

AVANTIS · SECURITY REVIEW

Avantis
V2.

A security review of V2.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
Avantis	V2	Nicholas Chew, Andreas Zottl, Michael Lett, Carlos Polop	July 6 – September 11, 2026	September 11, 2026

§ 00

Contents.

291 pages: overview, confidence ranking, scope, 199 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. 09
	Main Review 1 · 119 findings	P. 17
	Remediation Review · 53 findings	P. 197
	Main Review 2 · 27 findings	P. 258
§ 05	Disclaimer	P. 290
§ 06	About Guardian	P. 291

§ 01

Overview.

Guardian reviewed V2 across 3 review rounds. The final report includes 4 Critical, 18 High, 62 Medium, 66 Low, 49 Informational findings.

AUDIT FIRM	REPOSITORY
Guardian	https://github.com/Avantis-Labs/avantis-contracts
CLIENT	METHODOLOGY
Avantis	Smart contract · Manual · Main Review 1 · Remediation Review · Main Review 2
PROJECT	MAIN REVIEW 1 COMMIT
V2	d74b350cac86
PREPARED BY	REMEDATION REVIEW COMMIT
Nicholas Chew, Andreas Zottl, Michael Lett, Carlos Polop	afda24cfc9e1
AUDIT WINDOW	MAIN REVIEW 2 COMMIT
July 6 – September 11, 2026	254dd8fd1f47
FINAL REPORT	
September 11, 2026	

1.1 Confidence Ranking

1 / 5

Very Low Confidence

Guardian assigns a Confidence Ranking of 1 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	FIX REVIEW
● Critical	4	0	0	0	4	0
● High	18	0	4	0	14	0
● Medium	62	1	35	0	25	1
● Low	66	1	49	0	15	1
● Informational	49	6	39	0	4	0
All severities	199	8	127	0	62	2

§ 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	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 ASSIGNED	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 V2 in `https://github.com/Avantis-Labs/avantis-contracts`. The scope includes 27,116 source lines across 27 files.

ALSO IN SCOPE

EIP7702Template.sol

BuilderCode.sol

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 1/3.

FILE	NSLOC	TOTAL
MAIN REVIEW 1		
src/AvantisTranche.sol	211	403
src/Execute.sol	68	126
src/Multicall.sol	532	838
src/PairInfos/PairInfos.sol	561	976
src/PairInfos/PairInfosExtension.sol	76	123
src/PairStorage/PairStorage.sol	430	868
src/PairStorage/PairStorageExtension.sol	200	309
src/PriceAggregator.sol	465	696
src/Referral.sol	294	505
src/Timelock.sol	10	13
src/Trading/DelegateModule.sol	94	134
src/Trading/LimitTradingModule.sol	249	369
src/Trading/MarketTradingModule.sol	563	718
src/Trading/PositionUpdatesModule.sol	457	601
src/Trading/RfqTradingModule.sol	270	345
src/Trading/TradingRouter.sol	264	543
src/Trading/TwapTradingModule.sol	292	394
src/TradingCallbacks/CallbackRouter.sol	135	297
src/TradingCallbacks/IntentCallbackModule.sol	530	615
src/TradingCallbacks/LimitCallbackModule.sol	486	575
src/TradingCallbacks/MarketCallbackModule.sol	436	510
src/TradingCallbacks/TradeRegistry.sol	333	426
src/TradingStorage/TradingStorage.sol	561	1,203
src/TradingStorage/TradingStorageCoreExtension.sol	567	844
src/TradingStorage/TradingStorageIntentExtension.sol	332	450
src/VaultManager.sol	376	654
src/Whitelists/Whitelists.sol	52	104
REMEDIATION REVIEW		
src/AvantisTranche.sol	235	437
src/Execute.sol	68	126
src/Multicall.sol	507	824
src/PairInfos/PairInfos.sol	575	1,002
src/PairInfos/PairInfosExtension.sol	78	125
src/PairStorage/PairStorage.sol	456	902
src/PairStorage/PairStorageExtension.sol	201	313

Files in Scope 2/3.

FILE	NSLOC	TOTAL
src/PriceAggregator.sol	506	743
src/Referral.sol	294	505
src/Timelock.sol	10	13
src/Trading/DelegateModule.sol	94	134
src/Trading/LimitTradingModule.sol	250	370
src/Trading/MarketTradingModule.sol	557	708
src/Trading/PositionUpdatesModule.sol	475	623
src/Trading/RfqTradingModule.sol	271	346
src/Trading/TradingRouter.sol	280	567
src/Trading/TwapTradingModule.sol	318	425
src/TradingCallbacks/CallbackRouter.sol	135	297
src/TradingCallbacks/IntentCallbackModule.sol	582	673
src/TradingCallbacks/LimitCallbackModule.sol	508	608
src/TradingCallbacks/MarketCallbackModule.sol	447	524
src/TradingCallbacks/TradeRegistry.sol	340	436
src/TradingStorage/TradingStorage.sol	548	1,171
src/TradingStorage/TradingStorageCoreExtension.sol	590	878
src/TradingStorage/TradingStorageIntentExtension.sol	334	454
src/VaultManager.sol	437	751
src/Whitelists/Whitelists.sol	52	104
MAIN REVIEW 2		
src/AvantisTranche.sol	235	437
src/Execute.sol	68	126
src/Multicall.sol	507	824
src/PairInfos/PairInfos.sol	575	1,002
src/PairInfos/PairInfosExtension.sol	78	125
src/PairStorage/PairStorage.sol	456	902
src/PairStorage/PairStorageExtension.sol	201	313
src/PriceAggregator.sol	505	742
src/Referral.sol	294	505
src/Timelock.sol	10	13
src/Trading/DelegateModule.sol	94	134
src/Trading/LimitTradingModule.sol	250	370
src/Trading/MarketTradingModule.sol	565	721
src/Trading/PositionUpdatesModule.sol	470	617
src/Trading/RfqTradingModule.sol	271	346

Files in Scope 3/3.

FILE	NSLOC	TOTAL
src/Trading/TradingRouter.sol	272	552
src/Trading/TwapTradingModule.sol	318	425
src/TradingCallbacks/CallbackRouter.sol	135	297
src/TradingCallbacks/IntentCallbackModule.sol	569	658
src/TradingCallbacks/LimitCallbackModule.sol	508	608
src/TradingCallbacks/MarketCallbackModule.sol	447	524
src/TradingCallbacks/TradeRegistry.sol	340	436
src/TradingStorage/TradingStorage.sol	548	1,171
src/TradingStorage/TradingStorageCoreExtension.sol	590	878
src/TradingStorage/TradingStorageIntentExtension.sol	334	454
src/VaultManager.sol	432	745
src/Whitelists/Whitelists.sol	52	104

§ 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 1					
C-01	Omitted Lazer exponent inflates oracle price	Validation	CRITICAL	RESOLVED	p. 17
C-02	Partial closes erase accrued fees	Logical Error	CRITICAL	RESOLVED	p. 19
C-03	Exponentless Lazer updates let traders mint funding collateral and drain LP assets	Informational	CRITICAL	RESOLVED	p. 22
C-04	Rounded DCA positions poison coin OI and enable a repeatable LP funding drain	Informational	CRITICAL	RESOLVED	p. 23
H-01	Stale TP executions can overpay traders	Logical Error	HIGH	RESOLVED	p. 24
H-02	Alternate close paths misapply PnL fees	Logical Error	HIGH	RESOLVED	p. 27
H-03	Position increase ignore unrealized losses	Validation	HIGH	RESOLVED	p. 29
H-04	Header freshness masks stale Lazer prices	Validation	HIGH	RESOLVED	p. 31
H-05	Referrer rebates can block closes	DoS	HIGH	ACKNOWLEDGED	p. 33
H-06	LP withdrawals can block liquidations	DoS	HIGH	RESOLVED	p. 35
H-07	Direct donations bypass an exhausted deposit cap to forge OI capacity and freeze rollover-dependent settlement	DoS	HIGH	RESOLVED	p. 38
H-08	Repeated dust RFQ closes create an unworkarroundable close-settlement freeze	Informational	HIGH	RESOLVED	p. 39
H-09	Fixed-coin increases bypass leverage limits by counting impaired collateral at nominal value	Informational	HIGH	RESOLVED	p. 40
H-10	A front-run TWAP slice expands a prepared TP into a retroactive LP-funded payout	Informational	HIGH	RESOLVED	p. 42
H-11	One-tick guaranteed-stop PnL straddles create an unpriced LP-funded volatility option	Informational	HIGH	ACKNOWLEDGED	p. 43
H-12	Cross-pair forex hedges retroactively erase target rollover debt	Logical Error	HIGH	ACKNOWLEDGED	p. 44
H-13	Full-sized partial TP bypasses PnL performance fees	Informational	HIGH	RESOLVED	p. 45
H-14	Insolvent one-sided payers manufacture funding liabilities that drain LP principal	Informational	HIGH	ACKNOWLEDGED	p. 46
M-01	Rollover fee divides by zero at max utilization	Error	MEDIUM	RESOLVED	p. 47

