uses the adjusted report prices to determine the maximum fee-token amount. A searcher can retain a pre-update report, submit it in the short post-update validity window, and receive more accrued protocol fee tokens for the same buyback payment than is justified by either the old policy or a fresh post-update report. Repeated buybacks can consume the affected fee inventory before the report expires.

RECOMMENDATION

Version the spread-reduction policy and apply the version active at the report observation time. Alternatively, record an activation time and reject reports observed before it until their validity window expires. Consider valuing fee-token buybacks directly with conservative raw prices so spread-reduction updates cannot remove buyback protection.

RESOLUTION

Status: Acknowledged.

MEDIUM M-17

ACKNOWLEDGED

Rate cuts let GLV overredeem assets

CATEGORY	Business logic errors
LOCATION	`contracts/config/Config.sol:362-392; contracts/market/MarketUtils.sol:2917-2937, 3148-3176; contracts/glv/glvWithdrawal/GlvWithdrawalUtils.sol:138-205, 294-326`
REVIEW	Main Review

DESCRIPTION

Config.setUint can lower a market-side BORROWING_FACTOR without checkpointing cumulative borrowing under the prior rate. getNextCumulativeBorrowingFactor later applies the newly lowered rate across the entire interval since the previous checkpoint, retroactively reducing the market's pending borrowing-fee receivable.

Before an anticipated rate cut, GLV valuation includes the affected constituent's old-rate pending borrowing fees. A GLV holder can withdraw shares into a healthy constituent market and receive physical assets based on that valuation. After the rate update, the affected constituent's stale receivable is recomputed lower, while the healthy assets have already left the GLV. Remaining GLV holders bear the shortfall.

RECOMMENDATION

Treat borrowing-rate changes as prospective by checkpointing each affected market side under the old configuration before storing the new rate. If an oracle-backed checkpoint cannot be performed inline, stage the update and restrict GLV valuation and withdrawal actions until old-rate accrual has settled.

RESOLUTION

Status: Acknowledged.

MEDIUM M-18

ACKNOWLEDGED

Aged positions drain impact inventory

CATEGORY	Business logic errors
LOCATION	contracts/pricing/PositionPricingUtils.sol:248-307; contracts/position/DecreasePositionCollateralUtils.sol:149-170, 426-510; contracts/market/MarketUtils.sol:995-1066, 1099-1117
REVIEW	Main Review

DESCRIPTION

With token-based open-interest balancing, the price-impact curve values a decrease's token-OI change at the current index price, while negative impact settlement is capped using the order's stored sizeDeltaUsd. For an aged full position after substantial appreciation, raw negative impact reflects the larger current token value but the permanent charge is limited by the smaller historical notional; the excess is recorded as claimable collateral.

An attacker can open matched long and short positions when the token is cheap, later close an aged leg, open a current-notional balancing position that earns positive impact under a larger cap, repeat on the other aged leg, and unwind the new position. The attack restores token OI and leaves no attacker positions, while the attacker claims back capped-away negative impact and retains larger positive impact credits paid from the position impact pool.

RECOMMENDATION

When token-OI pricing values a decrease at current token prices, derive its negative-impact ceiling from the same current USD exposure rather than historical stored sizeDeltaUsd. Ensure any closed sequence restoring token OI cannot be net-profitable after claimable-collateral refunds and fees.

RESOLUTION

Status: Acknowledged.

MEDIUM M-19

RESOLVED

Adjacent distributions strand GMX emissions

CATEGORY	Business logic errors
LOCATION	contracts/fee/FeeDistributor.sol:496-505, 563-572
REVIEW	Main Review

DESCRIPTION

FeeDistributor permits one distribution completion per calendar week but does not require the prior seven-day emission schedule to finish. A late completion near a week boundary sets a one-week tokensPerInterval rate. A second valid completion immediately after the next calendar week begins overwrites that rate with the new allocation without carrying forward the prior schedule's funded but unaccrued remainder.
If the new allocation is small or zero, nearly all of the prior weekly GMX allocation can remain in the contract without any continuing schedule for stakers to accrue it. The proof completes two public epochs across a week boundary and shows the second completion, two seconds later, replaces the prior rate while nearly all of the first allocation remains unvested.

RECOMMENDATION

Checkpoint and carry the prior schedule's funded, unaccrued balance into the next emission-rate calculation. Alternatively, require a full week between rate replacements or store explicit schedule start and end times. A zero or small new allocation must not discard an unsettled prior schedule.

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/237

MEDIUM M-20

ACKNOWLEDGED

Cross-group shifts bypass virtual impact

CATEGORY	Business logic errors
LOCATION	contracts/shift/ShiftUtils.sol:110-119, 303-325; contracts/pricing/SwapPricingUtils.sol:159-215
REVIEW	Main Review

DESCRIPTION

A GM shift is permitted whenever the source and destination markets share long and short tokens. During execution, the destination deposit sets includeVirtualInventoryImpact=false on the assumption that the paired source withdrawal and destination deposit net within one virtual-inventory group. Shift creation does not require equal virtualMarketId values.

For different groups, the source withdrawal changes group A while the destination deposit worsens group B's imbalance. The actions do not net, but the destination omits the group-level negative impact that an ordinary deposit would pay. A user can shift into excess destination GM, receive excess destination shares, and redeem those shares against real pool assets, diluting incumbent LPs and leaving the destination virtual group more imbalanced.

This condition exists on live Arbitrum. The enabled ETH/USD \[WETH-USDC\] market at 0x70d95587d40A2caf56bd97485aB3Eec10Bee6336 uses virtual market ID 0xf5134a0a1379cd7f246d7a04d2463c57aa177bf09a34e93dafc5e768c05cea63. The enabled NEAR/USD \[WETH-USDC\] market at 0x63Dc80EE90F26363B3FCD609007CC9e14c8991BE has virtual market ID 0x0000000000000000000000000000000000000000. Both use WETH as long token and USDC as short token, so ShiftUtils permits shifting between them despite the different groups.

RECOMMENDATION

Disable destination virtual-inventory impact only where source and destination have the same nonzero virtualMarketId. For different or zero IDs, execute the destination deposit with includeVirtualInventoryImpact=true, or reject the cross-group shift. Add tests covering enabled markets with identical collateral tokens but different virtual group assignments.

RESOLUTION

Status: Acknowledged.

MEDIUM M-21

ACKNOWLEDGED

Signed orders do not bind UI fee rate

CATEGORY	Signatures
LOCATION	`contracts/router/relay/RelayUtils.sol:225-250; contracts/order/OrderUtils.sol:309-329; contracts/fee/FeeUtils.sol:171-175; contracts/pricing/PositionPricingUtils.sol:515-535`
REVIEW	Main Review

DESCRIPTION

A multichain EIP-712 create-order authorization commits to uiFeeReceiver but neither the receiver's uiFeeFactor nor a user-approved maximum. On the destination chain, OrderUtils reads the receiver's current configured factor when the signed authorization is relayed and snapshots that value on the order. A selected UI can quote a low or zero fee, obtain the trader's signature, raise its factor before relaying, and execute the unchanged valid authorization at the higher rate. Position UI fees are assessed against sizeDeltaUsd, so the maximum 0.1% factor can consume roughly 10% of posted collateral at 100x leverage and increase liquidation risk. The proof signs while the destination UI rate is zero, raises it, then completes real multichain funding, order creation, execution, accrual, and fee claim.

RECOMMENDATION

Include an exact uiFeeFactor or a user-approved maximum factor or fee amount in the signed create-order struct. On relay, reject creation if the current factor or calculated fee exceeds that signed bound, then preserve the accepted factor for execution.

RESOLUTION

Status: Acknowledged.

MEDIUM M-22

RESOLVED

Weekly emissions bypass RewardDistributor

CATEGORY	Business logic errors
LOCATION	`contracts/fee/FeeDistributor.sol:496-504`
REVIEW	Main Review

DESCRIPTION

FeeDistributor resolves rewardTracker.distributor() and programs that distributor's emission rate, but transfers the weekly GMX allocation to RewardTracker instead of RewardDistributor. In the standard GMX V1 integration, RewardTracker credits only the amount returned by RewardDistributor.distribute(), while RewardDistributor is limited by its own GMX balance.
After any pre-existing RewardDistributor inventory is exhausted, scheduled emissions stop accruing to stakers. Permissionless reward updates advance the distributor checkpoint while returning zero, consuming the elapsed reward period. The weekly allocations remain unaccounted for in RewardTracker and require privileged recovery or manual redistribution.

RECOMMENDATION

Transfer weekly feeAmountGmx to the resolved RewardDistributor before setting its emission rate. Validate distributor.rewardToken() is GMX and add an end-to-end V1 RewardTracker/RewardDistributor integration test covering distribution, update, and claim after the distributor's initial balance is exhausted.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/236`

MEDIUM M-23

RESOLVED

Mixed-time reports bypass bridge-fee cap

CATEGORY	Oracle
LOCATION	`contracts/oracle/Oracle.sol: _validatePrices(); contracts/multichain/MultichainTransferRouter.sol: _bridgeOutWithOraclePrices(), _swapBridgeFeeIfNeeded()`
REVIEW	Main Review

DESCRIPTION

The multichain bridge-out path is an atomic action. It accepts caller-supplied reports from any globally approved atomic provider and validates each report's signature and age independently. It only restricts the aggregate timestamp range, so a caller can combine valid observations from different moments into a cross-asset rate that did not exist at a single snapshot.

The bridge-fee swap then uses the attacker-selected fee amount. Unlike the ordinary relay-fee swap, this path does not enforce the configured USD relay-fee swap cap. A caller can select an older favorable fee-token price and a newer favorable WNT price, swap caller-sized input at the artificial cross-rate, and retain the WNT output beyond the actual messaging fee. The reported production-parameter proof obtains more than 17 WNT of net LP loss on a 600-token swap, while coherent same-time reports are unprofitable.

RECOMMENDATION

Require all reports used in one atomic bridge-fee swap to share one authenticated observation timestamp or snapshot. Bind the atomic provider to the token's configured provider where possible, and enforce MAX_RELAY_FEE_SWAP_USD on bridgeFee.feeAmount. Cap the bridge buffer to the provider's quoted messaging fee plus a small tolerance and refund any excess.

RESOLUTION

Status: Resolved.

Guardian: The Team 1 M-05 remediation limits bridge fee swaps by their USD input value and the provider's quoted messaging fee, but it does not require the fee token and WNT reports to represent one observation. Atomic actions still accept any enabled atomic provider for each token, and `Oracle._setPrices` only requires the selected report timestamps to fit within `MAX_ORACLE_TIMESTAMP_RANGE`.

The same reports are used by both bridge fee and relay fee `AtomicSwap` paths. A caller can therefore combine valid observations from different permitted moments to obtain a cross asset rate that did not exist at one time. The mismatch is largest on a volatile fee token

M-23 Mixed-time reports bypass bridge-fee cap CONTINUED

swapped to WNT, for example WBTC to WETH. The bridge path limits the nominal conversion to the quoted messaging fee times `MAX_BRIDGE_FEE_SWAP_FACTOR`. The relay path has no equivalent quote relative bound: it only applies `MAX_RELAY_FEE_SWAP_USD`, deducts the actual relay fee from the resulting WNT, and returns the remainder to the user.

Recommendation
Retain the existing freshness and bridge fee limits, and cap relay fee swap output relative to the actual relay fee. Treat the remaining permitted timestamp skew as a documented bounded risk.

`https://github.com/gmx-io/gmx-synthetics-private/commit/f65caafbce2a081b487b48ec07fe3369b3f88c1d`

MEDIUM **M-24** ACKNOWLEDGED

Payout caps can underfund impact reserve

CATEGORY	Business logic errors
LOCATION	`contracts/market/MarketUtils.sol: getPoolValueInfo(), getCappedPnl(), getPositivePnlToPoolFactor(); contracts/adl/AdlUtils.sol: updateAdlState(); contracts/position/DecreasePositionCollateralUtils.sol: processCollateral()`
REVIEW	Main Review

DESCRIPTION

GM NAV deducts the position-impact pool because it backs future positive-impact credits. However, the trader-PnL cap and ADL trigger compare positive PnL with gross directional poolAmount without deducting the same reserve.

A profitable close can therefore pass the gross PnL cap while ADL remains inactive, even though the payout consumes liquidity that is still recorded as position-impact backing. After settlement, ordinary poolAmount can be lower than positionImpactPoolAmount. Later positive-impact credits may then be underfunded until liquidity or accounting is restored.

This is distinct from the existing borrowing-utilization finding: that finding understates borrowing rates; this one allows trader-PnL settlement to consume impact-reserve backing. The supplied proof requires a large simulated price move and does not establish a completed LP loss, so the impact is bounded but the reserve invariant is broken.

RECOMMENDATION

Calculate trader-PnL and ADL thresholds from unencumbered directional liquidity by deducting the applicable position-impact reserve, net of legitimate lent amounts. Alternatively, convert reserve consumed by ordinary PnL settlement into explicit lent-impact accounting. Add an invariant that prevents poolAmount from falling below recorded impact backing and regression tests for large impact reserves.

RESOLUTION

Status: Acknowledged.

MEDIUM M-25

ACKNOWLEDGED

Late GMX claims divert remote rewards

CATEGORY	Business logic errors
LOCATION	`contracts/fee/FeeUtils.sol: incrementClaimableFeeAmount(); contracts/fee/FeeHandler.sol: claimFees(); contracts/fee/FeeDistributor.sol: initiateDistribute(), processLzReceive(), bridgedGmxReceived(), getGmxAmounts()`
REVIEW	Main Review

DESCRIPTION

The weekly FeeDistributor snapshot includes withdrawable GMX and the FeeDistributorVault balance, but excludes accrued GMX-denominated market fees that are still claimable. During the asynchronous LayerZero read window, a local staker can permissionlessly claim those market fees.

When `processLzReceive` later executes, it withdraws the newly claimable GMX into the local FeeDistributorVault while calculating allocations from the earlier, smaller snapshot. The bridge-receipt flow then uses the live inflated vault balance as the distribution amount. Local stakers therefore receive GMX fee revenue that should have been allocated across the participating chains, reducing the remote cohort's weekly reward allocation.

The supplied production-path proof registers a GMX-long market, accrues and claims GMX fees through the public fee-claim path, and shows a local staker captures a material excess while the remote allocation is reduced.

RECOMMENDATION

Include every production claimable GMX fee source in an immutable epoch snapshot, or reserve fees claimed after the snapshot for the next epoch. Track bridge receipts by epoch/message and cap each chain's current distribution at its authenticated allocation, reserving excess balances for the following global allocation.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-26** RESOLVED

Funding mode switch reverses stale accrual

CATEGORY	Configuration
LOCATION	`contracts/config/ConfigSyncer.sol: sync(); contracts/config/Config.sol: setUint(); contracts/market/MarketUtils.sol: updateFundingState(), getNextFundingFactorPerSecond()`
REVIEW	Main Review

DESCRIPTION

ConfigSyncer allowlists `FUNDING_INCREASE_FACTOR_PER_SECOND` and can set it to zero without an oracle-priced funding checkpoint. This changes a market from adaptive/ramping funding to the static funding-factor path.