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
M-02	TWAP RFQ slices override user size	Validation	MEDIUM	RESOLVED	p. 49
M-03	Rollover config changes apply retroactively	Logical Error	MEDIUM	ACKNOWLEDGED	p. 50
M-04	Stale close callbacks hit new positions	Logical Error	MEDIUM	ACKNOWLEDGED	p. 52
M-05	Pending rewards can be sniped	Gaming	MEDIUM	ACKNOWLEDGED	p. 54
M-06	Market closes ignore wanted price	Validation	MEDIUM	ACKNOWLEDGED	p. 55
M-07	Coin OI rounding blocks full closes	Rounding	MEDIUM	RESOLVED	p. 56
M-08	Liquidations use stale funding price	Logical Error	MEDIUM	RESOLVED	p. 57
M-09	Partial closes can strand dust positions	Logical Error	MEDIUM	ACKNOWLEDGED	p. 58
M-10	Stale prices allow unsafe margin withdrawals	Logical Error	MEDIUM	ACKNOWLEDGED	p. 60
M-11	Dev gov fees can pull tranche funds	Logical Error	MEDIUM	ACKNOWLEDGED	p. 62
M-12	Open TWAP RFQ slices merge into recycled trade	Logical Error	MEDIUM	ACKNOWLEDGED	p. 64
M-13	Legacy delegates retain TWAP cancel access	DoS	MEDIUM	RESOLVED	p. 66
M-14	Unsafe negative Pyth price cast	Validation	MEDIUM	RESOLVED	p. 67
M-15	Most pairs lack effective oracle backups	Oracle	MEDIUM	ACKNOWLEDGED	p. 69
M-16	Intent coin OI bypasses exposure caps	Logical Error	MEDIUM	RESOLVED	p. 71
M-17	Short increases enable triggered SL	Logical Error	MEDIUM	RESOLVED	p. 72
M-18	Guaranteed stops enable cheap straddles	Gaming	MEDIUM	ACKNOWLEDGED	p. 74
M-19	Builder can change fees after signing	Unexpected Behavior	MEDIUM	ACKNOWLEDGED	p. 76
M-20	Position rolls omit funding settlements	Logical Error	MEDIUM	RESOLVED	p. 77
M-21	Backup deposits reset funding at zero	Logical Error	MEDIUM	ACKNOWLEDGED	p. 79
M-22	Stale close notional inflates rebates	Logical Error	MEDIUM	RESOLVED	p. 81
M-23	LPs Can Dodge Trader Wins	Gaming	MEDIUM	ACKNOWLEDGED	p. 83
M-24	Close Intents Consume Open Slots	DoS	MEDIUM	RESOLVED	p. 84
M-25	One-sided funding creates phantom liability	Logical Error	MEDIUM	ACKNOWLEDGED	p. 85
M-26	Pair updates leave usdOI unsynced	Logical Error	MEDIUM	RESOLVED	p. 87
M-27	Coin RFQ opens exceed OI caps and underflow	Informational	MEDIUM	RESOLVED	p. 89
M-28	Stale close fee fabricates profit and erases funded LP PnL rewards	Informational	MEDIUM	RESOLVED	p. 90
M-29	TWAP race cashes replacement exposure out at a pre-existence price	Informational	MEDIUM	ACKNOWLEDGED	p. 91
M-30	Newest Pyth Lazer report can predate RFQ exposure and strip enlarged-position collateral	Informational	MEDIUM	ACKNOWLEDGED	p. 92

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
M-31	RFQ close lets a stale market close cash out a replacement position	Informational	MEDIUM	ACKNOWLEDGED	p. 93
M-32	LP withdrawals retroactively reprice aged rollover and instantly liquidate healthy traders	Informational	MEDIUM	ACKNOWLEDGED	p. 94
M-33	OI-cap reduction retroactively confiscates aged positions' collateral	Informational	MEDIUM	RESOLVED	p. 96
M-34	Unbounded rollover curve freezes at cap and confiscates just below it	Informational	MEDIUM	RESOLVED	p. 97
M-35	Legacy PnL positions create repeatable unfunded funding withdrawals	Informational	MEDIUM	RESOLVED	p. 99
M-36	Stale funding reserve enables ERC4626 share dilution and LP asset theft	Informational	MEDIUM	RESOLVED	p. 100
M-37	Low-leverage longs can promise more profit than the tranche reserves	Informational	MEDIUM	ACKNOWLEDGED	p. 101
M-38	Pause allows OI migration race	Warning	MEDIUM	RESOLVED	p. 102
M-39	Recursive entry-price flooring converts losing DCA legs into LP-funded profit	Informational	MEDIUM	PENDING	p. 103
M-40	Zero-value deposits can move liquidatable positions outside liquidation	Informational	MEDIUM	RESOLVED	p. 104
M-41	Limit orders can open with crossed TP or SL and wipe trader collateral	Informational	MEDIUM	ACKNOWLEDGED	p. 106
L-01	Market fulfill burns intent orders	Validation	LOW	ACKNOWLEDGED	p. 107
L-02	Expired market closes lock exits	DoS	LOW	ACKNOWLEDGED	p. 108
L-03	Margin deposits leave stale max TP	Logical Error	LOW	ACKNOWLEDGED	p. 110
L-04	Position roll disables liquidations	Unexpected Behavior	LOW	ACKNOWLEDGED	p. 112
L-05	Signed TWAP RFQ bypass pause gates	Unexpected Behavior	LOW	RESOLVED	p. 114
L-06	Signed intents cannot settle in backup mode	Unexpected Behavior	LOW	ACKNOWLEDGED	p. 115
L-07	Loss protection is inert	Unexpected Behavior	LOW	ACKNOWLEDGED	p. 116
L-08	ERC4626 tranche first deposit inflation	Warning	LOW	ACKNOWLEDGED	p. 117
L-09	Large payouts hit tranche pull cap	Logical Error	LOW	ACKNOWLEDGED	p. 118
L-10	Clamped TWAP RFQ close overcharges fee	Logical Error	LOW	RESOLVED	p. 119
L-11	Low price TP rounds to entry	Rounding	LOW	ACKNOWLEDGED	p. 120
L-12	Operator spread can set arbitrary prices	Validation	LOW	ACKNOWLEDGED	p. 121
L-13	USDC Blacklisted Traders Are Not Liquidatable	DoS	LOW	ACKNOWLEDGED	p. 123

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
L - 14	Pending opens bypass risk controls	Warning	LOW	ACKNOWLEDGED	p. 124
L - 15	Reserve accounting omits liabilities	Logical Error	LOW	ACKNOWLEDGED	p. 126
L - 16	Validator selection is not signed	Signatures	LOW	ACKNOWLEDGED	p. 128
L - 17	Post execution context can be overwritten	Validation	LOW	ACKNOWLEDGED	p. 129
L - 18	Revoked validator executes in same bundle	Validation	LOW	ACKNOWLEDGED	p. 131
L - 19	Delegate can execute any trader action	Improper access control	LOW	ACKNOWLEDGED	p. 132
L - 20	Nested actions bypass builder fees	Logical Error	LOW	RESOLVED	p. 133
L - 21	Zero target bypasses validator policies	Validation	LOW	ACKNOWLEDGED	p. 135
L - 22	Dust closes avoid sweep taker fee	Logical Error	LOW	ACKNOWLEDGED	p. 137
L - 23	Backup mode consumes close orders	Logical Error	LOW	ACKNOWLEDGED	p. 139
L - 24	Bad leverage bounds lock increases	Validation	LOW	RESOLVED	p. 140
L - 25	Delegations compose across cancels	Access Control	LOW	ACKNOWLEDGED	p. 142
L - 26	Pooled ETH funds underpaid Pyth updates	Oracle	LOW	ACKNOWLEDGED	p. 144
L - 27	Forced close corrupts pair funding	Oracle	LOW	ACKNOWLEDGED	p. 145
L - 28	RFQ close slices uncapped against authorized	Informational	LOW	ACKNOWLEDGED	p. 147
L - 29	Collateral TWAP slices retain stale leverage after a live risk-cap reduction	Informational	LOW	ACKNOWLEDGED	p. 148
L - 30	Signed partial TP/SL closes stale-basis positions during active rolls	Informational	LOW	RESOLVED	p. 149
L - 31	Unchecked fixed-collateral TWAP leverage can extract LP principal	Informational	LOW	RESOLVED	p. 150
L - 32	Position rolls bypass OI caps and can freeze pair-wide settlement	Informational	LOW	ACKNOWLEDGED	p. 151
L - 33	Persistent phantom PnL rewards let LPs extract future traders' collateral	Informational	LOW	ACKNOWLEDGED	p. 153
L - 34	Reentrant consecutive nonces bypass validator lifetime limits and drain the account	Informational	LOW	ACKNOWLEDGED	p. 155
L - 35	Missed funding snapshots waive fees	Warning	LOW	ACKNOWLEDGED	p. 156
L - 36	Coin OI batches can split pair funding	Warning	LOW	ACKNOWLEDGED	p. 157
L - 37	Risk manager forges funding rebates	Validation	LOW	RESOLVED	p. 158
I - 01	PnL liquidations skip vault fee	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 161
I - 02	Stale decrease size order type	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 163
I - 03	Voluntary closes forfeit liquidation buffer	Unexpected Behavior	INFORMATIONAL	ACKNOWLEDGED	p. 164

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
I-04	overwriteTrade silently skips missing slots	Warning	INFORMATIONAL	ACKNOWLEDGED	p. 166
I-05	Update mode writes Pyth state unnecessarily	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 167
I-06	RWA oracle states lack an action policy	Suggestion	INFORMATIONAL	ACKNOWLEDGED	p. 168
I-07	Tranche pull cap is per call only	Warning	INFORMATIONAL	ACKNOWLEDGED	p. 171
I-08	Multicall upgrade bricks trade settlement	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 172
I-09	Avantis price source trusts publisher	Trust Assumptions	INFORMATIONAL	ACKNOWLEDGED	p. 174
I-10	Builder code hijack diverts user trade fees	Improper access control	INFORMATIONAL	ACKNOWLEDGED	p. 176
I-11	Extreme Funding Freezes Settlement	Validation	INFORMATIONAL	ACKNOWLEDGED	p. 178
I-12	Roll Keeper Forges Entry Price	Trust Assumptions	INFORMATIONAL	ACKNOWLEDGED	p. 180
I-13	TWAP Schedule Can Be Bypassed	Business logic errors	INFORMATIONAL	ACKNOWLEDGED	p. 182
I-14	Close slices exceed authorized amount	Trust Assumptions	INFORMATIONAL	ACKNOWLEDGED	p. 184
I-15	Forex skew test misses production branch	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 186
I-16	FOREX util skips category blend	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 187
I-17	Returning governance source can replay a consumed upgrade and downgrade the Pyth proxy	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 189
I-18	Retired PnL positions manufacture funding credits that drain LP principal	Informational	INFORMATIONAL	RESOLVED	p. 190
I-19	Signed TP/SL update for a closed position can destroy a replacement at the reused index	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 191
I-20	Cross-mode raw-flow netting can turn an unfunded maker credit into same-price LP loss	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 192
I-21	Partial guaranteed stop losses settle at the gap price	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 194
I-22	Collateral increases can retroactively unlock previously capped PnL	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 195
I-23	Per-holder rollback bypasses reserve limits before aggregate OI handoff	Informational	INFORMATIONAL	ACKNOWLEDGED	p. 196

REMEDIATION REVIEW

H-01	Floored DCA entry mints long close PnL	Rounding	HIGH	RESOLVED	p. 197
M-01	Negative ratios disable withdrawal fee	Math	MEDIUM	ACKNOWLEDGED	p. 200
M-02	Abandoned funding uses settlement-time OI	Logical Error	MEDIUM	ACKNOWLEDGED	p. 201
M-03	TWAP and RFQ closes use mutable fee mode	Logical Error	MEDIUM	ACKNOWLEDGED	p. 203
M-04	Queued callbacks bypass the pair roll freeze	Logical Error	MEDIUM	RESOLVED	p. 204