If funding is stale, the next action applies the new static factor and current open-interest skew to the entire period since `fundingUpdatedAt`, rather than applying the old adaptive state to the elapsed interval. The funding payer can reverse retroactively: a side that should have received funding can instead be charged, while the historical payer avoids its debt and receives claims. The incorrect accrual can liquidate otherwise healthy positions and changes the market's accrued accounting and GM token value without an external price move.

This is distinct from the previously reported minimum funding-ramp activation issue because it demonstrates a mode change that selects the wrong paying side.

RECOMMENDATION

Remove funding-mode keys from ConfigSyncer, or require a validated oracle-priced funding checkpoint under the old configuration before applying the new value. If retained, extend sync to settle every affected market atomically using old funding state and prices before writing the new configuration. Add regression coverage for an adaptive-to-static transition with stale funding and opposite saved/current payer directions.

RESOLUTION

Status: Resolved.

`https://github.com/gmx-io/gmx-synthetics-private/pull/205`

LOW **L-01** RESOLVED

GLV swap flag does not block execution

CATEGORY	Business logic errors
LOCATION	`contracts/exchange/GlvWithdrawalHandler.sol:49-54,87-110; contracts/glv/glvWithdrawal/GlvWithdrawalUtils.sol:228-292; contracts/withdrawal/ExecuteWithdrawalUtils.sol:103-105`
REVIEW	Main Review

DESCRIPTION

GLV withdrawals with a long or short token swap path validate `withdrawalSwapFeatureDisabledKey(address(this))` when they are created in `GlvWithdrawalHandler`. This resolves the key using the `GlvWithdrawalHandler` address. During execution, however, `GlvWithdrawalUtils._processMarketWithdrawal` converts the request into a regular `Withdrawal` and calls `WithdrawalHandler.executeWithdrawalFromController`. The shared `ExecuteWithdrawalUtils.executeWithdrawal` check executes in the `WithdrawalHandler` context through a library delegatecall, so `address(this)` resolves to `WithdrawalHandler` and a different feature key is validated.

As a result, an administrator can disable the GLV-handler-scoped withdrawal swap feature after a request has been created, but an already-pending GLV withdrawal with a swap path can still execute as long as the regular WithdrawalHandler-scoped key remains enabled. This makes the GLV-specific emergency switch effective for creation only and can allow queued swaps to execute during an incident despite the GLV swap feature having been disabled.

Example:

1. A GLV withdrawal with a non-empty swap path is created while both flags are enabled.
2. The administrator sets `withdrawalSwapFeatureDisabledKey(glvWithdrawalHandler)` to true.
3. A keeper executes the pending GLV withdrawal.
4. Execution validates only `withdrawalSwapFeatureDisabledKey(withdrawalHandler)`, so the GLV withdrawal succeeds if that separate flag is false.

RECOMMENDATION

When executing a GLV withdrawal with either swap path populated, also validate `withdrawalSwapFeatureDisabledKey(address(this))` in the `GlvWithdrawalHandler` execution context, preferably near the existing `executeGlvWithdrawalFeatureDisabledKey` check in `_executeGlvWithdrawal`. This ensures the same GLV-handler-scoped swap flag gates both creation and execution. The shared WithdrawalHandler-scoped validation may remain as an additional global safeguard for all withdrawals.

L-01 GLV swap flag does not block execution CONTINUED

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/184

LOW L-02 RESOLVED

Fee swaps lack a dedicated pause

CATEGORY	Validation
LOCATION	`contracts/multichain/MultichainTransferRouter.sol:186-234; contracts/router/relay/RelayUtils.sol:115-143; contracts/swap/SwapHandler.sol:23-32; contracts/router/relay/BaseGelatoRelayRouter.sol:41-52,469-471`
REVIEW	Main Review

DESCRIPTION

The protocol performs direct atomic swaps for two kinds of fee conversion without validating a swap-specific feature flag.

`MultichainTransferRouter._swapBridgeFeeIfNeeded` converts a non-WNT LayerZero bridge fee token to WNT by calling `SwapHandler.swap` directly. Likewise, `RelayUtils.swapFeeTokens` converts non-WNT relay fee tokens through the same handler. `SwapHandler.swap` only requires its caller to have the CONTROLLER role; it does not validate a feature flag based on `SwapPricingType.AtomicSwap`.

Consequently, disabling MarketSwap/LimitSwap order creation or execution does not stop these swaps because neither pathway passes through OrderHandler. `WITHDRAWAL_SWAP_FEATURE_DISABLED` also does not stop them. Relay fee swaps can only be halted indirectly by disabling the entire per-router `GASLESS_FEATURE_DISABLED` flag, which also disables every other gasless action and must be configured separately for each relay router. Bridge-fee swaps reached from controller-triggered bridge-outs are not covered by that gasless flag.

During a pool or pricing incident, governance therefore lacks a targeted mechanism to stop fee-token swaps while preserving unrelated gasless and bridge functionality. Fee conversion may continue through these direct SwapHandler paths even while ordinary swap-order and withdrawal-swap features are disabled.

RECOMMENDATION

Add an explicit pause for direct atomic fee swaps. One option is a shared `ATOMIC_SWAP_FEATURE_DISABLED` key validated in `SwapHandler.swap` whenever `params.swapPricingType == AtomicSwap`, scoped to the SwapHandler address. This provides a single emergency switch for relay-fee and bridge-fee conversions and any other direct atomic-swap callers.

If the operations must remain independently controllable, add dedicated `RELAY_FEE_SWAP_FEATURE_DISABLED` and `BRIDGE_FEE_SWAP_FEATURE_DISABLED` checks in the respective action layers. Perform each check immediately before transferring the fee token or invoking SwapHandler. Add tests showing these paths revert while their swap flag is disabled and remain unaffected by unrelated feature flags.

L-02 Fee swaps lack a dedicated pause CONTINUED

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/186

LOW L-03

RESOLVED

Bridge-fee swaps lack slippage protection

CATEGORY	Validation
LOCATION	`contracts/router/relay/IRelayUtils.sol:93-104; contracts/multichain/MultichainTransferRouter.sol:186-248; contracts/router/relay/MultichainRelayUtils.sol:307-327`
REVIEW	Main Review

DESCRIPTION

When a user pays a LayerZero bridge fee in a non-WNT token, `MultichainTransferRouter._swapBridgeFeeIfNeeded` swaps the signed `feeAmount` through GMX pools to obtain WNT. However, `BridgeFeeParams` contains only `feeToken`, `feeAmount`, and `feeSwapPath`; it provides no minimum acceptable WNT output. The router consequently calls `SwapHandler.swap` with `minOutputAmount: 0`.

The bridge-out EIP-712 hash commits to the input token, input amount, and path but likewise contains no output bound. A user therefore cannot protect the fee conversion against adverse prices, price impact, or execution timing. The complete signed fee-token input can be consumed even if the swap returns materially less WNT than expected.

The post-swap check only verifies that the output token is WNT, not that a sufficient amount was received. If the account already has WNT in its multichain balance, LayerZeroProvider may still deduct the required native messaging fee and complete the bridge, masking the poor conversion while the user's fee-token input suffers excessive loss. Otherwise, the later bridge may revert for insufficient WNT, reverting the swap atomically, but this does not protect cases with an existing WNT balance.

RECOMMENDATION

Add a minimum WNT output field, such as `minBridgeFeeOutputAmount`, to `BridgeFeeParams`. Include this field in `BRIDGE_OUT_TYPEHASH`, `getBridgeOutStructHash`, and all off-chain typed-data/signing helpers so a relayer cannot alter it. Pass the signed value as `SwapParams.minOutputAmount` instead of zero.

Add tests demonstrating that bridge-fee conversion reverts when the WNT output is below the signed minimum, including a case where the account already has enough WNT to pay the LayerZero messaging fee.

RESOLUTION

Status: Resolved.

L-03 Bridge-fee swaps lack slippage protection CONTINUED

https://github.com/gmx-io/gmx-synthetics-private/pull/196

LOW L-04 RESOLVED

Zero-pool state blocks collateral top-ups

CATEGORY	Business logic errors
LOCATION	`contracts/order/ExecuteOrderUtils.sol:64-76; contracts/position/IncreasePositionUtils.sol:111-127,139-175; contracts/market/MarketUtils.sol:2996-3007`
REVIEW	Main Review

DESCRIPTION

`ExecuteOrderUtils.executeOrder` permits zero-pool borrowing accrual only when the order is a decrease and `sizeDeltaUsd > 0`. A collateral-only top-up is instead represented as an increase order with a positive `initialCollateralDeltaAmount` and `sizeDeltaUsd == 0`, so the new allowance remains false.

Before the increase-order logic can process and add the collateral, `executeOrder` calls `PositionUtils.updateFundingAndBorrowingState`. If the position side has nonzero open interest/reserved USD but its pool value is zero, borrowing-factor calculation reverts with `UnableToGetBorrowingFactorEmptyPoolUsd`. The order is therefore cancelled before `IncreasePositionUtils` reaches the supported collateral-only path.

This prevents users from adding collateral to improve the health of an existing position precisely while the relevant pool is depleted. Users can still reduce position size through the separately relaxed nonzero decrease path, which limits the impact, but they cannot choose the non-exposure-changing recovery action of topping up collateral. A position may consequently remain closer to liquidation until pool liquidity is restored or the user reduces it.

RECOMMENDATION

Treat collateral-only increase orders as recovery actions for borrowing accrual. Set `allowZeroPoolBorrowingFactor` when either (1) the order is a decrease with nonzero `sizeDeltaUsd`, or (2) the order is an increase with `sizeDeltaUsd == 0` and a nonzero collateral deposit.

Keep ordinary size-increasing orders strict so users cannot expand exposure while borrowing is uncomputable. Add a regression test that creates a position, depletes the relevant pool side while open interest remains nonzero, and verifies that a zero-size increase order can add collateral while a positive-size increase remains blocked.

RESOLUTION

Status: Resolved.

L-04 Zero-pool state blocks collateral top-ups CONTINUED

https://github.com/gmx-io/gmx-synthetics-private/pull/190

LOW L-05

RESOLVED

Zero-pool market blocks healthy GLV shifts

CATEGORY	DoS
LOCATION	`contracts/glv/glvShift/GlvShiftUtils.sol:243-257; contracts/glv/GlvUtils.sol:52-70,95-120,190-199; contracts/market/MarketUtils.sol:2996-3007`
REVIEW	Main Review

DESCRIPTION

`GlvShiftUtils.executeGlvShift` tries to make its final `GlvValueUpdated` calculation best-effort by calling `GlvUtils.tryGetGlvValue` and skipping the event when `success` is false. However, this call passes `allowZeroPoolBorrowingFactor = false`.

`tryGetGlvValue` is not a general non-reverting wrapper: it calls `_getGlvValue` directly and only returns `success = false` when that function reports a market with negative pool value. `_getGlvValue` iterates every supported market in the GLV. If any constituent has reserved USD/open interest on a side whose `poolUsd` is zero, `MarketUtils.getBorrowingFactorPerSecond` reverts with `UnableToGetBorrowingFactorEmptyPoolUsd`, and that revert propagates through `tryGetGlvValue`.

Consequently, after shifting between healthy markets A and B, the informational full-GLV valuation can revert because of unrelated constituent C. Since the valuation occurs before the transaction completes, the revert rolls back the entire withdrawal/deposit shift. One depleted constituent can therefore prevent rebalancing among otherwise healthy markets, contrary to the code comment stating that an unrelated market should not block shifts.

This reduces GLV rebalancing availability during market distress, when keepers may need to move liquidity while another constituent remains temporarily depleted.

RECOMMENDATION

For the informational post-shift valuation, pass `allowZeroPoolBorrowingFactor = true`. This call is only used to emit `GlvValueUpdated`; it does not determine settlement amounts or enforce the source/destination shift limits.

Alternatively, make the best-effort boundary catch all valuation failures by moving the calculation behind an external view call and wrapping it in `try/catch`, omitting `GlvValueUpdated` whenever valuation reverts. Preserve strict zero-pool behavior in the source and destination valuations that protect shift accounting.

Add a regression test with three supported markets: shift between healthy A and B while C has zero pool value and nonzero open interest/reserved USD, and assert that the A-to-B shift completes while the informational event is either skipped or emitted using relaxed valuation.

L-05 Zero-pool market blocks healthy GLV shifts CONTINUED

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/190

LOW L-06 RESOLVED

Recovery executions lack usable reader quotes

CATEGORY	Business logic errors
LOCATION	`contracts/reader/Reader.sol:78-137,176-211,327-348; contracts/reader/ReaderDepositUtils.sol:109-134; contracts/reader/ReaderPositionUtils.sol:145-187; contracts/reader/ReaderUtils.sol:170-231; contracts/reader/GlvReader.sol:21-61; contracts/glv/GlvUtils.sol:211-232; contracts/market/MarketUtils.sol:165-185,2200-2213`
REVIEW	Main Review

DESCRIPTION

Recovery executions intentionally pass `allowZeroPoolBorrowingFactor = true`, but their important reader and quote paths retain strict valuation and revert with `UnableToGetBorrowingFactorEmptyPoolUsd` in the same state.

The mismatches cover all of the following cases:

1. A market deposit can execute because `ExecuteDepositUtils` relaxes both borrowing-state updates and pool valuation. However, `Reader.getDepositAmountOut` calls `ReaderDepositUtils` with the flag false, and `Reader.getMarketTokenPrice` reaches a hardcoded-false `MarketUtils.getPoolValueInfo`; both revert before users can quote the executable deposit.
2. A GLV deposit can execute because `ExecuteGlvDepositUtils` uses relaxed valuation for the GLV and destination market. However, `GlvReader.getGlvValue` passes false and `GlvReader.getGlvTokenPrice` calls a helper that hardcodes false. Either reader reverts if any GLV constituent has a depleted pool side with remaining open interest, preventing a quote for the executable GLV deposit.
3. A nonzero position decrease can execute because `ExecuteOrderUtils` relaxes borrowing accrual for that order. However, `Reader.getPositionInfo`, `getPositionInfoList`, and `getAccountPositionInfoList` all call `ReaderPositionUtils.getPositionInfo`, whose pending-fee calculation reaches `MarketUtils.getNextBorrowingFees` with false. Position/close quotes therefore revert even though the decrease is executable.
4. `Reader.getMarketInfo` computes both borrowing rates with false, and `getMarketInfoList` loops through that strict function. One depleted side makes the single-market endpoint revert and one affected market reverts the entire batch, hiding information for unaffected markets as recovery operations continue on-chain.

This does not directly prevent execution, but it impairs UI, keeper, and integrator visibility and quoting during the distressed state the recovery changes are designed to support.

RECOMMENDATION

Align quote semantics with each executable action rather than hardcoding strict borrowing valuation throughout the readers:

L-06 Recovery executions lack usable reader quotes CONTINUED

- Use relaxed pool valuation in `ReaderDepositUtils.getDepositAmountOut`, matching market-deposit execution.
- Add recovery/deposit-oriented GLV value and token-price reader variants (or an explicit flag) that use relaxed valuation, and use them when quoting GLV deposits.
- Propagate an `allowZeroPoolBorrowingFactor` option through position-info fee calculation and enable it when quoting a nonzero decrease.
- Make market-info endpoints remain usable during depletion, for example by returning zero plus an explicit `borrowingFactorAvailable` status per side. Ensure list endpoints isolate a failed market instead of reverting the entire batch.
- Provide an explicitly relaxed market-token price reader for deposit/recovery quoting while retaining a strict variant where strict valuation is required.

Add regression tests for every endpoint above in a state with zero pool value and nonzero open interest. Assert market and GLV deposit quotes succeed consistently with execution, nonzero decrease position quotes remain available, and one depleted market cannot revert the full market-info batch. Keep withdrawal and ordinary size-increase quotes strict where their executions remain strict.

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/194

LOW L-07

ACKNOWLEDGED

Zero-pool periods waive borrowing fees

CATEGORY	Business logic errors
LOCATION	`contracts/market/MarketUtils.sol:1862-1892,2917-2937,2996-3006; contracts/deposit/ExecuteDepositUtils.sol:126-139; contracts/order/ExecuteOrderUtils.sol:58-74; contracts/market/PositionImpactPoolUtils.sol:156-174`
REVIEW	Main Review

DESCRIPTION

When a pool side has nonzero open interest/reserved USD but `poolUsd == 0`, recovery paths may call borrowing accrual with `allowZeroPoolBorrowingFactor = true`. `getBorrowingFactorPerSecond` then returns zero.

This does more than avoid charging fees at the instant of the recovery action. `getNextCumulativeBorrowingFactor` multiplies that zero rate by the full duration since the last borrowing update, producing a zero delta. `updateCumulativeBorrowingFactor` nevertheless advances `cumulativeBorrowingFactorUpdatedAt` to the current timestamp. The elapsed interval is therefore permanently removed from cumulative borrowing fees and cannot be recovered after pool liquidity returns.

This behavior is reached by:

- market deposits, including the underlying market-deposit leg of GLV deposits, because `ExecuteDepositUtils` relaxes both sides before adding liquidity;
- nonzero position decreases, because `ExecuteOrderUtils` relaxes both sides before processing the reduction; and
- reductions of position-impact-pool lending, because `PositionImpactPoolUtils.reduceLentAmount` also relaxes both sides.

Repeated recovery actions continue advancing the timestamp with a zero delta while the side remains depleted. The deposit that ultimately restores liquidity also first updates borrowing in the depleted pre-deposit state, crystallizing the final skipped interval. Because the standard accrual model applies the current rate to all time since the prior update, the waived period can also begin before the pool actually became zero if no update occurred at that transition.

Borrowers on the depleted side consequently receive a borrowing-fee holiday, while the protocol/LP fee recipients forgo the corresponding revenue. Skipping an uncomputable rate may be a reasonable tradeoff to preserve recovery liveness, but the accounting consequence should be an explicit design decision.

RECOMMENDATION

L-07 Zero-pool periods waive borrowing fees CONTINUED

Consider whether permanently waiving borrowing fees for the elapsed zero-pool interval is the desired economic behavior and risk tradeoff for preserving recovery-action liveness. If it is desired, document the fee holiday explicitly, including that the first relaxed update advances the accrual timestamp with a zero delta and that the skipped fees are not later recovered. No code change is otherwise necessary.

RESOLUTION

Status: Acknowledged.

LOW L-08 RESOLVED

Subaccount approvals replay across wallets

CATEGORY	Signatures
LOCATION	contracts/router/relay/RelayUtils.sol:174-194; contracts/router/relay/SubaccountRouterUtils.sol:43-94; contracts/router/relay/SubaccountGelatoRelayRouter.sol:77-100
REVIEW	Main Review

DESCRIPTION

The signed SubaccountApproval omits the main account. Because the account is used only as the expected ERC-1271 signer, the same digest and signature can be accepted by multiple contract wallets that share an owner or signing policy. Each sibling has independent approval nonce, revocation, and action-count state, so consuming the approval on one wallet does not prevent its reuse on the others. A malicious or compromised delegated subaccount can then sign account-bound outer actions for each sibling.
Impact: An approval intended for one smart wallet can grant unauthorized order-management authority over every compatible sibling wallet with Router allowances. The attacker can pull their collateral into orders and subject those accounts to attacker-selected swaps, leveraged positions, fees, trading losses, protective-order cancellation, and liquidation risk. Independent action budgets make the exposure scale across the wallet fleet.

RECOMMENDATION

Add the main account address to a versioned SUBACCOUNT_APPROVAL_TYPEHASH and struct hash, and derive the approval digest with that account so pre-fix account-agnostic approvals cannot remain valid.

PROOF OF CONCEPT https://sentry.guardianaudits.com/shared/findings/PHccPGOBkG87?findingId=019f888e-13f3-7969-aeea-24a573ed1501

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/203

LOW L-09 RESOLVED

Collateral caps block risk reductions

CATEGORY	DoS
LOCATION	contracts/market/MarketUtils.sol:1342-1363; contracts/position/DecreasePositionUtils.sol:301-308
REVIEW	Main Review

DESCRIPTION

applyDeltaToCollateralSum checks the resulting collateral sum against MAX_COLLATERAL_SUM regardless of delta direction. If a routine risk reduction lowers the cap below current utilization, any decrease or liquidation that reduces collateralSum but leaves it above the new cap reverts. The fork PoC first successfully liquidates a real BTC position, restores state, lowers the cap through Config, and proves the identical liquidation then reverts with MaxCollateralSumExceeded.
Impact: Small closes and liquidations in an over-cap market/collateral/side bucket can be blocked until one action could bring the whole aggregate below the cap or configuration restores a higher cap. With many positions, no individual liquidation may satisfy that condition, delaying removal of insolvent exposure and increasing LP bad-debt risk. Trusted operators can remediate by raising the cap, limiting the supported severity.

RECOMMENDATION

Enforce MAX_COLLATERAL_SUM only for positive deltas or when nextValue exceeds the previous value. Always allow a valid negative delta that monotonically moves an over-cap aggregate toward compliance while retaining the underflow check.

PROOF OF CONCEPT https://sentry.guardianaudits.com/shared/findings/PHccPGOBkG87?findingId=019f889b-329d-7eda-9ded-d06952c189fa

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/commit/2f63b136ad70619bcb513233860ad2335f4f3fcc

LOW **L-10** RESOLVED

Staking signatures permit sibling-wallet reuse

CATEGORY	Signatures
LOCATION	contracts/router/relay/StakingUtils.sol:88-106; contracts/router/relay/BaseGelatoRelayRouter.sol:426-447,462-467; contracts/multichain/MultichainStakingRouter.sol:101-130; contracts/multichain/MultichainStakingUtils.sol:162-179
REVIEW	Main Review

DESCRIPTION

The staking EIP-712 action hashes omit the operated account. This creates a cross-wallet reuse edge case for ERC-1271 wallet families where multiple sibling accounts accept the same signature for an identical raw digest. A relayer may substitute a different sibling as the separately supplied account while the signature remains valid.

This does not affect EOAs, because ECDSA recovery must match the supplied account. It also does not affect contract wallets whose signature-validation scheme binds signatures to the individual wallet address. For staking-transfer theft, the signer must additionally authorize a receiver controlled by or colluding with the attacker, the relayer must substitute the valuable sibling before the intended transaction consumes the globally single-use digest, and the receiver must complete the transfer-acceptance handshake. The relayer cannot change the signed receiver.

Impact: Under these uncommon conditions, a staking-transfer signature intended for a low-value sibling may be reused against a higher-value sibling, transferring that sibling's transferable staking portfolio. Outside this narrow wallet and receiver configuration, the issue is limited to cross-wallet action substitution or griefing. This is considered a Low-severity edge case because exploitability depends on an uncommon ERC-1271 validation model and several user-specific preconditions despite the potentially significant conditional impact.

RECOMMENDATION

Include address account in every staking EIP-712 typehash and struct hash so each authorization is bound to the operated account. Update signing clients and tests accordingly, including regression coverage for sibling ERC-1271 wallets that share an owner or validation module.

PROOF OF CONCEPT https://sentry.guardianaudits.com/shared/findings/PHccPGOBkG87?findingId=019f8b46-e554-7ed5-a068-951b180c0501

L-10 Staking signatures permit sibling-wallet reuse CONTINUED

RESOLUTION

Status: Resolved.

https://github.com/gmx-io/gmx-synthetics-private/pull/195

LOW L-11

ACKNOWLEDGED

Data Stream depth is not bound to notional

CATEGORY	Oracle
LOCATION	`contracts/oracle/ChainlinkDataStreamProvider.sol:61-130; contracts/oracle/Oracle.sol:244-332`
REVIEW	Main Review

DESCRIPTION

Chainlink Data Stream bid and ask prices are documented as simulated price impact at a specified liquidity-depth utilization. GMX validates the signed report but retains only the price values: ValidatedPrice and Oracle carry neither the represented depth nor a maximum executable notional, and do not track cumulative use of the report.

If those bid/ask values are finite-depth execution quotes rather than unlimited reusable marks, a trader can apply one report to an action far larger than its represented depth or reuse it across several actions. The proof applies the same verifier-approved report to two 5,000-WETH market swaps, moving reserve-scale value. It proves the missing size and reuse binding, but not Chainlink's unpublished absolute capacity, profitability after GMX fees and price impact, or an actual deficit. This is therefore Low severity and conditional on the endpoint's intended depth semantics.

RECOMMENDATION

Confirm whether the Data Stream endpoint is intended to quote finite executable depth. If so, authenticate the represented depth or maximum notional, bind each action's USD size to it, and track cumulative use per report or feed timestamp across swaps and position actions. If it is intentionally an unlimited reusable mark, document why extrapolating it to arbitrary notional remains conservative.

RESOLUTION

Status: Acknowledged.

LOW L-12

ACKNOWLEDGED

Atomic withdrawal arbitrages oracle providers

CATEGORY	Oracle
LOCATION	`contracts/oracle/Oracle.sol:252-304; contracts/exchange/WithdrawalHandler.sol:154-195`
REVIEW	Main Review

DESCRIPTION

Queued actions enforce each token's configured oracle provider, while atomic withdrawals accept any globally approved atomic provider. An LP can mint GM under a higher configured queued-provider price, then redeem through an atomic withdrawal under a lower approved provider price.
For a market with profitable long open interest, the lower redemption quote reduces recognized trader-PnL liability and raises the GM redemption value. The proof uses an authenticated cross-provider basis and shows a public mint-and-atomic-withdraw cycle that remains modestly profitable after the configured 0.5% atomic-withdrawal fee and other costs. The issue is bounded by provider divergence, fees, liquidity, and the requirement that the alternate provider be globally approved for atomic actions.

RECOMMENDATION

Bind each token's atomic validation to its configured oracle provider. If different providers are intentionally permitted between queued and atomic modes, enforce a tight on-chain deviation bound between the sources before NAV-sensitive atomic actions.

RESOLUTION

Status: Acknowledged.

LOW L-13

ACKNOWLEDGED

Mixed fee vault migrations stall epochs

CATEGORY	Business logic errors
LOCATION	`contracts/config/TimelockConfig.sol:171-187; contracts/fee/FeeDistributor.sol:85-158, 258-284, 370-413`
REVIEW	Main Review

DESCRIPTION

Fee-receiver migration is chain-local and does not use a shared receiver generation. If a destination chain activates a new FeeDistributorVault while a source chain still maps that destination to its old vault, both distributors can begin otherwise valid new cycles. The source's OFT transfer resolves the stale peer-vault mapping and sends the epoch to the obsolete vault, while the destination's new distributor only recognizes the new vault.

The source cycle can complete while the destination remains waiting for data it will not receive. After response expiry, the ordinary state machine cannot advance or restart. Governance can recover the funds from the old vault and repair the configuration, so this is a recoverable cross-chain distribution liveness and reconciliation issue rather than permanent loss.

RECOMMENDATION

Introduce a cross-chain receiver generation and require peer mappings and the destination distributor to acknowledge the same generation before an epoch starts. Quiesce distributions during migrations, bind transfers to the acknowledged generation, and provide a timelocked timeout-safe recovery/reset transition.

RESOLUTION

Status: Acknowledged.

LOW L-14

ACKNOWLEDGED

Direct GLV price bypasses negative-market check

CATEGORY	Oracle
LOCATION	`contracts/glv/GlvUtils.sol:76-122; contracts/glv/GlvUtils.sol:171-208`
REVIEW	Main Review

DESCRIPTION

GlvUtils.getGlvValue normally traverses supported constituents and reverts with GlvNegativeMarketPoolValue if any held market has negative pool value. However, when Oracle supplies a nonempty direct GLV-token price, getGlvValue returns that price multiplied by GLV supply before traversing any constituent.

The direct GLV price is unsigned, so this is not a check for a negative oracle price. The missing validation is that a whole-GLV oracle price can be used while a held constituent has negative value. Operations priced through this branch can proceed without the normal constituent-level negative-value circuit breaker, creating a discrepancy between direct GLV pricing and constituent-derived pricing.

RECOMMENDATION

When a direct GLV-token oracle price is used for minting, redemption, or other value-sensitive actions, still validate that all supported constituents with nonzero GLV balances have nonnegative pool value. Alternatively, restrict the direct-price branch to informational reads or require a trusted upstream GLV oracle to attest to the constituent health condition.

RESOLUTION

Status: Acknowledged.

LOW L-15

ACKNOWLEDGED

Negative GLV removal rewards late deposits

CATEGORY	Business logic errors
LOCATION	`contracts/glv/GlvUtils.sol:76-122, 383-416; contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol:63-89; contracts/glv/glvWithdrawal/GlvWithdrawalUtils.sol:294-326`
REVIEW	Main Review

DESCRIPTION

With a direct GLV-token oracle price, whole-GLV valuation can bypass constituent traversal and permit deposits while a supported constituent has negative pool value. If governance disables that distressed constituent and later removes it under a dust threshold, its negative contribution is omitted from subsequent GLV valuation without an equivalent finalized supply adjustment.

A depositor can enter after cleanup is signaled but before removal, minting GLV at the depressed aggregate value. After removal, the newly minted shares capture the upward valuation discontinuity and can be redeemed from healthy constituent backing. The proof deposits healthy GM before removal and withdraws more healthy value afterward, shifting the distressed-market write-off to incumbent holders. This is Low severity because it requires a negative constituent, a direct GLV oracle-price path, a governance cleanup sequence, and a timing window.