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
M-05	Cross-pair forex hedges retroactively erase target rollover debt	Informational	MEDIUM	REPORTED	p. 205
M-06	Preassigned codes bypass private approval	Validation	MEDIUM	ACKNOWLEDGED	p. 206
M-07	Coin intents exceed leverage caps after fees	Validation	MEDIUM	ACKNOWLEDGED	p. 207
M-08	Borrow caps retroactively reprice positions	Logical Error	MEDIUM	RESOLVED	p. 208
M-09	Split opens reduce nonlinear spread	Gaming	MEDIUM	ACKNOWLEDGED	p. 209
M-10	Legacy loss protection covers added notional	Gaming	MEDIUM	ACKNOWLEDGED	p. 210
M-11	Increases rebase TP/SL at old leverage	Logical Error	MEDIUM	RESOLVED	p. 211
M-12	Legacy positions pay no borrow fees	Logical Error	MEDIUM	ACKNOWLEDGED	p. 213
M-13	Margin withdrawals can manufacture retroactive guaranteed stops	Logical Error	MEDIUM	ACKNOWLEDGED	p. 214
M-14	Early price-loss clamp lets funding receivers extract LP principal near liquidation	Logical Error	MEDIUM	ACKNOWLEDGED	p. 215
M-15	Equal-time Pyth cache pinning creates retroactive guaranteed stop-loss insurance	Logical Error	MEDIUM	RESOLVED	p. 216
M-16	Insolvent closes reclass fees as profit	Logical Error	MEDIUM	ACKNOWLEDGED	p. 217
L-01	ABIs misdeclare delegaterecall-emitted events	Informational	LOW	RESOLVED	p. 219
L-02	Forced PnL closes apply the regular spread	Logical Error	LOW	RESOLVED	p. 220
L-03	Zero-cap groups pass validation	Validation	LOW	RESOLVED	p. 221
L-04	LimitExecuted emits price as limitIndex	Events	LOW	ACKNOWLEDGED	p. 222
L-05	Final full withdrawal reverts at zero assets	DoS	LOW	ACKNOWLEDGED	p. 223
L-06	USDC blacklist strands roll limit orders	DoS	LOW	ACKNOWLEDGED	p. 224
L-07	Operator outage strands open-RFQ collateral	DoS	LOW	ACKNOWLEDGED	p. 225
L-08	Dust full closes retain maker fee tier	Logical Error	LOW	ACKNOWLEDGED	p. 226
L-09	Legacy Trading retains cancellation access	Configuration	LOW	ACKNOWLEDGED	p. 227
L-10	Signed increases override pending full exits	Warning	LOW	ACKNOWLEDGED	p. 228
L-11	Wrapped short SL becomes immediately active	Unexpected Behavior	LOW	REPORTED	p. 230
L-12	Newer Pyth updates can flip partial TPs into losses	Informational	LOW	RESOLVED	p. 232
L-13	Zero-amount market closes cannot execute	Validation	LOW	ACKNOWLEDGED	p. 233
L-14	getTradeValuePure wraps negative net fees	Logical Error	LOW	ACKNOWLEDGED	p. 234
L-15	Stale consent overwrites referral choice	Gaming	LOW	ACKNOWLEDGED	p. 235
L-16	ERC-4626 limits ignore the deposit cap	Validation	LOW	RESOLVED	p. 236

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
L-17	SL correction can install a retroactive guaranteed stop across the current market	Logical Error	LOW	ACKNOWLEDGED	p. 237
I-01	Declared output is never returned	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 238
I-02	Rollover multiplier drops at full utilization	Math	INFORMATIONAL	ACKNOWLEDGED	p. 239
I-03	Intent closes permit stale settlement	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 240
I-04	Installed Pyth Lazer parser rejects timestamp	Configuration	INFORMATIONAL	ACKNOWLEDGED	p. 241
I-05	Risk-manager rotation script omits aggregator	Configuration	INFORMATIONAL	ACKNOWLEDGED	p. 242
I-06	Multicall rollover quotes diverge from fees	Logical Error	INFORMATIONAL	RESOLVED	p. 243
I-07	Funding updater strands keeper oracle fees	DoS	INFORMATIONAL	ACKNOWLEDGED	p. 244
I-08	Zero-fee orders operator cleanup	Gas Griefing	INFORMATIONAL	ACKNOWLEDGED	p. 245
I-09	Funding updates bypass backup-only mode	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 246
I-10	updatePair can zero a live pair cap	DoS	INFORMATIONAL	RESOLVED	p. 247
I-11	Permissionless validation spends contract ETH	Access Control	INFORMATIONAL	ACKNOWLEDGED	p. 248
I-12	Migration traps live FET/USD positions	Logical Error	INFORMATIONAL	RESOLVED	p. 249
I-13	Margin deposits realize unrelated fee state	Warning	INFORMATIONAL	ACKNOWLEDGED	p. 250
I-14	Dust margin updates erase fee fractions	Rounding	INFORMATIONAL	ACKNOWLEDGED	p. 252
I-15	Inactive fulfillments retain native value	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 253
I-16	Protocol rounding in favor of the user	Rounding	INFORMATIONAL	ACKNOWLEDGED	p. 254
I-17	Position rolls distort flow-based price impact	Logical Error	INFORMATIONAL	ACKNOWLEDGED	p. 255
I-18	maxWithdraw overstates fee-bearing exits	Validation	INFORMATIONAL	ACKNOWLEDGED	p. 256
I-19	Lowered cap hides high-index positions	Configuration	INFORMATIONAL	ACKNOWLEDGED	p. 257

MAIN REVIEW 2

H-01	Coin close wanted price inflates fee	Business logic errors	HIGH	RESOLVED	p. 258
H-02	Delegate can drain fee allowance	Access Control	HIGH	RESOLVED	p. 259
H-03	Wrapped margin withdrawal mints collateral	Math	HIGH	RESOLVED	p. 260
M-01	Signed max rate follows live cap	Signatures	MEDIUM	ACKNOWLEDGED	p. 262
M-02	Market cancellations retain builder fees	Unexpected Behavior	MEDIUM	ACKNOWLEDGED	p. 263
M-03	Deleted codes can be reclaimed	Access Control	MEDIUM	RESOLVED	p. 264
M-04	Breaker pause blocks liquidations	DoS	MEDIUM	ACKNOWLEDGED	p. 265
M-05	Changing decay factor reprices elapsed volume	Math	MEDIUM	ACKNOWLEDGED	p. 266
L-01	Guaranteed SL accepts prerequest prices	Oracle	LOW	ACKNOWLEDGED	p. 267

Findings continued.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
L-02	Lowered cap leaves old rates active	Configuration	LOW	ACKNOWLEDGED	p. 268
L-03	TWAP and RFQ fees are unreachable	Unexpected Behavior	LOW	RESOLVED	p. 269
L-04	Delegate expiry mismatch on fees	Access Control	LOW	RESOLVED	p. 270
L-05	Withheld payouts remain redeemable by LPs	Logical Error	LOW	ACKNOWLEDGED	p. 271
L-06	Calls decoded from unsigned signature bytes	Signatures	LOW	RESOLVED	p. 273
L-07	Pull breaker excludes live margin payouts	Trust Assumptions	LOW	ACKNOWLEDGED	p. 275
L-08	Rebates remain payable after breaker trip	Logical Error	LOW	ACKNOWLEDGED	p. 277
L-09	Long coin orders can inflate builder fees	Gaming	LOW	ACKNOWLEDGED	p. 278
L-10	Builder fees can claim untagged orders	Access Control	LOW	ACKNOWLEDGED	p. 280
L-11	Coin RFQs can finish below signed target	Business logic errors	LOW	PENDING	p. 281
L-12	Coin TWAPs do not complete at their target	DoS	LOW	ACKNOWLEDGED	p. 282
I-01	Delegate return is double encoded	Compatibility	INFORMATIONAL	PENDING	p. 283
I-02	Sub-1% PnL distributions revert	Rounding	INFORMATIONAL	PENDING	p. 284
I-03	Pull breakers default to zero	Configuration	INFORMATIONAL	ACKNOWLEDGED	p. 285
I-04	Position rolls bypass leverage limits	Informational	INFORMATIONAL	PENDING	p. 286
I-05	Partial closes leave phantom pair coin OI	Rounding	INFORMATIONAL	PENDING	p. 287
I-06	Terminal closes strand group OI reserves	Rounding	INFORMATIONAL	PENDING	p. 288
I-07	Fees can strand fixed-coin market opens	Validation	INFORMATIONAL	PENDING	p. 289

CRITICAL C-01

RESOLVED

Omitted Lazer exponent inflates oracle price

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/PriceAggregator.sol#L747>, <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/PriceAggregator.sol#L721>

REVIEW Main Review 1

DESCRIPTION

`_parseFeedProperties` never asserts that Price, Confidence, and Exponent are all present. A caller can subscribe to Lazer with only `[price, confidence]`; the live signer signs this payload and the deployed Base `PythLazer` accepts it. Since `data.exponent` remains zero, Avantis scales a normal `-8` feed by `1e10` instead of `1e10 / 1e8`, inflating price and confidence by `1e8`.

```

` ` plain text
} else {
uint32 absExponent = uint32(uint16(-data.exponent));
price = (uint64(data.rawPrice) * _PRECISION) / 10 ** absExponent;
conf = (data.rawConf * _PRECISION) / 10 ** absExponent;
}
`

```

The confidence check still passes because `conf/price` is unchanged. The Chainlink comparison is skipped when the backup is missing or more than two minutes old.

The trader can exploit this without keeper participation through `updateMargin(DEPOSIT)`. This entrypoint consumes trader supplied price data synchronously. Funding accrual is linear in `currentPrice`; when the trader is on the funding receiving side, the inflated price produces a large negative `fundingFees`. `_calculateNewLeverage` subtracts that negative fee and records it as collateral. The deposit branch has no PnL or maximum profit check, and the attacker can tune the skew so the resulting leverage remains above the pair minimum. The callback then snapshots funding at the manipulated price, allowing the attacker to close at an honest price and withdraw the fabricated collateral from the vault.

The PoC creates a `130 USDC` long and a `100 USDC` attacker short at `50,000`. After one block, the attacker deposits `1 USDC` using a no exponent payload. The fabricated funding credit is `378.646834 USDC`, raising collateral from `100` to `479.646771 USDC`. An honest close at the unchanged `50,000` price pays `478.646271 USDC`.