RECOMMENDATION

Freeze GLV deposits before a constituent with negative value can be disabled or removed, or atomically crystallize its signed value and adjust GLV supply/accounting at removal. Do not allow a direct whole-GLV oracle-price branch to bypass constituent-health validation for value-sensitive minting. Make removal dust thresholds value-aware and bind cleanup requests to a lifecycle epoch begun at disablement.

RESOLUTION

Status: Acknowledged.

LOW L-16

ACKNOWLEDGED

Swaps bypass open-interest reserve buffer

CATEGORY	Validation
LOCATION	`contracts/swap/SwapUtils.sol:364-408; contracts/market/MarketUtils.sol:2061-2083; contracts/position/IncreasePositionUtils.sol:172-188`
REVIEW	Main Review

DESCRIPTION

Position increases validate both the ordinary reserve factor and OPEN_INTEREST_RESERVE_FACTOR after adding exposure. A swap that removes tokenOut from pool liquidity validates only the ordinary reserve factor. Where the ordinary factor is looser, a user can reduce reserve-side liquidity below the level required by the configured OI reserve for unchanged outstanding positions.
The proof demonstrates a state that passes the ordinary reserve check but fails the OI-reserve requirement for the same open interest. It does not demonstrate realized user loss. This is Low severity and depends on OPEN_INTEREST_RESERVE_FACTOR being intended as a persistent liquidity buffer rather than solely an admission check for new increases.

RECOMMENDATION

If the OI reserve is intended to remain satisfied after liquidity changes, validate MarketUtils.validateOpenInterestReserve for the tokenOut side after swap pool deltas. Otherwise document that the OI reserve is only an increase-admission control and swaps are governed solely by the ordinary reserve factor.

RESOLUTION

Status: Acknowledged.

LOW L-17

ACKNOWLEDGED

Disabled markets accrue trapped borrowing debt

CATEGORY	Business logic errors
LOCATION	`contracts/exchange/BaseOrderHandler.sol:69-74; contracts/market/MarketUtils.sol:2849-2853, 2917-2937; contracts/position/PositionUtils.sol:564-599`
REVIEW	Main Review

DESCRIPTION

A full IS_MARKET_DISABLED pause blocks position execution and liquidation before borrowing can be checkpointed, but borrowing accrual remains based on elapsed wall-clock time since the last cumulative-factor update. When the market is re-enabled, the first successful action applies borrowing across the entire disabled period.

Traders cannot close or reduce exposure during the pause, yet their borrowing debt can grow enough to make a previously healthy position liquidatable at unchanged prices once the market resumes. This is a Low-severity emergency-pause policy issue, not an untrusted exploit.

RECOMMENDATION

If disabled-time borrowing is not intended, checkpoint borrowing at disablement and exclude paused wall-clock time on re-enablement. Alternatively permit reduce-only closes and liquidations with validated prices during a full market pause. Store explicit pause and resume times rather than deriving accrual solely from the last successful market action.

RESOLUTION

Status: Acknowledged.

LOW L-18

ACKNOWLEDGED

Depeg midpoint overpays positive swap impact

CATEGORY	Math
LOCATION	`contracts/oracle/ChainlinkPriceFeedProvider.sol:38-73; contracts/price/Price.sol:25-27; contracts/swap/SwapUtils.sol:210-316; contracts/multichain/MultichainTransferRouter.sol:63-79, 186-248`
REVIEW	Main Review

DESCRIPTION

For a downward-depegged stablecoin, ChainlinkPriceFeedProvider constructs an oracle range from the live price to the configured peg. The base swap values tokenIn at the conservative live minimum, while SwapUtils measures inventory change and positive impact with the range midpoint.
At a severe depeg, the two valuations diverge sharply: for a \$0.001 live price and \$1 peg, input is valued near \$0.001 while inventory improvement is valued near \$0.50. A bridge-fee atomic swap can therefore receive an oversized positive-impact credit in WNT. The proof uses honest depeg pricing and shows roughly \$10 of live-value input receiving about \$495 after fees through the bridge-fee conversion path. The excess is paid from LP-owned swap-impact inventory. This is Low severity because it requires a severe honest depeg, available impact inventory, and the bridge-fee conversion path.

RECOMMENDATION

Use consistent executable live prices for both base swap value and inventory/impact deltas during a peg-spanning stablecoin price band, or represent the peg reference separately from the tradable oracle band. Add tests for both depeg directions. As defense in depth, cap bridge-fee conversion value and only convert or credit WNT actually required for the bridge.

RESOLUTION

Status: Acknowledged.

LOW L-19

ACKNOWLEDGED

Smaller-side exemption forgives stale borrowing

CATEGORY	Configuration
LOCATION	`contracts/config/Config.sol:277-300; contracts/config/RiskOracleConfig.sol:164-179; contracts/market/MarketUtils.sol:1862-1886, 2917-2937, 2945-2994`
REVIEW	Main Review

DESCRIPTION

Enabling the global SKIP_BORROWING_FEE_FOR_SMALLER_SIDE mode does not checkpoint markets under the prior rules. At the next checkpoint, a smaller-side position's complete stale interval is evaluated under the newly enabled exemption, produces a zero borrowing rate, and advances cumulativeBorrowingFactorUpdatedAt. Borrowing fees accrued before activation are permanently forgiven. The erased fees are a market receivable included in GM valuation, so this governance transition can also lower GM price without a token transfer when the receivable is written down. Borrowers retain value otherwise owed to LPs and fee recipients. This is a Low-severity configuration-transition issue, requiring a global mode change while markets are stale.

RECOMMENDATION

Checkpoint both sides of all enabled markets under the old mode before activation, or introduce an activation timestamp and piecewise borrowing accrual. Do not expose the transition only as an immediate global boolean write without an atomic settlement mechanism.

RESOLUTION

Status: Acknowledged.

LOW L-20

ACKNOWLEDGED

Gross swap impact ignores fee-net pool credit

CATEGORY	Math
LOCATION	contracts/swap/SwapUtils.sol:223-249, 255-316, 364-380; contracts/pricing/SwapPricingUtils.sol:280-306
REVIEW	Main Review

DESCRIPTION

SwapUtils computes inventory impact from gross mid-price token deltas before calculating and allocating fees. The recorded tokenIn pool credit excludes claimable UI and fee-receiver amounts. A trader controlling the UI fee receiver can reclaim the UI fee while receiving a positive-impact rebate priced from inventory improvement that never fully reaches the pool.

By traversing two markets with opposite imbalances, the trader can collect rebates based on gross improvements and turn an otherwise losing rebalance profitable. The proof executes two real swap legs and turns about \$500 into roughly \$505.43; an identical fee-net-delta control finishes below \$500. The opportunity requires suitable opposite imbalances, available impact inventory, and a self-controlled UI fee receiver, so this is Low severity.

RECOMMENDATION

Calculate fees before price impact and use only the tokenIn amount actually credited to poolAmount plus the tokenOut debit that execution records. Exclude claimable UI and fee-receiver amounts, then validate the final impact quote against final pool deltas.

RESOLUTION

Status: Acknowledged.

LOW L-21

ACKNOWLEDGED

GLV bypasses reserve withdrawal lock

CATEGORY	Business logic errors
LOCATION	`contracts/glv/glvDeposit/ExecuteGlvDepositUtils.sol: _processMarketDeposit(); contracts/glv/glvWithdrawal/GlvWithdrawalUtils.sol: _getMarketTokenAmount(); contracts/market/MarketUtils.sol: validateReserve()`
REVIEW	Main Review

DESCRIPTION

A direct GM withdrawal validates the market's reserve factor, so a market token that would reduce backing below the required open-interest reserve cannot be redeemed. The direct GM-to-GLV deposit path does not perform that reserve validation, while a GLV withdrawal checks the reserve only for the selected output constituent.

A holder can deposit reserve-locked GM into a GLV at its accounted value and redeem the resulting fungible GLV through a separate healthy constituent. The locked exposure remains in the GLV while healthy constituent liquidity exits. This bypasses the per-market withdrawal circuit breaker and gives early GLV exiters priority during a reserve incident. It does not pay more than accounted GLV NAV, so the impact is a bounded liquidity-isolation and safety-policy failure.

RECOMMENDATION

Apply reserve validation before accepting direct GM into a GLV when that GM can contribute to withdrawals from other constituents. Alternatively, make GLV withdrawals pro rata or ensure each constituent's attributed reserve requirement remains satisfied after a cross-constituent redemption.

RESOLUTION

Status: Acknowledged.

LOW L-22

ACKNOWLEDGED

Impact-pool withdrawals enable sandwiches

CATEGORY	MEV
LOCATION	`contracts/market/PositionImpactPoolUtils.sol: withdrawFromPositionImpactPool(); contracts/market/MarketUtils.sol: applyDeltaToPoolAmount(), applyDeltaToVirtualInventoryForSwaps(); contracts/pricing/SwapPricingUtils.sol: getPriceImpactUsdWithVirtualInventory()`
REVIEW	Main Review

DESCRIPTION

A scheduled withdrawal from the position-impact pool removes long and short pool amounts proportionally. The generic pool accounting also changes the shared virtual swap inventory, but this administrative transition does not calculate or settle swap price impact.

A searcher can swap in the imbalance-improving direction before a disclosed withdrawal and receive positive impact at the larger pre-withdrawal inventory scale. Once the withdrawal contracts the real pools and virtual inventory, the searcher reverses the swap and pays less negative impact under the convex impact curve. The difference can exceed trading fees, extracting value from the market's impact/pool backing.

The demonstrated outcome is conditional on a sufficiently large scheduled withdrawal, market imbalance, available impact reserve, and reliable transaction ordering, so this is rated Low.

RECOMMENDATION

Do not mutate shared virtual swap inventory through proportional administrative impact-pool withdrawals without a no-arbitrage rescaling or explicit impact settlement. Preserve the pre-withdrawal virtual inventory, or apply a dedicated transition that ensures an equal pre/post round trip cannot profit after fees. Consider bounding withdrawals relative to pool inventory.

RESOLUTION

Status: Acknowledged.

LOW L-23

ACKNOWLEDGED

Dust improvement discounts full deposit

CATEGORY	Business logic errors
LOCATION	`contracts/pricing/SwapPricingUtils.sol: _getPriceImpactUsd(), _getSwapFees(); contracts/deposit/ExecuteDepositUtils.sol: executeDeposit(), _executeDeposit()`
REVIEW	Main Review

DESCRIPTION

Deposit fee selection reduces the whole two-token request to a single comparison: whether the aggregate post-deposit imbalance is smaller than the initial imbalance. In an imbalanced market, a balanced X/X deposit does not improve the imbalance and pays the adverse fee tier on both legs. Adding one unit of the deficient token changes the aggregate result to improving, so the entire X/(X+1) deposit receives the lower improving tier.

The dust improvement therefore discounts a much larger neutral notional. With the production 0.05% improving and 0.07% adverse fee factors, the supplied example avoids 400 fee units on two 1,000,000-unit legs, reducing protocol revenue and LP fee accrual while minting more GM to the depositor. The effect is capital-intensive and limited by market caps.

RECOMMENDATION

Apply the improving fee only to the part of a deposit that reduces the pre-action imbalance. Split the request into improving and neutral/worsening portions, or prorate the discount by the actual magnitude of imbalance improvement. Document the intended aggregation semantics and add regression coverage for a dust deficient-token pad on a large balanced deposit.

RESOLUTION

Status: Acknowledged.

LOW L-24

ACKNOWLEDGED

Paused source strands peer fee allocation

CATEGORY	Configuration
LOCATION	`contracts/fee/FeeDistributor.sol: processLzReceive(), distribute(), _distributeFees(), bridgedGmxReceived()`
REVIEW	Main Review

DESCRIPTION

Release 2.2.1 adds a chain-local `FEE_DISTRIBUTOR_DISTRIBUTE_FEES` switch, but the cross-chain fee snapshots and peer allocation calculations do not include that state.

If a surplus source chain disables distribution while an enabled destination expects its weekly GMX allocation, the source treats itself as sufficiently funded, skips the planned bridge, completes its round, and sends GMX to its local treasury. The destination has already booked the source GMX as inbound and cannot complete its distribution without the missing funds. Re-enabling the source does not reopen its completed round, so the ordinary flow can remain stuck after the read-response window expires.

This requires a governance emergency pause on a surplus source, so it is a low-severity configuration/liveness issue rather than an untrusted exploit.

RECOMMENDATION

Commit every chain's distribution-enabled state into the shared epoch snapshot and exclude disabled-chain inventory from enabled peer allocations. Alternatively, require a globally consistent mode before any chain processes the round. A disabled surplus source should bridge the committed allocation before treasury distribution, or safely cancel all peer rounds. Add a recovery path that reconciles custody without raw-state intervention.

RESOLUTION

Status: Acknowledged.

LOW L-25

ACKNOWLEDGED

Funding debts overstate loss offsets

CATEGORY	Business logic errors
LOCATION	`contracts/position/PositionUtils.sol: getPositionPnlUsd(); contracts/market/MarketUtils.sol: getPositivePnl(); contracts/position/DecreasePositionCollateralUtils.sol: processCollateral(), handleEarlyReturn()`
REVIEW	Main Review

DESCRIPTION

Trader-PnL caps and ADL checks net raw PnL across positions in the same side and collateral bucket. A losing position's full raw negative PnL can therefore offset a winner's profit for the cap calculation.
On liquidation, accrued funding is paid before negative trading PnL. A position can have enough gross collateral to support its full raw loss in the cap metric, but after senior funding is paid only a small amount remains collectible by the pool. A winner can close first using the overstated loss offset, while the later liquidation realizes less loss for LPs. The funding receiver retains the senior payment.
The effect depends on large accrued funding, close ordering, and liquidation conditions, so it is rated Low.

RECOMMENDATION

For trader-PnL caps and ADL exposure, limit negative-PnL offsets to the collateral collectible after accrued funding and other senior obligations. Alternatively reserve funding obligations in the collectibility metric or make settlement bounds independent of winner/liquidation ordering. Add regression tests for a winning position paired with a funding-encumbered loser.

RESOLUTION

Status: Acknowledged.

LOW L-26 ACKNOWLEDGED

PnL-cap changes reprice GLV shares

CATEGORY	Configuration
LOCATION	`contracts/config/Config.sol: setUint(); contracts/config/ConfigUtils.sol: validateRange(); contracts/market/MarketUtils.sol: getPoolValueInfo(); contracts/glv/GlvUtils.sol: _getGlvMarketValue(); contracts/glv/glvDeposit/GlvDepositCalc.sol: getMintAmount(); contracts/glv/glvWithdrawal/GlvWithdrawalUtils.sol: _getMarketTokenAmount()`
REVIEW	Main Review

DESCRIPTION

`Config.setUint` applies changes to `MAX_PNL_FACTOR_FOR_DEPOSITS` immediately. Lowering the cap changes a profitable constituent market's recognized liability and can increase its GM and aggregate GLV NAV without any custody or supply change.

A temporary depositor can mint GLV through a healthy constituent before an anticipated legitimate cap reduction, then redeem through that healthy constituent after the NAV increase. The uplift is attributable to a distressed constituent already held by GLV, so incumbent holders bear the dilution. The supplied example deposits \$800 before an 80%-to-20% cap reduction, then redeems \$875; without the update it redeems \$800.

This is a configuration-transition variant of the existing GLV revaluation finding. It requires a privileged, observable cap change and is rated Low.

RECOMMENDATION

Pause affected GM and GLV deposits and withdrawals around NAV-changing PnL-cap updates, activate the update from a snapshot with a value-neutral share adjustment, or prevent one constituent's discontinuous revaluation from being redeemed against healthy constituents. Add tests for mint-before/redeem-after cap transitions.

RESOLUTION

Status: Acknowledged.

LOW L-27

ACKNOWLEDGED

Cross-token lending can halt liquidations

CATEGORY	DoS
LOCATION	`contracts/market/MarketUtils.sol: capPositiveImpactUsdByPositionImpactPool(), applyDeltaToPoolAmount(); contracts/position/DecreasePositionUtils.sol: pnlToken selection; contracts/position/DecreasePositionCollateralUtils.sol: processCollateral()`
REVIEW	Main Review

DESCRIPTION

Positive-impact lending capacity is calculated from the combined USD value of both market-token pools. But a realized positive-impact rebate is paid in the position's `pnlToken` only—longs use the long-token pool and shorts use the short-token pool.

The aggregate check can authorize lending because the opposite token pool has sufficient value, while the actual payout-token pool lacks enough raw tokens. During liquidation or ADL, settlement then attempts to debit more of the payout token than that pool holds and reverts on the negative-pool invariant. The position remains liquidatable but uncloseable while the side-specific shortage persists.

The issue requires impact lending to be enabled and independently moving payout tokens, so it is rated Low.

RECOMMENDATION

Cap positive impact by the actual available `poolAmount` of the payout token in addition to aggregate lending limits, so the pool deduction cannot underflow. During liquidation and ADL, haircut an unaffordable positive-impact rebate rather than reverting. Add tests with asymmetric token pools and independently moving index, long, and short token prices.

RESOLUTION

Status: Acknowledged.

LOW L-28

ACKNOWLEDGED

Stable price bands discount funding

CATEGORY	Oracle
LOCATION	`MarketUtils.getNextFundingAmountPerSize; Oracle atomic-price validation; WithdrawalHandler.executeAtomicWithdrawal`
REVIEW	Main Review

DESCRIPTION

Funding is defined in USD but converted into collateral tokens using the stablecoin's maximum price. During a depeg, that transfers too few tokens to satisfy the USD funding target at the conservative minimum price. The same maximum-price conversion is used for payer and receiver, so token counts balance while the receiving side is underpaid in USD. Any GM holder can trigger the funding checkpoint with a small atomic withdrawal using an enabled stable-price provider. At a 12% depeg, \$10m paying exposure and a 0.01% hourly rate over 24 hours, the payer avoids \$2,880 of a \$24,000 funding obligation.

RECOMMENDATION

Convert the funding USD amount using the same conservative minimum price for both payer and receiver. Alternatively, prevent stable-price providers from checkpointing funding or define and tightly bound a dedicated common funding mark.

RESOLUTION

Status: Acknowledged.

LOW L-29

ACKNOWLEDGED

Relay approvals bypass disabled integrations

CATEGORY	Access Control
LOCATION	`Subaccount relay approval handler and SubaccountGelatoRelayRouter.createOrder`
REVIEW	Main Review

DESCRIPTION

Subaccount approval signatures commit to an integration identifier, but the relay approval handler does not store or validate that identifier. An approval signed while an integration is enabled can therefore establish a relay delegate session after that integration has been disabled. The delegate can consume its bounded action allowance to create, update, or cancel the owner's orders despite the emergency disablement. Receiver restrictions and action limits constrain direct theft, but malicious trading or cancellation of protective orders can still expose the owner to loss.

RECOMMENDATION

Persist the integration identifier from relay approvals and validate, at session establishment and each delegated action, that the corresponding integration remains enabled. Invalidate pending approvals for a disabled integration.

RESOLUTION

Status: Acknowledged.

LOW L-30

ACKNOWLEDGED

ADL ignores collectible borrowing fees

CATEGORY	Business logic errors
LOCATION	`ADL trigger and borrowing-fee checkpoint paths`
REVIEW	Main Review

DESCRIPTION

The ADL trigger evaluates profitability and PnL-cap state without first checkpointing accrued borrowing fees, even when those fees are fixed and collectible. A keeper can execute ADL before a negligible decrease realizes the fees, causing a profitable victim position to be force-deleveraged under a lower trader-PnL cap. If borrowing is checkpointed first, the additional pool backing can make ADL unnecessary at the same price. This creates avoidable loss through keeper ordering discretion.

RECOMMENDATION

Checkpoint accrued borrowing for the relevant market and side before computing the ADL trigger and payout cap, or conservatively include collectible accrued borrowing in the ADL accounting.

RESOLUTION

Status: Acknowledged.

LOW L-31

ACKNOWLEDGED

Bridge signatures lack fee cap

CATEGORY	Validation
LOCATION	`contracts/multichain/MultichainTransferRouter.sol:63-82; contracts/multichain/LayerZeroProvider.sol:196-278`
REVIEW	Main Review

DESCRIPTION

The signed bridge-out authorization commits to the bridged token, amount, provider, destination data, minimum destination output, and optional bridge-fee swap input. It does not commit to a maximum native LayerZero/Stargate messaging fee. At execution, LayerZeroProvider.prepareSend quotes the current messaging fee and bridgeOut deducts that live WNT amount from the account's multichain balance.
A relayer can execute a still-valid signed bridge request when the live messaging cost is materially higher than the cost the user anticipated. If the account has sufficient WNT, the larger fee is charged; otherwise the bridge reverts. The user has a minimum output bound for the bridged asset but no equivalent bound for the messaging fee.

RECOMMENDATION

Add a signed maximum messaging-fee field to BridgeOutParams and reject execution when the quoted native fee exceeds it. The signed cap should cover both direct-WNT payment and a fee-token conversion path, with an explicit tolerance for normal quote movement. Preserve the existing minimum destination-output check independently.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-01 RESOLVED

CREATE2 wallets are build-coupled

CATEGORY	Upgradeability
LOCATION	`contracts/staking/GmxAccountWalletFactory.sol:22-32,35-48`
REVIEW	Main Review

DESCRIPTION

`GmxAccountWalletFactory` derives each account wallet with CREATE2 using the factory address, an account-derived salt, and the hash of `GmxAccountWallet` creation code concatenated with the configured `RoleStore` constructor argument. The predicted wallet identity is therefore coupled to the exact factory deployment, wallet creation bytecode, compiler version/settings and metadata, and `RoleStore` address. Redeploying the factory, modifying `GmxAccountWallet`, changing its compilation settings, or configuring a different `RoleStore` produces a different wallet address for every account. Callers use `getWalletAddress(account)` as the canonical destination for staking operations, so a replacement factory will resolve accounts to new wallets while existing staking positions and assets remain associated with wallets produced by the previous factory.

This is expected CREATE2 behavior rather than an error in the address formula, but it creates an important upgrade and migration constraint. Operational procedures that treat the factory as freely replaceable may fragment an account's staking state or make old positions inaccessible through integrations that only query the replacement factory.

RECOMMENDATION

Document the factory, exact wallet init-code hash, compiler configuration, and `RoleStore` as permanent components of wallet identity. Before upgrading or replacing any of them, inventory wallets and staking positions created by the old factory and provide an explicit migration or legacy-routing path.

Consider exposing the wallet init-code hash and a factory/version identifier so deployment tooling and integrations can verify the expected wallet namespace. Regression tests should pin representative predicted addresses for the production compiler configuration and confirm that upgrade procedures continue to resolve or operate legacy wallets.

RESOLUTION

Status: Resolved.

I-01 CREATE2 wallets are build-coupled CONTINUED

https://github.com/gmx-io/gmx-synthetics-private/pull/249

INFORMATIONAL I-02

ACKNOWLEDGED

Min-size update promotes old order

CATEGORY	Unexpected Behavior
LOCATION	contracts/position/DecreasePositionUtils.sol:198-216, 282-290; contracts/position/PositionUtils.sol:806-852; contracts/order/DecreaseOrderUtils.sol:132-149
REVIEW	Main Review

DESCRIPTION

A pending zero-size collateral-only decrease order is handled by the minimum-position-size cleanup as though it were a partial size decrease. If governance later raises MIN_POSITION_SIZE_USD above a grandfathered position, the execution path rewrites the order's sizeDeltaUsd to the full position size and clears its explicit collateral delta.
The order retains its original receiver, output and swap settings even though those were chosen for a collateral-only action. This can unexpectedly close the full position, realize PnL, apply full-close fees, and cancel sibling protection orders after a legitimate configuration update.

RECOMMENDATION

Do not promote zero-size collateral-only orders during minimum-size cleanup. Grandfather existing positions and collateral-only orders, or cancel affected orders and require a newly created full-close request with fresh receiver, price, swap, and output terms.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-03

ACKNOWLEDGED

Transient OI bypasses collateral buffer

CATEGORY	Unexpected Behavior
LOCATION	contracts/market/MarketUtils.sol:2597-2606; contracts/position/PositionUtils.sol:545-561; contracts/position/DecreasePositionUtils.sol:131-164, 321-342; contracts/position/IncreasePositionUtils.sol:190-208
REVIEW	Main Review

DESCRIPTION

The open-interest-scaled minimum collateral factor is evaluated only when the target position is changed. A user can temporarily reduce same-side open interest by closing a separate position, execute a collateral-only decrease on a target while the dynamic factor is lower, and then reopen the separate position. When open interest is restored, the target is not revalidated against the restored dynamic collateral requirement. This leaves the target below the intended high-open-interest buffer, although it does not by itself bypass the separate liquidation threshold or demonstrate direct loss.

RECOMMENDATION

If the dynamic OI-scaled factor is intended as a persistent safety requirement, preserve a per-position floor, introduce time-weighting or cooldown before a lower factor applies, or revalidate affected positions when same-side OI is raised.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-04 ACKNOWLEDGED

Delegated resize preserves non-owner payout

CATEGORY	Access Control
LOCATION	contracts/subaccount/SubaccountUtils.sol:37-48; contracts/router/SubaccountRouter.sol:153-182; contracts/order/OrderUtils.sol:178-218; contracts/position/DecreasePositionUtils.sol:257-290
REVIEW	Main Review

DESCRIPTION

Subaccount order creation requires receiver and cancellationReceiver to equal the main account, but delegated update of an owner-created order does not re-enforce that invariant. A manager distinct from an existing non-owner receiver can enlarge a small decrease order while preserving its recipient. If the replacement size fully closes the position, full-close processing adds all remaining collateral and realized PnL to the output paid to that receiver.

A compromised order-management key can therefore combine delegated sizing authority with a preexisting non-owner payout order to close the entire position to that recipient. The manager and recipient may be separate accounts, so the owner need not have granted either actor both capabilities. This is informational because it requires both delegated management authority and a preexisting non-owner receiver configuration.

RECOMMENDATION

Enforce main-account receiver and cancellation-receiver invariants whenever a subaccount updates an order, or prohibit delegated size increases on orders whose receiver is not the owner. Consider separate create, update, and cancel permissions with owner-configured size limits.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-01** RESOLVED

Swap restoration backcharges borrowing fees

CATEGORY	Logical Error
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/swap/SwapUtils.sol#L360-L376
REVIEW	Remediation Review 1

DESCRIPTION

Borrowing checkpoints multiply the elapsed time by the rate calculated from the current pool. When reserved USD remains but one side's pool is zero, recovery paths correctly treat that side's rate as zero. However, ordinary swap orders have `market == address(0)`, so the pre-order update does not checkpoint the markets in `swapPath`. `SwapUtils` can therefore replenish the depleted token pool without advancing its borrowing timestamp.

The next market checkpoint sees a nonzero restored rate and applies it to all time since the old timestamp, retroactively charging borrowers for the zero-liquidity interval. This can materially reduce borrower collateral or make positions liquidatable.

RECOMMENDATION

Be aware and document this behavior or consider to checkpoint borrowing for every swap-path market immediately before changing its pool amounts. Use the pre-swap state and allow zero accrual for a depleted side so its timestamp advances without retroactive fees.

RESOLUTION

Status: Resolved.

MEDIUM M-02 RESOLVED

USD restore inflates reconstructed PnL cap

CATEGORY Logical Error
LOCATION contracts/position/PositionUtils.sol:getPositionPnlUsd()
REVIEW Remediation Review 1

DESCRIPTION

The M-09 cumulative cap records only USD. A capped decrease pays `basePnlUsd / pnlTokenPrice.max` tokens from the pool, then stores `realizedPnlUsd`. The next decrease rebuilds backing as remaining tokens at the current minimum price plus that historical USD amount:

```
cache.poolTokenUsd =
    cache.poolTokenAmount * cache.poolTokenPrice +
    cache.paidPnlUsd.toUint256() /
        MarketUtils.getPoolDivisor(market.longToken, market.shortToken);
```

The tokens actually removed are worth `paidPnlUsd * currentMin / historicalMax` at the later price. Adding face value USD overstates `poolTokenUsd` whenever `historicalMax > currentMin`, including a normal oracle min versus max spread. `getCappedPnl` and the cumulative allowance then pay extra profit from the pool.

The gap is largest on synthetic longs whose index can stay profitable while the payout token falls (many Arbitrum markets pay long PnL in WETH). Same price partitioned closes still match a one shot close. The extra is about `maxPnlFactor * paidPnlUsd * (1 - currentMin / historicalMax)` while the cap remains binding, and the same path is used if the residual is later decreased, liquidated, or ADL'd.

For example, a binding 50% cap, \$1,000,000 index PnL, and 10,000 payout tokens: a half close at \$100 pays \$250,000 and removes 2,500 tokens. After the payout token falls to \$80, remaining tokens are worth \$600,000. Restoring 2,500 tokens at \$80 gives \$800,000 backing and a \$400,000 cap, so \$150,000 should remain. The USD restore instead uses \$850,000 backing and a \$425,000 cap, and the second close pays \$175,000.

RECOMMENDATION

Reconstruct previously paid capped PnL from the tokens actually removed, valued at the current minimum price (and the single token market divisor), so cumulative trader payout cannot exceed the cap applied to that current token value.

M-02 USD restore inflates reconstructed PnL cap CONTINUED

RESOLUTION

Status: Resolved.

MEDIUM **M-03** RESOLVED

Historical reads survive remote reset

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L210-L219 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L438-L443 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L210-L219 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L438-L443)
REVIEW	Remediation Review 1

DESCRIPTION

LayerZero Read fixes each remote read at the destination's request timestamp. `resetDistribution()` only clears the source chain's current snapshot epoch and local expected GUID, so a response captured before the reset still contains the old valid snapshot.