The fabricated credit scales with attacker collateral and can be repeated across affected pairs, allowing aggregate withdrawals to exhaust all available protocol and LP liquidity. V1.5 has a separate permissionless attack path through its public `updateSl` and `updateTpAndSl` functions. On a guaranteed SL pair, an underwater long can submit the no exponent

C-01 Omitted Lazer exponent inflates oracle price

CONTINUED

payload so `updateSLCallback` compares the requested SL against the inflated price and accepts an already crossed, near entry stop. For example, after a 100x long opened at `50,000` falls to `49,700`, the trader can retroactively set an SL at `49,999`. The honest keeper sees `49,700 <= 49,999` and executes it, while the guaranteed SL callback settles at `49,999`, reducing an approximately 60% accrued loss to approximately 0.2% before fees. The stored SL also causes liquidation to revert with `HAS_SL`. Repeating this near risk free long strategy across scaled positions and affected pairs can exhaust vault liquidity.

— RECOMMENDATION

Track which properties were parsed and revert unless Price, Confidence, and Exponent are each present exactly once. Alternatively, use `PythLazerLib.parseUpdateFromPayload` and its presence helpers. Never treat an absent exponent as zero.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/224cfa2f9326895e7356ff17b1ac68dc6db6e4aa#diff-2dcffb08e28fb31b32ca3304e20aa36f099698ecc2c37e8f11563e18cd2ab588>

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

CRITICAL C-02

RESOLVED

Partial closes erase accrued fees

CATEGORY	Logical Error
LOCATION	PairInfos.sol
REVIEW	Main Review 1

— DESCRIPTION

C-02 Partial closes erase accrued fees

CONTINUED

Partial-close settlement charges the full position's accrued borrowing and funding fees against only the collateral slice being closed. `getTradeValue()` calculates borrowing fees from `_trade.initialPosToken` and funding fees from the position's full coin size, but passes the smaller `collateral` slice into payout calculation:

After settlement, `unregisterTrade()` reduces the live position by only the closed slice and resets both fee baselines. A trader can therefore wait for fees to accrue, close the minimum permitted collateral slice, and erase the remaining position's much larger fee debt. In the reproduced case, closing `0.100001 USDC` reset `63.417871 USDC` of borrowing fees and `12.689999 USDC` of funding fees while the remaining collateral decreased by only `0.100001 USDC`.

The signed partial TP/SL path makes this cheaper because `executePartialTpSl()` accepts `coinSize = 0`. `_transformTpSlIntent()` turns it into zero collateral, but the callback still calls `_unregisterTrade()` and resets the full position's fee baselines. A trader can therefore erase all accrued fees without closing any collateral.

RFQ closes allow the same reset with a nonzero size. Neither the order nor slice enforces a minimum `coinSize`, so conversion can round it to zero collateral. Settlement then leaves the position unchanged but resets its full fee baselines, while `coinSizeClosed` still advances and may complete the RFQ.

When fees exceed the slice, `getTradeValuePure()` returns no payout and reclassifies the excess as positive PnL. Settlement then credits borrowing rewards and consumes PnL rewards without reducing the remaining position's collateral claim, causing lost fee revenue and inconsistent vault reward accounting.

```
``plain text
// getTradeValuePure()
if (value <= (int256(collateral) * int256(100 - liqThreshold)) / 100) {
value = lossProtectedPnl - pnl;
//@audit if fees exceed collateral, this accounting hack becomes positive which is wrong
lossProtectedPnl = fees - int256(collateral) + value;
}
``
```

— RECOMMENDATION

Reject zero-size partial TP/SL intents. For RFQs, enforce minimum order and slice sizes and require nonzero converted collateral. On partial closes, charge only proportional accrued fees, or deduct all realized fees from the remaining collateral before resetting snapshots. Never reset snapshots or advance close progress when nothing was removed.

C-02 Partial closes erase accrued fees

CONTINUED

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/f5fba1e63f7021bdbaaad422e258aa69b6266a5e>

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

CRITICAL C-03

RESOLVED

Exponentless Lazer updates let traders mint funding collateral and drain LP assets

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_parsePythonLazerPrice():L702-L723 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_parseLazerFeeds():L726-L745 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_parseFeedProperties():L747-L767 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L483 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccFundingFees():L963-L992
REVIEW	Main Review 1

DESCRIPTION

PriceAggregator treats locating the configured Lazer feed as sufficient and does not require Price, Exponent, and Confidence to be present in that feed. A valid Price-only response therefore leaves exponent and confidence at zero, inflating an ordinary -8 price by 1e8 while passing the confidence check. Unlike operator-mediated market fulfillment, updateMargin lets a position owner synchronously submit the malformed signed response. The callback applies that inflated endpoint to elapsed pair-wide funding, converts negative funding into stored receiver collateral, and checkpoints the poisoned accumulator. Impact: A trader can maintain nearly balanced payer/receiver positions, wait for ordinary funding accrual, and use a Price-only signed Lazer response on a nominal margin deposit to manufacture receiver collateral far beyond the payer's bounded collateral. A normal close realizes that credit from VaultManager and the tranche. The end-to-end PoC creates over \$20,000 of collateral from a \$1 deposit and proves more than \$15,000 of LP loss; the exposure-bounded cycle can be repeated against tranche liquidity without compromising an operator, publisher, or governor. This is a permissionless full-fund-loss vector on eligible no-backup/stale-backup Lazer pairs.

RECOMMENDATION

Parse each feed into fresh state and track explicit property-presence flags. Accept the configured feed only when Price, Exponent, and Confidence are all present in that same feed; reject missing or duplicate mandatory fields. Use the bundled canonical parser where possible and safely consume recognized extra properties. Add missing-field, multi-feed, and trader-driven margin/funding regression tests.

RESOLUTION

Status: Resolved.

CRITICAL C-04

RESOLVED

Rounded DCA positions poison coin OI and enable a repeatable LP funding drain

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:_computeNewOpenPrice():L432-L442 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:_updateOpenInterestUSDC():L723-L746 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccFundingFees():L963-L992 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L483
REVIEW	Main Review 1

DESCRIPTION

Increase legs add independently rounded fee-net coin exposure to aggregate OI, while a merged trade's exposure is later reconstructed from a repeatedly truncated harmonic entry price. Repeated increases on realistically low-priced assets can make that reconstructed amount materially exceed the trade's booked contribution. Its full close then uses the side-wide clamp to consume nearly all coin OI belonging to another live same-side trade. Funding divides the pair-wide transfer by this poisoned near-zero side aggregate but multiplies the accumulator by the surviving trade's unchanged reconstructed coin size. A zero-value margin deposit crystallizes the amplified credit, and a withdrawal pays it out while resetting the snapshot and leaving the poisoned denominator intact. The revised passing PoC uses $\backslash\$0.0000045$ - $\backslash\$0.0000065$ price magnitudes, matching Pyth/backup values, nonzero open/close fees, minimum-size legs, 10x leverage, and a production-style funding rate. After a 452-leg setup, 40 real margin cycles extract over $\backslash\$10$, leave attacker-controlled wallets net profitable after setup costs, exhaust manager cash, and reduce tranche assets. Impact: An unprivileged trader can pay a one-time setup cost and repeatedly withdraw funding credits far exceeding opposing funding debt. The poisoned denominator persists across settlements, allowing extraction to continue into LP tranche assets and ultimately threaten protocol solvency. The surviving position's ordinary full-close paths also underflow, leaving it trapped.

RECOMMENDATION

Persist canonical per-trade coin OI, blend entry price from that stored exposure, and remove only the exact recorded contribution on partial/full close.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-01**

RESOLVED

Stale TP executions can overpay traders

CATEGORY	Logical Error
LOCATION	https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/LimitCallbackModule.sol#L191
REVIEW	Main Review 1

— DESCRIPTION

H-01 Stale TP executions can overpay traders

CONTINUED

`executeLimitCloseOrderCallback` uses the TP/SL trigger predicate only to calculate the operator reward. It does not gate settlement. On guaranteed-SL pairs, it instead selects the trade's current stored TP as the settlement price before calculating PnL:

```
``plain text
values.price = guaranteedSlEnabled
? pendingOrder.orderType == ITradingStorage.LimitOrder.TP
? existingTrade.tp
: pendingOrder.orderType == ITradingStorage.LimitOrder.SL
? profitLockingSl ? pricePostSpread : existingTrade.sl
: pricePostSpread
: pricePostSpread;
`
```

The current keeper design makes this reachable without a malicious operator. The central limit order service detects a TP using a snapshot containing the trigger price, open timestamp, TP, and SL. It sends that context to command execution, but `CommandExecutionGrpcController` explicitly drops those fields. `LimitOrderCommandHandlerService` then checks only that a position still exists before scheduling `executeLimitOrder`. Neither layer verifies that the position or TP is unchanged.

A trader can therefore race the queued TP execution with a legitimate signed TP update or trader-initiated position increase. If the update settles first, the callback reloads the changed trade and uses its new TP even though the attached oracle price reached only the old TP.

For example:

- A long opens at 2000 with TP 2400, and the keeper dispatches a TP execution when the oracle reaches 2400.
- Before the queued execution settles, the trader increases the position at 2400. `normalizeTpSl` moves the TP to 2618.18 while preserving its profit percentage.
- The stale TP execution arrives with oracle price 2400. The trigger predicate for the new TP fails, so the operator receives no reward, but the close still settles at 2618.18.
- The trader has earned approximately 100% profit at the oracle price but is credited approximately 200%; the vault funds the difference.

— RECOMMENDATION

Before selecting the execution price or calculating PnL, require the raw oracle price to cross the current stored TP or SL. Apply the same check to full and partial TP/SL callbacks.

Also preserve the keeper's trigger price, open timestamp, and trade state through command execution and revalidate them immediately before scheduling or broadcasting the transaction.

H-01 Stale TP executions can overpay traders

CONTINUED

Kartik:

With V2 we expect BE to work in tandem with contracts on risk validations. We don't want to duplicate checks on both BE and contracts. The expectation is BE would refetch the TP as final validation just before submitting TP closes. In worst case we can add a check on contracts. Also please note we would be not using full position TP/SL in v2. BE would only use partial TP/SL flow. Even full position TP would also execute via this method.