`_validateSnapshotEpochs()` accepts that historical value whenever it matches the destination's committed epoch. It does not establish that the source still participates in the round. Another chain can therefore accept the abandoned snapshot and bridge the source's old allocation after the source has reset. The reset delay is measured from the source's last local state update, not from the destination's in-flight request time. A new read after reset sees epoch 0 and steps back, but an in-flight read does not. The reset chain cannot distribute that round, so received GMX enters a later snapshot and is allocated using a different staking cohort.

RECOMMENDATION

Ensure a reset cannot race any valid historical read. For example, define a shared initiation cutoff, reject new reads after it, and permit reset only after every response initiated before the cutoff has expired.

RESOLUTION

Status: Resolved.

MEDIUM **M-04** ACKNOWLEDGED

Late completion consumes the next epoch

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L410-L411 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L769-L774 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L410-L411 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L769-L774)
REVIEW	Remediation Review 1

DESCRIPTION

An old fee distribution is allowed to complete after the next weekly epoch begins, but `distribute()` records its completion using the current `block.timestamp`. `_validateDistributionNotCompleted()` then treats that wall clock timestamp as proof that the current epoch was already distributed.

```
_setUint(Keys2.FEE_DISTRIBUTOR_DISTRIBUTION_TIMESTAMP, block.timestamp);

if (lastDistributionTime > startOfDistributionWeek) {
    revert Errors.FeeDistributionAlreadyCompleted(...);
}
```

The week boundary is only a clock change in `_getEpochId()`. Mid-round state is unchanged. If keepers then `distribute()` epoch E after E+1 has started, state returns to `None` and the completion timestamp lands inside E+1. Every later E+1 `commitSnapshot()` reverts until E+2. Reset cannot unstick it because reset requires state other than `None`. Because all chains require matching snapshot epochs, one late completion can prevent the protocol wide weekly reward update.

RECOMMENDATION

Track the finalized snapshot epoch separately from its completion time. `distribute()` and `resetDistribution()` should mark the committed epoch as finalized or abandoned, and the completion guard should compare epoch identifiers.

RESOLUTION

Status: Acknowledged.

LOW **L-01** ACKNOWLEDGED

Old snapshots are overwritten too early

CATEGORY	Warning
LOCATION	`contracts/fee/FeeDistributor.sol:commitSnapshot()`
REVIEW	Remediation Review 1

DESCRIPTION

`FeeDistributor` stores only one committed snapshot tuple per chain because the snapshot keys are derived from `chainId`, not from both `chainId` and `epoch`. A chain that completes an epoch can therefore overwrite its old tuple as soon as the next epoch begins. However, a delayed peer is allowed to initiate the old epoch until one week plus `FEE_DISTRIBUTOR_MAX_READ_RESPONSE_DELAY` after that epoch started.

```
_setUint(Keys2.feeDistributorSnapshotFeeAmountGmxKey(block.chainid), feeAmountGmx);
_setUint(Keys2.feeDistributorSnapshotStakedGmxKey(block.chainid), stakedGmx);
_setUint(Keys2.feeDistributorSnapshotEpochKey(block.chainid), epochId);

if (block.timestamp > committedEpoch + 1 weeks +
    _getUint(Keys2.FEE_DISTRIBUTOR_MAX_READ_RESPONSE_DELAY)) {
    revert Errors.StaleSnapshotEpoch(committedEpoch);
}
```

For example, chains A and B commit epoch E, but only A completes it. When E+1 starts, A commits again and overwrites its E tuple. If B initiates E during the permitted response delay, it reads A's E+1 tuple, fails the epoch check, and returns to `Committed`. Retrying cannot recover A's deleted E tuple. B must overwrite its own old snapshot or reset the round, causing its E rewards to be delayed or skipped and potentially distributed against a later stake set.

RECOMMENDATION

Version committed snapshot values by both chain and epoch, and retain each tuple until every peer's permitted completion window has ended. Each read should request the tuple for the initiator's committed epoch so starting a new round cannot invalidate an older round that is still considered fresh.

RESOLUTION

Status: Acknowledged.

LOW L-02 RESOLVED

Swap fix depends on PnL cap ordering

CATEGORY	Configuration
LOCATION	\`contracts/swap/SwapUtils.sol:_validateMaxPnl()\`
REVIEW	Remediation Review 1

DESCRIPTION

The H-02 remediation validates the growing side of a swap against `MAX_PNL_FACTOR_FOR_DEPOSITS`, while trader payouts and trader-context GM value use `MAX_PNL_FACTOR_FOR_TRADERS`. The configuration write path validates each value independently and does not enforce the documented invariant `deposit cap <= trader cap`.

If the trader cap is 10% and the deposit cap is 20%, a market with \$300,000 long PnL against a \$2 million long pool has a raw factor of 15%. Both swap checks pass against 20%. A value-neutral swap can grow the long pool to \$3.1 million, release the trader-context PnL cap from \$200,000 to the full \$300,000, and reduce GM pool value by \$100,000.

Current static and live Arbitrum and Avalanche samples had no violations, so the issue is dormant unless configuration drift creates the invalid ordering.

RECOMMENDATION

Enforce `MAX_PNL_FACTOR_FOR_DEPOSITS <= MAX_PNL_FACTOR_FOR_TRADERS` for each market and side on every relevant configuration update. As an additional safeguard, the growing-side swap check can use the lower of the deposit and trader factors so configuration drift cannot reopen the repricing transition.

RESOLUTION

Status: Resolved.

LOW L-03

ACKNOWLEDGED

Funding bounds remain cadence-dependent

CATEGORY	Math
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/market/MarketUtils.sol#L1854-L1868
REVIEW	Remediation Review 1

DESCRIPTION

Adaptive funding charges the mean of the bounded start and end rates across the full checkpoint interval. If the raw rate reaches the minimum or maximum partway through, the correct integral is piecewise, so endpoint averaging remains dependent on checkpoint timing.
Normal orders checkpoint funding, so otherwise identical elapsed time and skew can produce different funding transfers based on intervening activity.

RECOMMENDATION

Be aware and document this behavior.

RESOLUTION

Status: Acknowledged.

LOW L-04

ACKNOWLEDGED

Staking roles absent from role policy

CATEGORY	Configuration
LOCATION	\`config/roles.ts:requiredRolesForContracts\`
REVIEW	Remediation Review 1

DESCRIPTION

The new deployment scripts require `GmxAccountWalletFactory` to have `CONTROLLER` and `MultichainStakingRouter` to have both `CONTROLLER` and `ROUTER_PLUGIN`. Neither contract is included in the matching `requiredRolesForContracts` lists.
Direct role grants are skipped on existing mainnet deployments. The production role update and validation flow therefore does not require these grants and treats an attempted grant to either unlisted contract as unapproved. If the contracts are deployed without a separate manual role update, the factory cannot write the account wallet registry and the staking router cannot drive controller protected wallets or vault operations, leaving cross chain staking unusable.

RECOMMENDATION

Add both contracts to the `CONTROLLER` policy and add `MultichainStakingRouter` to `ROUTER_PLUGIN`. Include their deployed addresses in the Arbitrum role update used to activate the feature.

RESOLUTION

Status: Acknowledged.

LOW L-05

ACKNOWLEDGED

Stale delegation overrides newer choice

CATEGORY	Signatures
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/router/relay/BaseGelatoRelayRouter.sol#L452-L456 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainStakingRouter.sol#L111-L119 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/router/relay/BaseGelatoRelayRouter.sol#L452-L456 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainStakingRouter.sol#L111-L119)
REVIEW	Remediation Review 1

DESCRIPTION

Each delegateGovGmx authorization uses an independently generated random userNonce. Replay protection records only the exact digest, so executing a newer delegation does not invalidate an older distinct signature.

A relayer holding two unexpired signatures can execute the newer choice first and submit the older choice afterward. Both digests are valid, and the older call delegates the wallet's full current voting power back to the previous delegatee. Tokens remain in the account wallet, but the user's latest governance delegation can be overridden until the older signature expires.

```
if (digests[digest]) {
    revert Errors.InvalidUserDigest(digest);
}
digests[digest] = true;
```

RECOMMENDATION

Use short delegation deadlines and document that a newer delegation does not cancel older signed requests. If latest choice ordering is required, enforce a monotonic delegation nonce per account.

RESOLUTION

Status: Acknowledged.

LOW L-06

ACKNOWLEDGED

Zero stake blocks treasury fee mode

CATEGORY	Math
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L305
REVIEW	Remediation Review 1

DESCRIPTION

FeeDistributor calculates the current chain's proportional GMX allocation before checking FEE_DISTRIBUTOR_DISTRIBUTE_FEES. If every configured tracker reports zero supply, totalStakedGmx is zero and the calculation reverts with division by zero.

```
uint256 requiredGmxAmount = Precision.mulDiv(
    totalFeeAmountGmx,
    stakedGmxCurrentChain,
    totalStakedGmx
);
```

This also blocks the treasury only mode, even though that mode does not need a proportional staker allocation. A complete staking migration or shutdown can therefore prevent the weekly fee round from progressing until some tracker supply becomes nonzero.

RECOMMENDATION

When staker distribution is disabled, bypass the proportional calculation and proceed with the treasury path. If distribution is enabled with zero aggregate stake, handle that state explicitly rather than dividing by zero.

RESOLUTION

Status: Acknowledged.

LOW L-07

ACKNOWLEDGED

Failed withdrawal can close later vesting

CATEGORY	Signatures
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/router/relay/StakingUtils.sol#L22-L23 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainStakingUtils.sol#L176-L188 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/router/relay/StakingUtils.sol#L22-L23 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainStakingUtils.sol#L176-L188)
REVIEW	Remediation Review 1

DESCRIPTION

WithdrawVesting signs the account and relay parameters but no vesting amount or position state. If the action arrives while no position exists, the staking router reverts and LayerZero catches the failure. This also rolls back digest consumption while leaving the signed payload publicly available.
If the account creates or enlarges a vesting position before the signature expires, anyone holding that payload can submit it again. IVester.withdraw() then closes the entire current position. The returned GMX and esGMX are credited to the victim, but the remaining esGMX must be vested again and its conversion is delayed.

```
bytes32 public constant WITHDRAW_VESTING_TYPEHASH =
    keccak256(bytes("WithdrawVesting(address account,bytes32 relayParams)"));
```

RECOMMENDATION

Use short withdrawal deadlines and make users aware that failed signed requests remain retryable. If this retry behavior is not acceptable, bind the authorization to the current deposited vesting amount or a position nonce.

RESOLUTION

Status: Acknowledged.

LOW L-08

ACKNOWLEDGED

Fee sweep misclassifies buyback GMX

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L488-L490 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeHandler.sol#L59-L67 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L488-L490 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeHandler.sol#L59-L67)
REVIEW	Remediation Review 1

DESCRIPTION

The H-05 fix records post snapshot GMX as any FeeHandler withdrawable amount above the `commitSnapshot()` baseline. A separate `FEE_KEEPER` can call `FeeHandler.withdrawFees()` in that window, which moves the snapshotted GMX into `FeeDistributorVault` and resets the accumulator to zero. A later public buyback below the old baseline is then recorded as zero post snapshot fees.

```
uint256 postSnapshotFeeAmountGmx = withdrawableGmx > snapshotWithdrawableGmx
    ? withdrawableGmx - snapshotWithdrawableGmx
    : 0;
```

For example, 10,000 GMX withdrawable and 40,000 GMX already in the vault at snapshot, then an intermediate sweep, then a 9,900 GMX buyback: the recorder sees 9,900 against a 10,000 baseline and stores zero. `processLzReceive` then sweeps the 9,900 into the vault. `getGmxAmounts()` treats 59,900 minus zero as this week's balance, so a deficit chain can infer a bridge fill that never arrived. Tokens stay in protocol vaults and can be paid to the wrong weekly stake set.

RECOMMENDATION

Keep the post snapshot meter correct across `FEE_KEEPER` withdrawals, or do not sweep FeeHandler GMX between snapshot and record.

RESOLUTION

Status: Acknowledged.

LOW **L-09** RESOLVED

Bridge fee swaps are underpriced

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/deposit/DepositUtils.sol#L107-L112 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainTransferRouter.sol#L188-L252 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/deposit/DepositUtils.sol#L107-L112 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainTransferRouter.sol#L188-L252)
REVIEW	Remediation Review 1

DESCRIPTION

GM and GLV request creation stores the user supplied `dataList`, but execution fee validation counts only the request's principal swap paths. PR 179 adds `BridgeFeeParams.feeSwapPath` inside that unpriced data. During settlement, `BridgeOutFromControllerUtils` can execute every bridge fee swap hop before a failing minimum output check. The bridge failure is caught, so the principal request remains executed and keeper payment is capped by an execution fee that omitted those swap hops and oracle price work. A user can therefore make a keeper subsidize several terminal atomic swap attempts, or cause simulation aware keepers to refuse the underfunded request.

RECOMMENDATION

Decode bridge fee data when the request is created and include every fee swap hop and required oracle price in execution fee validation and keeper payment accounting.

RESOLUTION

Status: Resolved.

LOW **L-10** ACKNOWLEDGED

Legacy bridge payloads cannot be decoded

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/BridgeOutFromControllerUtils.sol#L85-L89 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/BridgeOutFromControllerUtils.sol#L213-L217 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/BridgeOutFromControllerUtils.sol#L85-L89 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/BridgeOutFromControllerUtils.sol#L213-L217)
REVIEW	Remediation Review 1

DESCRIPTION

PR 179 appends dynamic bridge fee tuples to the stored `BridgeOut` action schema without adding a version marker or legacy decoding branch. Pending deposits and withdrawals created by the old handlers retain the previous five field or eight field payload in `dataList`.
The new handler decodes those payloads using the bridge fee aware schema. This decode occurs before the caught call to `MultichainTransferRouter`, so it reverts the parent execution path. The handler then cancels the otherwise valid GM or GLV request, returns its principal on the controller chain, and charges execution gas. The user does not receive the intended destination chain output and must submit a new request.

RECOMMENDATION

Either drain every legacy request before activating the new handlers, or add an explicit payload version with a backward compatible decoder for requests already stored on chain.

RESOLUTION

Status: Acknowledged.

LOW L-11 ACKNOWLEDGED

Delegated updates divert execution fee escrow

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/order/OrderUtils.sol#L190-L211 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/gas/GasUtils.sol#L479-L489 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/order/OrderUtils.sol#L190-L211 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/gas/GasUtils.sol#L479-L489)
REVIEW	Remediation Review 1

DESCRIPTION

An authorized subaccount can change a callback bearing limit or stop loss decrease order from a swap enabled mode to `NoSwap`. The update stores the new mode before estimating execution gas, so the estimated gas limit falls by one `singleSwapGasLimit` unit. The resulting lower maximum fee is then applied to the sum of the account's existing execution fee and any WNT received during the update:

```
params.order.setDecreasePositionSwapType(params.decreasePositionSwapType);

cache.estimatedGasLimit = GasUtils.estimateExecuteOrderGasLimit(params.dataStore,
  params.order);
(uint256 executionFee, uint256 executionFeeDiff) = GasUtils.validateAndCapExecutionFee(
    params.dataStore,
    cache.estimatedGasLimit,
    params.order.executionFee() + cache.receivedWnt,
    cache.oraclePriceCount,
    params.shouldCapMaxExecutionFee
);
```

The direct `SubaccountRouter` enables this cap for callback bearing orders even if the subaccount supplies no WNT. The Gelato and multichain subaccount routes enable it when `executionFeeIncrease` is nonzero, which can be satisfied with one wei. Any amount above the lower cap is removed from the order and sent to `HOLDING_ADDRESS`, where the account has no onchain claim.

With the default 1,000,000 single swap gas limit, 1x estimation multiplier, and 100x maximum fee multiplier, the reduction can reach `100,000,000 * block.basefee` wei. At a base fee of 0.01 gwei, this is 0.001 WNT taken from the account's existing escrow. The subaccount does not receive the funds, but can force the account to lose custody of them.

RECOMMENDATION

L-11 Delegated updates divert execution fee escrow CONTINUED

Separate the existing execution fee from WNT received during the update. Fee capping should affect only newly received WNT; any reduction to existing escrow should remain with the order or be returned to the order account.

RESOLUTION

Status: Acknowledged.

LOW L-12

ACKNOWLEDGED

Market switchover can report false success

CATEGORY	Configuration
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/scripts/switchover/run.sh#L73-L89 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/scripts/compareMarketConfig.ts#L115-L143 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/scripts/switchover/run.sh#L73-L89 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/scripts/compareMarketConfig.ts#L115-L143)
REVIEW	Remediation Review 1

DESCRIPTION

The market hours wrapper suppresses the update command's exit status and treats the presence of any `tx sent:` line as success. It does not verify that every batch was submitted or that the reported transaction succeeded. A reverted or dropped transaction, or a failure after an earlier batch was sent, can therefore produce a success notification while some or all risk parameters remain unchanged.

```
timeout 180 bash "\$SCRIPT" > "\$RUN_OUTPUT" 2>&1 || true
TX_HASH="\$(tail -20 "\$RUN_OUTPUT" | awk '/tx sent:/ { sub(/.*tx sent: /, ""); print }' |
  tail -1)"

if [ -n "\$TX_HASH" ]; then
  send_telegram "... succeeded"
fi
```

The configuration monitor does not detect the wrong session. It accepts the open and closed value for each key independently, so the complete open profile and mixtures of both profiles pass during closed hours. For example, a failed GOLD close can leave the minimum collateral factor at 0.9% and maximum open interest at \$5 million instead of the intended 3.5% and \$2.5 million. The system can consequently retain the more permissive trading limits while the reference market is closed without generating an operational alert.

RECOMMENDATION

Report success only after the update command exits successfully and every transaction receipt confirms execution. Make the monitor derive the expected market session and require all monitored parameters to match that session's complete profile.

RESOLUTION

Status: Acknowledged.

LOW L-13

ACKNOWLEDGED

Config updater can undo oracle controls

CATEGORY	Configuration
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/scripts/updateMarketConfigUtils.ts#L9-L25
REVIEW	Remediation Review 1

DESCRIPTION

RiskOracleConfig update POSITION_FEE_FACTOR and MAX_BORROWING_FACTOR_PER_SECOND, but the standard market updater does not classify either key as Risk-Oracle-managed.
A normal CONFIG_KEEPER run can therefore treat a legitimate oracle value as drift and restore the static repository value.

RECOMMENDATION

Be aware and consider excluding both keys from ordinary reconciliation.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL **I-01** ACKNOWLEDGED

Read delay consumes bridge grace period

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L753-L758 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L138-L146 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L753-L758 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L138-L146)
REVIEW	Remediation Review 1

DESCRIPTION

`MultichainReader` embeds the request timestamp in the LayerZero Read response. `FeeDistributor` stores that value as `FEE_DISTRIBUTOR_READ_RESPONSE_TIMESTAMP` and also uses it as the start of the bridge shortfall grace period.

```
bool graceElapsed = block.timestamp - readResponseTimestamp > gracePeriod;
```

Incoming bridge transfers can only begin after the read response is processed. If an authentic response is delayed longer than the configured grace period but remains within the maximum response delay, the deficit chain can accept a partial balance immediately after processing the response. Current epoch rewards are underfunded and the later bridge receipt moves into a future epoch. This path is active only when both shortfall tolerance parameters are configured to nonzero values.

RECOMMENDATION

If shortfall tolerance is enabled, set the grace period longer than typical LayerZero Read latency, or start the timer when the contract enters `ReadDataReceived`.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-01** RESOLVED

Historical reads survive remote reset

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L210-L219 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L438-L443 (https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/multichain/MultichainReader.sol#L210-L219 https://github.com/gmx-io/gmx-synthetics-private/blob/95d5eddad39aa311f2b90f3cee8e981204edd046/contracts/fee/FeeDistributor.sol#L438-L443)
REVIEW	Remediation Review 2

DESCRIPTION

LayerZero Read fixes each remote read at the destination's request timestamp. `resetDistribution()` only clears the source chain's current snapshot epoch and local expected GUID, so a response captured before the reset still contains the old valid snapshot.

`_validateSnapshotEpochs()` accepts that historical value whenever it matches the destination's committed epoch. It does not establish that the source still participates in the round. Another chain can therefore accept the abandoned snapshot and bridge the source's old allocation after the source has reset. The reset delay is measured from the source's last local state update, not from the destination's in-flight request time. A new read after reset sees epoch 0 and steps back, but an in-flight read does not. The reset chain cannot distribute that round, so received GMX enters a later snapshot and is allocated using a different staking cohort.

RECOMMENDATION

Ensure a reset cannot race any valid historical read. For example, define a shared initiation cutoff, reject new reads after it, and permit reset only after every response initiated before the cutoff has expired.

RESOLUTION

Status: Resolved.

MEDIUM **M-02** RESOLVED

Live cap reprices forgone PnL history

CATEGORY	Logical Error
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/position/PositionUtils.sol#L274-L314 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/position/DecreasePositionUtils.sol#L270-L285 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/position/PositionUtils.sol#L274-L314 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/position/DecreasePositionUtils.sol#L270-L285)
REVIEW	Remediation Review 2

DESCRIPTION

The M-08 remediation stores the largest reconstructed pool-PnL denominator, which prevents a break-even size reduction from shrinking the denominator applied to historical PnL. However, `_boundHistoricalCappedPnl()` still calculates the historical allowance as `realizedUncappedPnlUsd * currentCappedPoolPnl / historicalPoolPnl`. `currentCappedPoolPnl` is recalculated from current pool-token backing and the current `MAX_PNL_FACTOR_FOR_TRADERS`, so later liquidity or a higher trader cap can improve the ratio applied to PnL that was already settled and forgone.

For example, a position with \$1,000,000 profit closes 75% at a 50% payout ratio. The \$750,000 tranche receives \$375,000 and forgoes \$375,000. The trader then temporarily deposits 200 WNT at \$5,000, adding \$1,000,000 of backing and raising the live ratio to 75%. The remaining \$250,000 tranche should receive \$187,500, but the historical formula reprices the first tranche and pays the full \$250,000, reviving \$62,500 of previously forgone PnL. After closing, the GM minted by the temporary deposit was withdrawn for value within \$1 of the original deposit. A trader-cap increase produced the same \$250,000 versus \$187,500 mismatch for dual-token longs, dual-token shorts, single-token longs, and single-token shorts.

RECOMMENDATION

Be aware and consider to persist each capped close's fixed PnL allocation or fixed forgone amount. Apply later live backing and cap changes only to the currently unrealized tranche; do not multiply historical uncapped PnL by a later `cappedPoolPnl`. Add permanent regressions for a permissionless deposit-close-withdraw round trip and for a trader-cap increase across long, short, dual-token, and single-token markets.

RESOLUTION

Status: Resolved.

MEDIUM M-03 RESOLVED

V1 migration strands staked GLP

CATEGORY Business logic errors

LOCATION https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L266-L282 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L243-L251 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L266-L282 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L243-L251)

REVIEW Remediation Review 2

DESCRIPTION

`acceptStakingTransferFromV1` makes the account wallet call V1 `RewardRouterV2.acceptTransfer` on the user's raw V1 address. That call moves the sender's entire staking account. After the GMX legs, V1 reads the sender's `feeGlpTracker` GLP deposit, restakes that GLP for the wallet, and copies `glpVester` stake accounting onto the wallet. The wallet holds fsGLP.

```
IGmxAccountWallet(wallet).execute(
    address(contracts.rewardRouter),
    abi.encodeCall(IRewardRouterV2.acceptTransfer, (sender))
);
```

The wallet API unstakes GMX and esGMX through the V1 router and sweeps the base token into the multichain balance. It exposes neither `unstakeAndRedeemGlp` nor a `glpVester` action. `GmxAccountWallet.execute` is controller only, so the user cannot redeem directly. `withdrawFromWallet` only sweeps a token to the multichain vault. fsGLP is in private transfer mode, so that sweep reverts. sGLP is the transfer wrapper: `StakedGlp.transfer` unstakes the GLP and restakes it for the recipient. The recipient is the vault, so the vault becomes the staker and the account is credited sGLP. Moving that credit to an EOA goes through `TokenUtils.transfer`, which reverts while `tokenTransferGasLimit` for sGLP is zero. Both live wrappers, `0x5402B5F40310bDED796c7D0F3FF6683f5C0cFfdf` and `0x2F546AD4eDD93B956C8999Be404cdCAFde3E89AE`, have that value at zero on Arbitrum DataStore `0xFD70de6b91282D8017aA4E741e9Ae325CAb992d8`. The stake stays on the vault, and the user still cannot call V1 `unstakeAndRedeemGlp`.

The remaining path is a transfer of the whole account to a fresh EOA. That EOA calls V1 `acceptTransfer(wallet)`, which compounds the wallet and restakes its GMX through `stakedGmxTracker.stakeForAccount`. The tracker pulls the GMX with `transferFrom`. The staking router grants the wallet's GMX allowance only inside wallet to wallet `acceptStakingTransfer`, then clears it. Reward claims and compound do the same. The EOA call never receives the allowance, so the transfer reverts while any GMX remains staked or any GMX fee remains claimable.

Recovering the GLP therefore requires unstaking all GMX and esGMX first. Both unstake

M-03 V1 migration strands staked GLP CONTINUED

paths burn multiplier points in proportion to the amount unstaked, so a full recovery burns the boost. A user who migrates 100 staked GLP together with 50 staked GMX can still claim WETH fees, but the 100 GLP stays in the wallet until that GMX position and its multiplier points are destroyed and a fresh EOA accepts the account.

RECOMMENDATION

Add a signed account wallet action that redeems the wallet's staked GLP through V1 `unstakeAndRedeemGlp`, sweeps the output token into the multichain vault, and credits the account. Bind the redemption amount, output token, and minimum output in the signature.

RESOLUTION

Status: Resolved.

MEDIUM **M-04** RESOLVED

Compose reentrancy consumes victim actions

CATEGORY	Frontrunning
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/LayerZeroProvider.sol#L95-L171 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/LayerZeroProviderUtils.sol#L263-L285 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/LayerZeroProvider.sol#L95-L171 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/LayerZeroProviderUtils.sol#L263-L285)
REVIEW	Remediation Review 2

DESCRIPTION

LayerZero's `EndpointV2.lzCompose` is permissionless and marks a queued compose message as received before calling `LayerZeroProvider`. A third party can therefore choose when any public queued message executes.

An attacker can submit their own valid relay action whose signed `externalCalls` invokes `EndpointV2.lzCompose` for a victim's queued message. The attacker's router entrypoint has already entered its `nonReentrant` guard when external calls run. The nested compose credits the victim's bridged tokens, then dispatches the victim's follow-up action to the same router. That call fails on the active reentrancy guard.

`LayerZeroProviderUtils` catches the router failure and returns normally, so the nested endpoint call succeeds and the victim's compose message remains permanently marked as received. The victim does not lose the bridged tokens, but their signed deposit, withdrawal, order, or staking follow-up is deterministically discarded and must be signed and submitted again. An attacker can repeat this for each visible compose message at the cost of their own valid relay action.

WdRecommendation

Do not let a caught router failure permanently consume the follow-up action. Persist failed action data by compose GUID and expose a safe retry path, or revert the compose and ensure the token credit remains idempotent so LayerZero can retry it. The fix should account for indirect calls to the endpoint rather than relying on a target denylist in `externalCalls`.

RESOLUTION

Status: Resolved.

MEDIUM **M-05** ACKNOWLEDGED

Migration disables multichain staking wallets

CATEGORY	Business logic errors
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/deploy/deployMultichainStakingRouter.ts#L38-L53 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/scripts/incentive/generateGovActions.ts#L292-L342 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/deploy/deployMultichainStakingRouter.ts#L38-L53 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/scripts/incentive/generateGovActions.ts#L292-L342)
REVIEW	Remediation Review 2

DESCRIPTION

`MultichainStakingRouter` stores its reward router immutably, and the Arbitrum deployment script always supplies the configured `RewardRouterV2`. The incentive migration later revokes V2's handler and minter permissions across the staking trackers, esGMX, the legacy vesters, bnGMX, and the governance token while granting the replacement permissions to `RewardRouterV3`.

After those revocations, account-wallet staking, unstaking, compounding, reward claims, and staking transfers continue to call the de-permissioned V2 router and revert. `vestEsGmx` also remains tied to the legacy GMX vester, where cap zeroing prevents new deposits, and `withdrawVesting` can service only legacy positions. No account-wallet action exposes the new RatioVester deposit, withdrawal, or claim functions. Users cannot call V3 directly from an account wallet because wallet execution is controller-only. The migration status script expects the staking router to be redeployed against V3, but the deployment script cannot produce that configuration. `LayerZeroProvider` also stores the staking-router address immutably, so replacing only the staking router does not repair compose-dispatched actions. Executing the migration as shipped can therefore make staked positions and new RatioVester claims held by multichain wallets unavailable until governance deploys and wires both replacements.

RECOMMENDATION

Before revoking V2's permissions, deploy a `MultichainStakingRouter` against V3 and a `LayerZeroProvider` against that router, grant the required roles, and switch all upstream routing atomically. Add account-wallet actions for each designated RatioVester and preserve a withdrawal and claim path for existing legacy-vester positions. Extend the migration gate to verify the provider points to the replacement staking router.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-06** RESOLVED

Vault handler bypasses esGMX transfer limits

CATEGORY	Logical Error
LOCATION	- https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/utils/multichain.ts#L96-L112 - https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainRouter.sol#L31-L75
REVIEW	Remediation Review 2

DESCRIPTION

Commit `d0ba42094` fixed esGMX staking transfers under V1 private transfer mode by requiring handler status for both `MultichainVault` and the staking router. Commit `01655f516` replaced the router's broad handler authority with a constrained `EsGmxTransferHandler`, but deliberately retained the vault's handler grant for vault to wallet transfers.