PROOF OF CONCEPT <https://gist.github.com/nicholas-guardian/06c9c592e015a43e209951728d52dff6>

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-02**

RESOLVED

Alternate close paths misapply PnL fees

CATEGORY	Logical Error
LOCATION	https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/Trading/LimitTradingModule.sol#L314 , https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/IntentCallbackModule.sol#L216
REVIEW	Main Review 1

DESCRIPTION

Avantis stores whether each live position uses PnL-based fees in `openPosType`, but several alternate close callbacks do not read that value:

- Signed partial TP/SL orders always use `LIMIT_PARTIAL_CLOSE`, whose aggregator dispatch hardcodes `isPnl = false`.
- TWAP and RFQ close callbacks use `isPnlOrderTypeAllowed(pairIndex)`, which describes the pair's current capabilities rather than the live position's stored fee mode.

```

`plain text
} else if (r.orderType == OrderType.LIMIT_PARTIAL_CLOSE) {
c.executePartialTpSlCallback(orderId, price, false, _spreadParams);
}
`

```

Consequently, a profitable PnL-fee position can close through signed partial TP/SL intents and pay the ordinary maker/taker close fee. This is not limited to small partial exits: the signed intent supplies `coinSize`, and `_transformTpSlIntent` maps any `coinSize` greater than or equal to the live position size to the full remaining collateral:

```

`plain text
_newTrade.initialPosToken = _coinSize >= storageT.positionSizeInCoin(...)
? _userTrade.initialPosToken
: _coinSize.usdcValueAtPrice(_userTrade.openPrice).unscaleByLeverage(_userTrade.leverage);
`

```

PnL trades are designed to charge fees from profitable positions. This path lets those positions route a full close through non-PnL settlement, so protocol and vault fee revenue that should be collected from profitable PnL closes is repeatedly bypassed.

The same mismatch applies in the opposite direction for TWAP and RFQ closes: callbacks infer fee mode from the pair's current `isPnlOrderTypeAllowed` flag instead of the position's stored mode. A standard-fee position can therefore be settled as a PnL-fee position, causing losing closes to pay no LP close fee because `getPnlBasedFee()` returns zero for negative profit, while profitable closes can be charged the wrong fee schedule.

```

`plain text

```

H-02 Alternate close paths misapply PnL fees

CONTINUED

```
bool isPnl = aggregator.pairsStorage().isPnlOrderTypeAllowed(state.pairIndex);
...
```

RECOMMENDATION

Load `getPosType(trader, pairIndex, index)` from the live position in every close callback and pass that stored value into settlement. Do not infer a position's fee mode from its close order type or from whether the pair supports PnL orders.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-03**

RESOLVED

Position increase ignore unrealized losses

CATEGORY Validation

LOCATION MarketCallbackModule.sol, IntentCallbackModule.sol

REVIEW Main Review 1

DESCRIPTION

Position increases enforce maximum leverage against stored collateral, not current equity:

```
`` vb.net
```

```
checked leverage = total OI / stored collateral
```

```
economic leverage = total OI / (stored collateral + unrealized PnL - fees)
```

```
`
```

```
_computeIntermediateDetails() deducts borrow and funding fees, but not unrealized PnL.
```

```
_computeNewLeverage() then treats the economically lost collateral as if it were still available margin:
```

```
`` plain text
```

```
uint256 newUSDOl = _trade.initialPosToken.scaleByLeverage(_trade.leverage) +
```

```
_newTrade.initialPosToken.scaleByLeverage(_newTrade.leverage);
```

```
uint256 collateral = _trade.initialPosToken + _newTrade.initialPosToken;
```

```
newLeverage = newUSDOl * _PRECISION / collateral;
```

```
``
```

The callback only checks this nominal result against `pairMaxLeverage`. It never calculates current PnL or verifies that the final position is above maintenance margin. The coin exposure path is especially permissive because it does not bound the added slice's leverage, only the merged nominal leverage.

Example:

- A 10x long has 100 stored collateral, 1,000 OI, and an open price of 100.

- At a raw price of 91.7, its 10 bps close price is 91.6083. Its loss is 83.917, just below the 85 liquidation threshold, leaving only 16.083 of current equity.

- The trader adds 9,000 OI through a PnL fee coin exposure increase with negligible added collateral.

- Avantix calculates about 100x leverage, $10,000 / 100$, which passes a 150x pair limit. The position's economic leverage is about 622x, $10,000 / 16.083$.

- After the added slice's entry spread and the position's closing spread, its loss is about 101.9. The position is liquidatable at the same raw oracle price.

A distressed trader can therefore reuse collateral already consumed by losses to take much

H-03 Position increase ignore unrealized losses**CONTINUED**

more exposure against LPs. The trader retains the upside of the enlarged position while risking only its small remaining equity, and the protocol must win the liquidation race to prevent that exposure. The current keeper performs no compensating position health check.

— RECOMMENDATION

Before committing an increase, calculate current equity using the spread adjusted close price, unrealized PnL, borrow and funding fees, and closing and liquidation fees. Enforce maximum leverage against that equity and require the final position to remain above maintenance margin with an execution buffer. Apply the same check to both increase paths and intent slices.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/guardian-pocs/test/guardian-pocs/H03PositionIncreaseIgnoreUnrealizedLoss.t.sol>

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-04**

RESOLVED

Header freshness masks stale Lazer prices

CATEGORY Validation

LOCATION `src/PriceAggregator.sol`, `\`_pythLazerPrice`, `\`_validateTimestamp`, and `\`parseFeedProperties`

REVIEW Main Review 1

DESCRIPTION

Pyth Lazer's payload header contains `timestampUs`, while each feed has a separate `feedUpdateTimestamp` describing when its price was actually generated. Since March 23, 2026, Pyth carries the last price forward during market closures. It also carries prices forward when too few publishers contribute. Pyth therefore instructs production consumers to request and validate `feedUpdateTimestamp` ([payload semantics](https://docs.pyth.network/price-feeds/pro/payload-reference#feedupdatetimestamp), [subscription guidance](https://docs.pyth.network/price-feeds/pro/subscribe-to-prices#2-configure-subscription-parameters)).

Avantis instead validates only the fresh header timestamp and then returns the selected feed's price:

```
``plain text
(bytes memory payload, ) = pythLazer.verifyUpdate{ value: pythLazer.verification_fee() }
(priceUpdateData[0]);
(uint64 timestamp,, uint8 feedsLen, uint16 pos) = PythLazerLib.parsePayloadHeader(payload);

_validateTimestamp(timestamp, r.orderType);
return _parsePythLazerPrice(payload, feedsLen, pos, r.pairIndex);
``
```

`_parseFeedProperties` accepts only price, confidence, and exponent. It neither requests nor parses `feedUpdateTimestamp`, `marketSession`, or `publisherCount`. A recently signed bundle can therefore contain a price and confidence value carried forward for hours or days and still pass every Lazer check. The Chainlink comparison does not reliably help because Avantis skips it when the backup feed is stale, which is expected during the same market closure.

The trader can consume this price directly through `updateMargin`. The callback uses it to calculate unrealized PnL and approve a collateral withdrawal.

For example, a trader opens a `100 USDC`, `10x RWA` long at `100`. During the weekend, adverse news indicates a likely reopen near `95`. The trader submits a fresh Lazer bundle carrying Friday's `100` price and withdraws `90 USDC`, raising the position to `100x`. The callback sees zero loss and pays the withdrawal. When a fresh `95`` price arrives, the remaining

H-04 Header freshness masks stale Lazer prices

CONTINUED

10 USDC is exhausted, although the position should have borne a 50 USDC loss before the withdrawal. The vault absorbs the difference. Opposing positions can similarly be reduced toward maximum leverage during the closure, making the attack less dependent on predicting direction.

— RECOMMENDATION

Request `feedUpdateTimestamp` in every signed EVM payload and parse it with the Pyth Lazer SDK. Reject the payload if the property is absent, after the header timestamp, or older than the action's maximum age. For actions that require a newly generated price, require `feedUpdateTimestamp == timestampUs`. Parse `marketSession` as an additional market hours gate, but do not use it as a substitute for feed freshness. When a feed is stale or its market is closed, permit margin deposits but block withdrawals, exposure increases, normal settlement, and trigger execution until a fresh feed update is available.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-05**

ACKNOWLEDGED

Referrer rebates can block closes

CATEGORY	DoS
LOCATION	https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/VaultManager.sol#L316
REVIEW	Main Review 1

— DESCRIPTION

Close settlement pays the dev and gov fee share and the trader payout through `VaultManager.sendUSDCToTrader()`, which can pull a shortfall from the junior tranche. It then pays the referrer rebate through `sendReferrerRebateToStorage()`, which performs a raw transfer from the current VaultManager balance and cannot pull from the tranche.

`sendUSDCToTrader()` computes `balanceAvailable = balance - totalRewards - fundingOwed` and withdraws any shortfall from the tranche before transferring. `sendReferrerRebateToStorage()` skips that solvency logic and directly transfers USDC to `TradingStorage`. After a large profitable close drains the liquid VaultManager buffer, a later referred non-PnL close can therefore revert on the rebate transfer even though trader payouts would otherwise be backed by tranche liquidity.

Illustration:

```
```vb.net
```

VaultManager liquid USDC: 0

Close trader payout: 100

Dev and gov fee share: 10

Referrer rebate: 2

Dev/gov transfer pulls 10 from tranche.

Trader payout pulls 100 from tranche.

Referrer rebate raw transfer tries to send 2 from VaultManager.

The transfer reverts, so the close does not settle.

```
```
This blocks normal market closes, TP/SL closes, and liquidations that carry a referrer rebate because all route through the same unregister path. Positions can remain open when users or keepers attempt to exit or liquidate them, allowing bad debt risk to grow during stressed conditions.
```

If VaultManager has raw balance but it is reserved as `totalRewards` or `fundingOwed`, the rebate transfer can still drain it because `sendReferrerRebateToStorage()` ignores `balanceAvailable`. The same bug therefore either blocks closes when raw balance is insufficient, or consumes funds that were already reserved for LP rewards or future funding payouts.

H-05 Referrer rebates can block closes**CONTINUED**

— RECOMMENDATION

Do not transfer referrer rebates during close settlement with a raw VaultManager transfer. Accrue the rebate claim and move funds only from fee backed available balance, or use a dedicated fee payout helper that respects `balanceAvailable` and never pulls LP tranche assets for dev, gov, or referrer fee distributions. Close settlement should not revert merely because an immediate rebate transfer cannot be funded.

— RESOLUTION

Status: Acknowledged.

HIGH **H-06**

RESOLVED

LP withdrawals can block liquidations

CATEGORY DoS

LOCATION `src/PairInfos/PairInfos.sol:getPendingAccRolloverFees`

REVIEW Main Review 1

DESCRIPTION

For pairs without a fixed hard cap, `pairMaxOI` falls with junior tranche assets. A public ERC4626 withdrawal can therefore reduce the pair cap below already open OI because withdrawals enforce only the tranche's global reserve ratio, while existing pair OI and rollover fee state remain unchanged. The long and short OI caps do not prevent this because they are checked only when exposure is added and they also shrink after the withdrawal.