The vault also serves generic signed transfer requests. These requests accept arbitrary token and receiver arrays, validate only that the receiver is nonzero, then decrement the signer's multichain balance and call `multichainVault.transferOut` for the selected token and receiver.
Because a handler can transfer esGMX while V1 private transfer mode is enabled, a holder with an esGMX multichain balance can include esGMX in a GM or GLV deposit's transfer requests and send it to any EOA. This bypasses the token's transfer restriction and lets holders transfer or sell esGMX outside the guarded whole-account staking migration flow.

RECOMMENDATION

Do not grant generic `MultichainVault` esGMX handler status. Route both esGMX directions through a constrained handler that fixes the vault side and validates the staking wallet, and reject esGMX in generic transfer request and withdrawal paths.

RESOLUTION

Status: Resolved.

LOW L-01

RESOLVED

Active wallets reject V1 position imports

CATEGORY	Compatibility
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L260-L283 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/mock/MockRewardRouterV2.sol#L204-L227 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L260-L283 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/mock/MockRewardRouterV2.sol#L204-L227)
REVIEW	Remediation Review 2

DESCRIPTION

`acceptStakingTransferFromV1` resolves the account's existing wallet and calls the real V1 `RewardRouterV2.acceptTransfer` with that wallet as the receiver. V1 accepts a staking transfer only into a fresh receiver: reward-tracker average stake and cumulative rewards, legacy-vester transfer history, and vester balances must all be zero.

An account wallet that has already accrued staking rewards therefore cannot import the same user's raw-address V1 position. The call reverts before moving the position, so funds are not lost, but the advertised migration path cannot consolidate the user's positions and there is no alternative wallet for that account.

The test that appears to cover this case first stakes through the wallet and then successfully imports the raw-address position. It passes because `MockRewardRouterV2.acceptTransfer` omits every receiver-freshness check. An immediate stake can also remain fresh until reward state is recorded, so the test must accrue or claim rewards to exercise the production condition.

RECOMMENDATION

Match V1's receiver validation in the mock and add a test with nonzero reward history. Require and clearly surface that V1 import must occur before the account wallet becomes active. If importing into an active wallet is required, add a fresh-wallet rotation or a separate migration mechanism because V1's transfer API cannot merge into that receiver.

RESOLUTION

Status: Resolved.

LOW **L-02** ACKNOWLEDGED

Staked GLP rewards can block V1 receivers

CATEGORY	DoS
LOCATION	https://github.com/gmx-io/gmx-contracts/blob/cf0d42d0ab89f322adfee10ee409bec82510dcf1/contracts/staking/StakedGlp.sol#L96-L100 https://github.com/gmx-io/gmx-contracts/blob/cf0d42d0ab89f322adfee10ee409bec82510dcf1/contracts/staking/RewardTracker.sol#L296-L302 (https://github.com/gmx-io/gmx-contracts/blob/cf0d42d0ab89f322adfee10ee409bec82510dcf1/contracts/staking/StakedGlp.sol#L96-L100 https://github.com/gmx-io/gmx-contracts/blob/cf0d42d0ab89f322adfee10ee409bec82510dcf1/contracts/staking/RewardTracker.sol#L296-L302)
REVIEW	Remediation Review 2

DESCRIPTION

`StakedGlp.transfer` can stake GLP for any address, including a predicted account wallet that has no code. The GLP trackers checkpoint rewards before adding the new stake. The first transfer finds a zero stake, so `averageStakedAmounts` and `cumulativeRewards` stay zero. A later transfer, after the reward index has moved, writes both fields. Nothing clears them.

V1 `signalTransfer` and `acceptTransfer` require both fields to be zero on the fee and staked GLP trackers. Once they are written, that address can never receive an account transfer, and a predicted account wallet has no second address.

Both live GLP distributors have `tokensPerInterval` of zero, so rewards do not accrue and a second transfer does not write the fields. The denial of service is not exploitable under the current reward configuration. It starts if an admin sets a nonzero rate on either distributor.

RECOMMENDATION

Keep both GLP distributors at a zero emission rate until account migration no longer depends on a fresh V1 receiver.

RESOLUTION

Status: Acknowledged.

LOW L-03

ACKNOWLEDGED

sGLP sweep traps stake on the vault

CATEGORY	Unexpected Behavior
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L285-L297 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/token/TokenUtils.sol#L57-L60 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L285-L297 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/token/TokenUtils.sol#L57-L60)
REVIEW	Remediation Review 2

DESCRIPTION

`withdrawFromWallet` accepts any signed token and has the wallet transfer that amount to the multichain vault. For sGLP, that transfer is `StakedGlp.transfer`: it unstakes the wallet's GLP and restakes it for the vault. The account is then credited sGLP, because `sGLP.balanceOf` reports the fee tracker's GLP deposit. Spending that credit calls `TokenUtils.transfer`, which reverts when the token has no transfer gas limit:

```
uint256 gasLimit = dataStore.getUint(Keys.tokenTransferGasLimit(token));
if (gasLimit == 0) {
    revert Errors.EmptyTokenTranferGasLimit(token);
}
```

The live Arbitrum DataStore `0xFD70de6b91282D8017aA4E741e9Ae325CAb992d8` stores zero for both sGLP wrappers, `0x5402B5F40310bDED796c7D0F3FF6683f5C0cFfdf` and `0x2F546AD4eDD93B956C8999Be404cdCAFde3E89AE`. After the sweep the vault is the staker. The user cannot call V1 `unstakeAndRedeemGlp`, and same chain `transferOut` cannot move the sGLP credit back to an EOA. The position stays on the shared vault.

RECOMMENDATION

Reject both StakedGlp wrappers in `withdrawFromWallet`. The sweep restakes the user's GLP for the vault, and the resulting credit cannot be transferred out while `tokenTransferGasLimit` is zero.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-01 ACKNOWLEDGED

Unused SafeERC20 import and declaration

CATEGORY	Superfluous Code
LOCATION	https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L6 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L27 (https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L6 https://github.com/GuardianOrg/gmx-synthetics-private-team1-1784151983342/blob/a0afb112f915f24de798d12782ce1f1e476c7d75/contracts/multichain/MultichainStakingUtils.sol#L27)
REVIEW	Remediation Review 2

DESCRIPTION

`MultichainStakingUtils` imports `SafeERC20` and declares `using SafeERC20 for IERC20`, but never invokes a SafeERC20 extension.

RECOMMENDATION

Consider to remove the unused SafeERC20 import and `using SafeERC20 for IERC20` declaration.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-01** RESOLVED

Donations checkpoint stale borrowing rates

CATEGORY	Configuration
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/7fbd664f3cbf89c748c0e86f42319aacfbf05259/scripts/roles/rolesToAdd/arbitrum.ts#L309-L320 https://github.com/gmx-io/gmx-synthetics-private/blob/7fbd664f3cbf89c748c0e86f42319aacfbf05259/deployments/arbitrum/MarketPoolDonationHandler.json#L627-L630
REVIEW	Remediation Review 3

DESCRIPTION

The Arbitrum role rollout grants `CONTROLLER` to `MarketPoolDonationHandler` at `0x2AB2B848F64637b5c69BbE42FE72800f2494DAc4`, whose deployed code links the old `MarketUtils` at `0x0b34021E939EDAD7cbaA1ab2016C48Ff796c5029` instead of the release library. The old library omits both the 100% kink utilization clamp and `MAX_BORROWING_FACTOR_PER_SECOND`. `donate` checkpoints borrowing on both sides before adding liquidity, so a successful keeper donation stores the old rate in the shared `DataStore`. The role was active at block `515282859`, and a successful donation batch at block `514683046` emitted nonzero borrowing factor updates.

For example, at 155% utilization, the configured 75% kink, 50% base APR and 130% upper APR yield about 333.5% APR under the old library, versus 130% before the release's separate rate cap. At Arbitrum block `515283068`, that cap read zero on both sides of the three donation markets, so the release library would return zero under those settings. One day at the old rate adds about 0.914% to the cumulative factor, potentially charging unrelated traders excess fees or making their positions liquidatable. This is an incomplete deployment of the fixes for Team 1 M-03 and Team 2 M-06; a specific historical trader loss has not been quantified.

RECOMMENDATION

Pause the affected donations or revoke this handler's `CONTROLLER` role until a handler linked to the release `MarketUtils` is deployed and authorized. Check other authorized borrowing writers for the same old link, and review borrowing factors recorded by successful donations for excess charges.

RESOLUTION

Status: Resolved.

M-01 Donations checkpoint stale borrowing rates CONTINUED

GMX Team: Not a live defect: the donation handler links the same borrowing library as every live contract, since the clamp and cap are v2.2.1 changes not deployed yet, and all 740 donation checkpoints since 2026-10-03 show zero excess (highest usage 63.7%). So no pause or revoke.

Fixed on the deployment side instead: the consumed donation deploy ids are bumped (PR #434), the v2.2.1 rollout redeploys the donation contracts linked to the new MarketUtils, moves the donation keeper to the new batcher at the cutover, revokes the old handlers' CONTROLLER with the rest of the previous stack, and sets MAX_BORROWING_FACTOR_PER_SECOND at the cutover.

LOW L-01

RESOLVED

OFT dust can block a fee distribution

CATEGORY	Business logic errors
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/08f14034a8cec1d5b8d3b75403fe2643ac383b3c/contracts/fee/FeeDistributor.sol#L455-L469 https://github.com/gmx-io/gmx-synthetics-private/blob/08f14034a8cec1d5b8d3b75403fe2643ac383b3c/contracts/fee/FeeDistributor.sol#L524-L549
REVIEW	Remediation Review 3

DESCRIPTION

_bridgeGmx rounds the sent GMX down to the OFT conversion unit, but getGmxAmounts calculates the destination minimum from the undusted deficit. With a 1e12 wei conversion unit, a 1,010,101,010,103 wei deficit and a 99% receipt factor, the source sends 1,000,000,000,000 wei while the destination requires 1,000,000,000,001 wei. Distribution then waits for shortfall tolerance or reset. The checked fee-distribution configuration is inactive, so current production impact was not established.

RECOMMENDATION

Calculate the destination minimum from the amount that can actually be bridged after OFT dust removal.

RESOLUTION

Status: Resolved.

GMX Team: PR #443

LOW L-02 RESOLVED

Refunded compose can consume later bridge funds

CATEGORY	Logical Error
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/08f14034a8cec1d5b8d3b75403fe2643ac383b3c/contracts/multichain/LayerZeroProvider.sol#L448-L457 https://github.com/gmx-io/gmx-synthetics-private/blob/08f14034a8cec1d5b8d3b75403fe2643ac383b3c/contracts/multichain/LayerZeroProvider.sol#L115-L135
REVIEW	Remediation Review 3

DESCRIPTION

`withdrawTokens` can return tokens staged for a failed Stargate compose without invalidating its queued message. The provider holds every pending delivery of a token in one balance, and `lzCompose` spends `amountLD` from that balance before checking the native top-up.

An attacker bridges 100 USDC with a compose that names the attacker and asks for a 1 wei top-up, while leaving the LayerZero compose option value at 0. Stargate pays the 100 USDC to `LayerZeroProvider` and queues the message. The LayerZero executor delivers only the prepaid option value, so a zero-value delivery reverts and the message stays queued. The attacker cannot call `withdrawTokens`. If a `CONTROLLER`, through the timelock, refunds that 100 USDC, the old message remains executable.

Once another user bridges 100 USDC, the attacker delivers the old message through `EndpointV2.lzCompose` with 1 wei. The provider moves the later user's tokens to `MultichainVault` and credits the attacker. The later user's compose stays queued with no tokens left to credit. Deployment and recovery policy for the pinned release were not verified.

RECOMMENDATION

Tie recovery to the authenticated compose and make a refunded message unusable before releasing its backing tokens. Prevent generic recovery from withdrawing balances reserved for queued messages.

RESOLUTION

Status: Resolved.

GMX Team: Recommendation implemented through the refundCompose addition (PR #437).

The generic withdrawTokens stays --> the provider is immutable and funds can reach it with no message behind them (direct transfers, Stargate sends without a compose).

L-02 Refunded compose can consume later bridge funds CONTINUED

For a queued message the procedure is refundCompose, withdrawTokens only for funds no message accounts for, and the timelock signal shows which path is used.

INFORMATIONAL I-01 ACKNOWLEDGED

Verify capped PnL history for positions retained across the executor upgrade

CATEGORY	Upgradeability
LOCATION	https://github.com/gmx-io/gmx-synthetics-private/blob/08f14034a8cec1d5b8d3b75403fe2643ac383b3c/contracts/position/PositionUtils.sol#L348-L370
REVIEW	Remediation Review 3

DESCRIPTION

The currently authorized July decrease executors on Arbitrum and Avalanche apply the trader PnL cap to each close but do not store per-position capped-close history. The staged v2.2.1 executors introduce the realized-history fields and CAPPED_PNL_ALLOWANCE_PER_PRICE together. If a position had a capped partial close under the July code and remains open at cutover, all of those new fields start at zero. Its earlier capped payout cannot be inferred from these new storage fields when a later close is assessed.

RECOMMENDATION

Be aware and document this behavior and consider to before enabling the new executors, check historical capped partial-decrease events against positions that remain open. If any do consider to account for their earlier payouts in a targeted migration or explicitly document how those positions will be treated.

RESOLUTION

Status: Acknowledged.

GMX Team: Checked on 2026-10-19: no open position on arbitrum or avalanche has ever had a capped partial close, so nothing to migrate;

Documented in the code (PR #435), and the check is repeated on activation day before the new executors are enabled.

INFORMATIONAL I-01 RESOLVED

Unused Cast.sol import in FeeDistributor

CATEGORY	Superfluous Code
LOCATION	https://github.com/GuardianOrg/mirror-gmx-io-gmx-synthetics-private-d2878edbeedf5810d4d3d06135818296/blob/d67794a560836fd0a02a05d094ce9d8d327df960/contracts/fee/FeeDistributor.sol#L13
REVIEW	Remediation Review 4

DESCRIPTION

PR #443 moved the only Cast.toBytes32 use from FeeDistributor.sol into FeeDistributorUtils.sol, but FeeDistributor.sol still imports Cast.sol. The import is unused and has no runtime effect.

RECOMMENDATION

Be aware and consider to remove the unused import.

RESOLUTION

Status: Resolved.

§ 05

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.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

$45B+
DIGITAL ASSETS UNDER REVIEW

200+
ENGAGEMENTS COMPLETED

500+
CRITICAL & HIGH FINDINGS REPORTED

Reach us at audits@guardianaudits.com, or read more at **guardian.security**.

This report and its findings are also published at guardian.security/portfolio/gmx/v2-2-1-review-2026-11-26.