`getBlendedUtilizationRatio` divides existing pair OI by the current `pairMaxOI` without capping the result. When utilization and side skew exceed the domain assumed by `getPendingAccRolloverFees`, the dynamic multiplier reverts before any close can reduce OI:

```
```plain text
```

```
uint256 pairUtilization = ((_longOI + _shortOI) * _PRECISION) /
pairsStorage.pairMaxOI(_pairIndex);
```

```
uint256 longMultiplier =
PRECISION * (blendedUtilizationRatio * skewRatio) /
(PRECISION * 1e2 - blendedUtilizationRatio * skewRatio);
``
```

Assume a 20% utilization threshold and a 55% skew threshold:

```
<table header-row="true">
<tr>
<td>State</td>
<td>Pair cap</td>
<td>Long OI</td>
<td>Short OI</td>
<td>Utilization</td>
<td>Long skew</td>
<td>Utilization × skew</td>
</tr>
<tr>
<td>Before LP withdrawal</td>
<td>100`</td>
```

**H-06** LP withdrawals can block liquidations

CONTINUED

```

<td> 48 </td>
<td> 32 </td>
<td> 80% </td>
<td> 60% </td>
<td> 48% </td>
</tr>
<tr>
<td>After LP withdrawal</td>
<td> 40 </td>
<td> 48 </td>
<td> 32 </td>
<td> 200% </td>
<td> 60% </td>
<td> 120% </td>
</tr>
</table>

```

- OI and skew do not change. Only the TVL derived pair cap falls.
- The denominator attempts `100% - 120%` and underflows.

The transition can happen in either of these ways:

- Naturally, ordinary LPs redeem shares while the pair has open OI and the tranche's global withdrawal check still passes.
- Deliberately, an LP times the withdrawal after establishing, or observing, sufficiently skewed OI. `TradeRegistry.unregisterTrade` calculates rollover fees before releasing OI, and liquidation reads the same fee while calculating the liquidation price. The resulting revert blocks:
  - market closes, TP and SL execution;
  - liquidations and margin updates; and
  - the risk manager force close.

The pair remains frozen until another party adds sufficient liquidity or governance changes configuration. An attacker can use this window to keep an adverse leveraged position unliquidatable and expose LPs to bad debt as losses exceed the position's collateral.

**— RECOMMENDATION**

Limit `maxWithdraw` and `maxRedeem`, and enforce the same condition during withdrawal, so post-withdraw group and side caps still cover existing OI and total pair OI remains strictly below `pairMaxOI`. Independently define a bounded rollover multiplier for products at or above `1e12` so close and liquidation paths remain available if this invariant is broken by rounding, configuration changes, or other tranche balance reductions.

**H-06** LP withdrawals can block liquidations**CONTINUED**

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/SentryRepo-Avantis-Fuzz-019f3d9087c57fccbd75070661ae90cf/blob/b2b4334e1b0a6ec8b4d580f11550b7bf3d5c1a94/sentry-test/fuzzing/FoundryBreakDebugging.sol#L1922>

**— RESOLUTION**

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

HIGH **H-07**

RESOLVED

# Direct donations bypass an exhausted deposit cap to forge OI capacity and freeze rollover-dependent settlement

CATEGORY DoS

LOCATION packages/avantis-contracts-team2-1783453751962/lib/openzeppelin-contracts-upgradeable/contracts/token/ERC20/extensions/ERC4626Upgradeable.sol:totalAssets():L102-L105 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:\_deposit():L271-L281 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:currentAdjustedBalanceUSDC():L398-L403 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:pairMaxOI():L482-L488 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695

REVIEW Main Review 1

## — DESCRIPTION

Deposit-cap enforcement exists only in `_deposit`, while inherited `totalAssets` counts raw USDC balance. Once ordinary deposits bring the tranche to `depositCap` minus one raw unit, every positive ERC4626 deposit is forbidden, but direct transfers still inflate `currentAdjustedBalanceUSDC` and soft OI caps. A dominant incumbent LP can donate temporary capacity, fill exposure rejected at genuine TVL, and withdraw the donation through pre-existing shares. OI then exceeds restored `pairMaxOI`; one-sided utilization above 100% underflows `PairInfos`' rollover denominator, reverting rollover-dependent settlement until temporary assets are restored. The passing PoC uses an identical-snapshot control to prove zero valid deposit headroom and rejection of the target fill, then proves donation-based admission, capital recovery, rollover and market-close panic, and selective unfreeze.

Impact: Even with the deposit cap exhausted, a dominant LP/trader can use recoverable 1%-TVL liquidity to create exposure above genuine pair limits and selectively deny rollover-dependent exits and liquidations during adverse prices. Reservations and risk remain live while temporary re-donation lets the attacker choose when settlement resumes. This materially extends the accepted deposit-only vector to cap-exhausted states.

## — RECOMMENDATION

Separate accounted vault assets from raw token balance and base risk capacity only on assets admitted through recognized capped flows. Do not let unsolicited transfers expand group or pair OI limits. Independently bound utilization or use non-underflowing rollover math so over-cap states cannot block exits and liquidations.

## — RESOLUTION

Status: Resolved.

HIGH **H-08**

RESOLVED

## Repeated dust RFQ closes create an unworkaroundable close-settlement freeze

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/Trading/RfqTradingModule.sol:\_validateCloseReq():L326-L338 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:\_executeCloseRfqSlice():L526-L587 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:unregisterTrade():L310-L386 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeValue():L258-L311 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:allocateRewards():L292-L297 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:\_sendUSDCToTrader():L488-L501 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:withdrawAsVaultManager():L144-L147

REVIEW Main Review 1

### — DESCRIPTION

RFQ closes and slices accept dust coin sizes that convert to zero collateral. IntentCallbackModule nevertheless settles the complete live trade, so PairInfos computes the full position's accrued obligations while TradeRegistry takes the zero-collateral partial branch, refreshes the full fee snapshots, and credits the borrow fee to VaultManager.totalRewards without collecting USDC. Repetition can make synthetic rewards exceed manager assets by more than the tranche's 500,000 USDC per-call manager-withdrawal cap. Every subsequent unregisterTrade settlement calls sendUSDCToTrader before reducing the position; the preexisting deficit alone then causes withdrawal to revert regardless of the requested close size.

Impact: The passing revised PoC uses realistic two-second Base block timing and repeats three 25-day accrual/dust-close cycles on a solvent 1,000,000 USDC, 10x position. Each cycle forgives about 685,000 USDC while leaving its collateral, OI, and reserve unchanged. The resulting reward deficit exceeds 500,000 USDC, after which independently requested 1, 10,000, and 50,000 USDC partial closes of an honest position all revert before changing it. Splitting exits cannot bypass the state; ordinary close settlements remain unavailable until sufficient assets recapitalize the manager or accounting is repaired, in addition to the fee evasion and LP loss already established by the prior finding.

### — RECOMMENDATION

Reject RFQ close targets and slices unless their converted collateral/OI reduction is positive and above a meaningful minimum. Settle borrow and funding obligations pro rata to exposure actually removed, preserve debt attributable to a surviving position, and never refresh full fee snapshots on a zero-sized reduction. Credit reward buckets only with collected backing, enforce a liability-versus-assets solvency invariant, and avoid requiring a single capped tranche withdrawal to cover the entire aggregate deficit during every close.

### — RESOLUTION

HIGH **H-09**

RESOLVED

## Fixed-coin increases bypass leverage limits by counting impaired collateral at nominal value

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:_increasePositionSizeWithCoinExposure():L421-L468 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/MarketCallbackModule.sol:increasePositionSizeCallbackWithCoinExposure():L287-L309 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncreaseWithCoinExposure():L258-L292 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:_computeNewLeverage():L444-L453
REVIEW	Main Review 1

### — DESCRIPTION

The fixed-coin increase request does not bound the leverage of the added leg. TradeRegistry derives that leverage from attacker-selected coin exposure and added collateral, while callback validation checks only the blended nominal leverage. The blend divides total OI by all stored collateral without accounting for the existing position's unrealized loss. Thus collateral in a nearly liquidatable seed is counted at full nominal value, allowing one dollar of added collateral to introduce vastly more OI than the pair's leverage limit would permit for that leg or the position's live equity. The passing public-flow fork PoC adds about 148,000 USDC OI for 1 USDC collateral to a distressed 2x seed: the marginal leg is 148,000x and effective leverage against live equity is far above the 150x pair maximum, yet the stored nominal leverage passes just below 150x.

Impact: An attacker can hedge a low-leverage seed while moving it close to liquidation, then remove the hedge and use a dust-collateral fixed-coin increase to obtain an LP-funded high-notional option with far less liquidation buffer than configured leverage limits permit. In the PoC, a 1% rebound pays over 1,500 USDC while an otherwise equivalent one-dollar increase capped to 150x pays under 300 USDC. Added OI and reserved tranche liability can scale to wallet/pair exposure limits; favorable moves consume protocol liquidity, while price gaps can leave the newly added exposure undercollateralized.

### — RECOMMENDATION

At execution, independently derive and enforce pair/request leverage bounds for the added fixed-coin leg. Also calculate post-increase effective leverage and liquidation health using live equity, including unrealized PnL and accrued fees, and reject increases that exceed pair risk limits or leave insufficient liquidation buffer.

H-09

Fixed-coin increases bypass leverage limits by counting impaired collateral at nominal value

CONTINUED

— RESOLUTION

Status: Resolved.

HIGH **H-10**

RESOLVED

# A front-run TWAP slice expands a prepared TP into a retroactive LP-funded payout

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/Trading/LimitTradingModule.sol:\_executeLimitOrder():L332-L379 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:openTwapCallback():L107-L178 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncrease():L169-L235 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L166-L205

REVIEW Main Review 1

## — DESCRIPTION

A generic TP transaction binds neither the observed trade incarnation nor its collateral/OI. While a legitimate crossed-TP report is propagating for a small TWAP-created position, another ordinary operator can settle a scheduled TWAP slice that increases the same tuple. The unchanged, still-fresh TP report then closes and values the enlarged position, granting the new exposure PnL at a price published before that exposure existed. The passing PoC expands a 100,000-USDC shell by 900,000 USDC at \$2,000, then uses an authentic earlier \$2,036 report within the configured backup deviation to settle 18% leveraged profit on the full position.

Impact: Independently valid keeper actions can be reordered so newly added exposure receives retroactive PnL from LP assets. The PoC yields the trader over 150,000 USDC and shows a matching protocol-asset reduction without TP mutation, guaranteed-price mode, forged oracle data, or stale-index reuse. Loss scales with the permitted increase, leverage, and oracle freshness/deviation window.

## — RECOMMENDATION

Version each trade mutation and bind TP/SL executions to the observed timestamp and notional/version, rejecting settlement if the target was increased, partially closed, or replaced. Alternatively serialize intent increases against active close executions and require oracle reports used for a close to postdate the bound position state.

## — RESOLUTION

Status: Resolved.

HIGH **H-11**

ACKNOWLEDGED

# One-tick guaranteed-stop PnL straddles create an unpriced LP-funded volatility option

CATEGORY	Informational
LOCATION	<code>packages/avantis-contracts-team2-1783453751962/src/Trading/MarketTradingModule.sol:_openTrade():L187-L218</code> <code>packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerTrade():L55-L85</code> <code>packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:correctSL():L293-L298</code> <code>packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:_getTradeRollOverFee():L720-L740</code> <code>packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeFundingFee():L932-L950</code> <code>packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L178-L205</code>
REVIEW	Main Review 1

## DESCRIPTION

If guaranteed SL is enabled on a PnL-order pair, opening validation requires a stop only to be directionally beyond entry and correctSL widens only overly distant stops. Equal long and short PnL legs can therefore retain stops one price unit from entry. PnL opens charge no fixed fee and their position type exempts them from rollover and funding. When the losing stop is genuinely crossed, settlement computes its PnL at the stored one-tick stop instead of the adverse oracle price, while the opposite TP leg realizes leveraged profit. The revised PoC passes and demonstrates zero carry, over 60 USDC net trader profit, and over 60 USDC tranche loss on 200 USDC collateral.

Impact: Once the supported guaranteed-SL flag is combined with a PnL market, users obtain a direction-neutral, near-zero-downside volatility option without premium or carry. They can occupy both OI sides and retain the winning leveraged payout without an offsetting losing-leg PnL. Repeated use can extract material tranche principal up to OI and liquidity limits. The unsafe combination is optional and disabled in fixture defaults, so High rather than Critical is appropriate.

## RECOMMENDATION

Enforce a meaningful minimum guaranteed-SL distance from final execution entry and prohibit guaranteed SL on PnL-order pairs unless a gap-risk premium, carry, and OI charge fully price the option. Alternatively, settle adverse gaps at a bounded oracle-derived price rather than an arbitrarily close user-selected stop.

## RESOLUTION

Status: Acknowledged.

HIGH **H-12**

ACKNOWLEDGED

# Cross-pair forex hedges retroactively erase target rollover debt

CATEGORY	Logical Error
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getBlendedSkew():L624-L640 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:_updateOpenInterestUSDC():L723-L762
REVIEW	Main Review 1

## — DESCRIPTION

Forex rollover blends pair-local skew with global directional `usdOI`, but the pending accumulator applies the current blend to every block since the target pair's last checkpoint. Opening an auxiliary forex position checkpoints only that pair before mutating the global `usdOI` used by all forex targets. An attacker can therefore age a costly target, briefly add opposite global-USD exposure on another listing, and checkpoint the target after the endpoint skew crosses into a cheaper fee regime. Closing the auxiliary leg does not restore the erased accumulator. The enhanced PoC uses nonzero open/close fees and identical snapshots: authentic carry makes the target liquidatable through the real LIQ path at the unchanged authenticated price, whereas the cross-pair hedge changes target skew without changing its checkpoint and makes the same liquidation revert.

Impact: The attack defeats a valid solvency liquidation, permanently erases aged carry, and leaves the trader with more aggregate USDC after fully unwinding both positions and paying all auxiliary fees than the control liquidation. The PoC also proves VaultManager retains correspondingly less USDC. One global-USD hedge can affect multiple forex targets, including hard-capped pairs whose `pairMaxOI` does not respond to tranche liquidity.

## — RECOMMENDATION

Maintain time-integrated per-pair and global-USD skew accumulators and charge each elapsed interval using the skew then in effect. At minimum, checkpoint every affected forex pair before any trade mutates global `usdOI`; checkpointing only the auxiliary pair is insufficient.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/f64c45a26e3a6dd39ad3211d2b57d0bab95e0c10>

## — RESOLUTION

Status: Acknowledged.

HIGH **H-13**

RESOLVED

# Full-sized partial TP bypasses PnL performance fees

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_executeCallback():L396-L436 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executePartialTpSlCallback():L327-L404 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:unregisterTrade():L310-L339 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:getPnlBasedFee():L206-L217
REVIEW	Main Review 1

## DESCRIPTION

A correctly tagged live PnL position is persistently misclassified only in the signed partial-close lifecycle. PriceAggregator dispatches every LIMIT\_PARTIAL\_CLOSE with isPnl=false rather than reading the position's canonical PairStorage posType. LimitCallbackModule forwards false into TradeRegistry, selecting fixed close-fee treatment instead of the PnL performance fee. A request covering the entire coin exposure fully deletes the position, yielding a complete fee-bypassing exit. This is independent of the accepted legacy-open issue because it requires no upgrade, legacy pending order, or initial posType corruption, and is not repaired by open-callback migration handling. It is also independent of partial-close debt findings because fee proration/snapshot fixes do not change ctx.isPnl.

Impact: Every profitable PnL position can use this exit to retain performance fees owed to protocol/LP liquidity. At the fixture pair's 500% maximum gain, the passing fork PoC charges under 5% of the correct fee and pays over 90% of the avoided fee—approximately the original collateral—to the trader per position. Unlike the legacy-open issue, exploitability is permanent rather than limited to one transition window.

## RECOMMENDATION

At partial-close settlement, derive fee mode from the canonical live PairStorage.getPosType value, or dispatch a dedicated LIMIT\_PARTIAL\_CLOSE\_PNL order. Apply stored position type consistently to fees and referral treatment.

## RESOLUTION

Status: Resolved.

HIGH **H-14**

ACKNOWLEDGED

# Insolvent one-sided payers manufacture funding liabilities that drain LP principal

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeValue():L258-L311 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:_recordFundingAudit():L352-L354 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:fundingOwed():L361-L365 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:_sendUSDCToTrader():L488-L501
REVIEW	Main Review 1

## — DESCRIPTION

One-sided OI creates funding charges without any receiver. More severely, close settlement records the full nominal funding fee even when it exceeds all collateral the payer can surrender. `fundingOwed()` treats that uncollected lifetime amount as cash owed, so `VaultManager` immediately pulls the deficit from the tranche even during the insolvent payer's own zero-payout close. The passing PoC uses ordinary 20%-annualized funding and public market functions: a 10,000-USDC 150x payer accrues about 500,600 USDC nominal funding, closes with zero payout, and forces roughly 490,600 USDC of LP principal into `VaultManager`. The pair then has zero OI but retains the complete ownerless liability. Repeating on an unrelated pair drains another roughly 490,600 USDC and compounds the residual above 1 million USDC.

Impact: An insolvent payer can manufacture much more funding liability than the protocol actually collected. The new trigger amplifies the accepted one-sided issue from stranded collected cash to an immediate tranche draw approaching 500,000 USDC per epoch. The PoC removes over 980,000 USDC from tranche assets using 20,000 USDC total payer collateral, a roughly 49x grieving amplification. Repeated pairs or epochs can drain LP NAV, permanently sequester manager liquidity, and reach the tranche withdrawal ceiling that blocks settlements.

## — RECOMMENDATION

Do not accrue funding when no counter-side OI can own the credit. Additionally, record `totalFundingPaid` only up to funding actually recovered from the payer after collateral/PnL floors; track current receiver liabilities per pair and epoch rather than a lifetime global audit net. Handle any residual funding bad debt explicitly instead of automatically pulling LP principal into an ownerless reserve.

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-01**

RESOLVED

# Rollover fee divides by zero at max utilization

CATEGORY Error

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/PairInfos/PairInfos.sol#L665>

REVIEW Main Review 1

## DESCRIPTION

In `getPendingAccRolloverFees`, the "above threshold" borrow multiplier for each side is computed as:

```
``plain text
```

```
: _PRECISION*(blendedUtilizationRatio*skewRatio)/(_PRECISION*1e2 -
blendedUtilizationRatio*skewRatio);
```

`blendedUtilizationRatio` is capped at `1e10` (open interest equal to `pairMaxOI`, i.e. 100% utilization) and `skewRatio` is capped at `100` (one side holds 100% of the pair's open interest). At that corner the denominator `_PRECISION*1e2 - blendedUtilizationRatio*skewRatio` equals `1e12 - 1e12 = 0`, and the division reverts.

A pair driven to exactly 100% utilization with fully one-sided open interest can no longer be opened, closed, liquidated, margin-updated, or rolled, since every settlement path reverts on the division.

The liquidation impact is the most severe. Liquidation reads this value twice, at initiation and at settlement. Both revert, so no position on the affected pair can be liquidated. An attacker can weaponize this: open a single large one-sided position that fills the pair to `pairMaxOI` at skew 100, and every position on that pair (their own and others') becomes un-liquidatable and un-closable, leaving the LPs stuck backing insolvent positions while the attacker chooses when to unwind.

Reaching `skewRatio == 100` together with 100% utilization additionally requires `pairMaxLongOI` or `pairMaxShortOI` to be configured up to `pairMaxOI` (`maxLongOiP / maxShortOiP` at full), so exploitability is config-gated.

## RECOMMENDATION

Clamp the utilization or skew inputs so the denominator cannot reach zero, or add a guard that caps the multiplier (for example, fall back to `pairBorrowFeesMultiplier` or a configured maximum) when `blendedUtilizationRatio*skewRatio` approaches `_PRECISION*1e2`. Verify pair OI caps so a single side cannot fill 100% of `pairMaxOI`.

**M-01** Rollover fee divides by zero at max utilization

CONTINUED

**— RESOLUTION**

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-02**

RESOLVED

# TWAP RFQ slices override user size

CATEGORY Validation  
LOCATION TwapTradingModule.sol  
REVIEW Main Review 1

## DESCRIPTION

Signed and direct TWAP/RFQ open orders store the trader's requested `defaultLeverage`, `maxLeverage`, and, for coin-sized orders, `coinSize`. The operator-controlled open slice later supplies its own `leverage` or `coinExposure`, and the callback uses those slice values when registering the trade. For normal collateral-sized slices, the first filled slice does not check `slice.leverage` against the stored user bounds or the pair leverage bounds. For coin-sized orders, the callback checks only the current slice leverage and never caps cumulative `coinExposure` against the stored `coinSize`.

As a result, a signed TWAP/RFQ can settle into higher leverage or larger total exposure than the trader authorized. This increases OI, fees, and liquidation risk while still using the trader's escrowed collateral.

## RECOMMENDATION

Validate every open slice against the stored user intent. Enforce pair min/max leverage and `state.defaultLeverage` or `state.maxLeverage` for collateral-sized slices, and track cumulative coin exposure so coin-sized orders cannot exceed `state.coinSize`.

## RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-03**

ACKNOWLEDGED

# Rollover config changes apply retroactively

CATEGORY Logical Error  
LOCATION src/PairInfos/PairInfos.sol  
REVIEW Main Review 1

## DESCRIPTION

`getPendingAccRolloverFees()` computes pending borrow-fee accrual from the last stored accumulator block using current `PairStorage` config. The direct rollover-rate setters snapshot first, but `PairStorage.updateAdditionalPairParams()` changes the multiplier and threshold inputs without calling `_storeAccRolloverFees()`:

```
``plain text
params.utilizationThreshold = _params.utilizationThreshold;
params.borrowFeesMultiplier = _params.borrowFeesMultiplier;
params.skewThreshold = _params.skewThreshold;
`
```

If governance doubles the borrow multiplier at block `B1` while the last stored rollover block is `B0`, the next close or accumulator update at `B2` charges `B0..B2` at the new multiplier instead of only `B1..B2`. This can overcharge traders or under-allocate borrow fees to the vault depending on whether the config change increases or decreases the effective rate.

The same issue applies to config that feeds utilization, including `pairMaxOI()` inputs such as group max OI, pair group allocation, asset hard caps, OI multiplier, TVL cap, and vault adjusted balance.

## RECOMMENDATION

Materialize rollover fees for affected pairs before changing any config used by rollover accrual.

## RESOLUTION

Status: Acknowledged.

**Guardian:** Rollover borrowing fees accrue from the last stored checkpoint using the capacity and utilization read at the next update. Although `updateAdditionalPairParams` and `updatePair` now materialize pending rollover fees first, `setOIMultiplier`, `setAssetOIParams`, and `updateGroup` still change inputs used by `pairMaxOI` or `groupMaxOI` without

**M-03** Rollover config changes apply retroactively

CONTINUED

checkpointing.

If one of these values changes between two accumulator updates, the next update applies the new capacity to every block since the previous checkpoint. Governance can therefore retroactively increase or decrease borrowing fees that traders already accrued.

**\*\*Recommendation\*\***

Materialize rollover accumulators for every affected pair before changing the OI multiplier, per-asset OI parameters, group capacity, or any other input used by `pairMaxOI` or `groupMaxOI`. For global and group-level setters, iterate or otherwise checkpoint all affected pairs before committing the new value.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-04**

ACKNOWLEDGED

# Stale close callbacks hit new positions

CATEGORY	Logical Error
LOCATION	MarketTradingModule.sol
REVIEW	Main Review 1

## — DESCRIPTION

A pending market close is bound only to `(trader, pairIndex, index)` and the requested collateral amount. The stored order does not preserve the original position's timestamp.

`beingMarketClosed` prevents another market-close request, but it does not prevent TP, SL, or liquidation settlement from closing the original position. A subsequent open may reuse the freed index because the open-position cap excludes pending market closes. When the delayed callback executes,

`closeTradeMarketCallback()` reloads whichever live trade currently occupies that index, clamps the old close amount to its collateral, and settles that replacement trade without checking position identity.

Therefore, an ordinary sequence of pending market close, TP/SL/liquidation close, replacement open, and delayed callback within the 15-block order window can partially or fully close an unrelated new position.

Example:

1. Position A is live at index 0
2. A market-close request is created for A, but remains pending
3. A is closed through TP/SL/liquidation
4. Position B's market-open request is created
5. B's open callback executes, registering B at index 0
6. A's old market-close callback executes and closes B

Pending position increases have the same identity bug. Both increase entypoints store a synthetic trade with `timestamp = 0`, then their callbacks reload the current trade by `(trader, pairIndex, index)` without comparing timestamps. If the original position closes and its index is reused before fulfillment, the stale increase, including its escrowed collateral, is merged into the replacement position.

## — RECOMMENDATION

Callbacks must target a position generation, not merely a reusable storage slot which is prone to corruption.

**M-04** Stale close callbacks hit new positions

CONTINUED

**— RESOLUTION**

Status: Acknowledged.

MEDIUM **M-05**

ACKNOWLEDGED

# Pending rewards can be sniped

CATEGORY Gaming  
LOCATION AvantisTranche.sol, VaultManager.sol  
REVIEW Main Review 1

## — DESCRIPTION

`AvantisTranche.totalAssets()` is the raw USDC balance held by the tranche. Trading losses and fees are first held in `VaultManager.pnlRewards` and `totalRewards`, so they do not increase the ERC4626 share price until a keeper later transfers them into the tranche.

A user can deposit after rewards have accrued but before distribution, receive shares at the old price, then redeem after the keeper transfers the rewards. The new depositor captures rewards earned entirely by prior LPs. In the reproduced case, the attacker deposited an amount equal to existing tranche assets immediately before an equally sized PnL-reward distribution, then redeemed about `1.5x` their deposit, capturing half of the pre-deposit rewards.

The attack is particularly low risk when the positions that generated the rewards are already closed and `totalReserved` is zero. The legacy lock-time variables in `VaultManager` are not enforced by the ERC4626 deposit or withdrawal paths.

## — RECOMMENDATION

Include accrued `totalRewards` and `pnlRewards` in tranche share valuation, or materialize all pending rewards before every deposit, mint, withdrawal, and redemption. If rewards remain asynchronously distributed, prevent newly minted shares from participating in rewards accrued before their deposit.

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-06**

ACKNOWLEDGED

# Market closes ignore wanted price

CATEGORY Validation  
LOCATION MarketTradingModule.sol  
REVIEW Main Review 1

## DESCRIPTION

Both market-close entrypoints accept `_wantedPrice`, but store every price and slippage field in the pending order as zero. `_wantedPrice` is used only in the initiation event, and the callbacks settle at the returned oracle price plus spread without enforcing the user's bound.

```
``plain text
ITradingStorage.PendingMarketOrder(
ITradingStorage.Trade(_trader, _pairIndex, _index, _amount, 0, 0, false, 0, 0, 0, 0),
0,
0,
0
)
`
```

A long trader can submit a close with a minimum acceptable price of `\$2,990`, yet the operator can fulfill it at `\$2,900` and fully close the position. Short closes similarly ignore the user's maximum price. Normal oracle movement or delayed execution can therefore realize losses beyond the explicit user limit.

## RECOMMENDATION

Store `_wantedPrice` in each pending close order and reject callback execution when the spread-adjusted close price violates it. Define the bound as a minimum for longs and a maximum for shorts, including the coin-exposure path.

Kartik:

`_wantedPrice` in closes is just a pass through for our indexer. We don't want to gate closes because of slippage. Close spread is determined by backend and we wish to enforce this with some flexibility on BE only.

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-07**

RESOLVED

# Coin OI rounding blocks full closes

CATEGORY Rounding

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/PairStorage/PairStorageExtension.sol#L115-L116>

REVIEW Main Review 1

## DESCRIPTION

Position increases recompute a blended `openPrice` with integer division, while aggregate coin OI is updated by adding each execution's independently rounded coin amount. Recomputing the final position's coin size from its total notional and rounded blended price can therefore exceed the stored aggregate coin OI by one unit.

Before a standard close updates storage, `pairCloseFeeP()` subtracts that recomputed position size from aggregate coin OI without a dust clamp:

```
``plain text
uint256 coinOiAfterClose = _buy
? initialCoinOiUSDCLong - _positionSizeCoinOi
: initialCoinOiUSDCShort - _positionSizeCoinOi;
`
```

For example, opening `\$100` notional at `\$2,900` and increasing by `\$100` at `\$2,902` produced aggregate coin OI of `689417523`, but the blended position reported `689417524`. A standard full market close then reverted with arithmetic underflow before reaching the later storage dust clamp. Full TP and SL settlement use the same fee path, so protective exits can also fail. Liquidation or a manual coin-exposure close using the exact aggregate amount may remain possible, but the trader cannot rely on normal full-close paths and can be liquidated while attempting to exit.

## RECOMMENDATION

Clamp the coin OI used by `pairCloseFeeP()` to the available side aggregate, consistent with the storage close path. Prefer storing and updating each position's exact coin exposure instead of reconstructing it from a rounded blended price.

## RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-08**

RESOLVED

# Liquidations use stale funding price

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/LimitCallbackModule.sol#L209">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/LimitCallbackModule.sol#L209</a>
REVIEW	Main Review 1

## DESCRIPTION

`executeLimitCloseOrderCallback()` validates liquidation using the seven-argument `getTradeLiquidationPrice()` overload. That overload calculates pending funding at the pair's last stored funding price instead of the current callback price:

```
``plain text
uint256 lastPrice = pairFundingFees[pairIndex].lastUpdatePrice;
return _getTradeLiquidationPrice(
trader, pairIndex, index, openPrice, long, collateral, leverage, lastPrice
);
``
```

Pending funding is price-dependent, so the liquidation threshold used by the callback diverges whenever the market price has moved since the funding keeper's last update. Settlement later materializes funding using the fresh callback price, meaning liquidation validation and the resulting position value use inconsistent fee amounts.

Depending on skew and direction, the inverse discrepancy can also permit a liquidation that current funding does not justify. This can delay liquidations into bad debt or prematurely close a trader's position.

## RECOMMENDATION

Pass the callback's validated current price into the eight-argument `getTradeLiquidationPrice()` overload. Use that same price consistently for liquidation validation and fee materialization.

## RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-09**

ACKNOWLEDGED

# Partial closes can strand dust positions

CATEGORY	Logical Error
LOCATION	MarketTradingModule.sol, TradeRegistry.sol
REVIEW	Main Review 1

## — DESCRIPTION

Market partial closes only check the collateral slice being closed against the pair minimum. They do not reject or round up a close that leaves the remaining live position below `pairMinLevPosUSDC`. After settlement, `TradeRegistry.unregisterTrade()` treats any close smaller than the full collateral as a partial close and calls `registerPartialTrade()`. That storage path subtracts the closed collateral and OI, but it does not validate the remaining position size.

Illustration:

```vb.net

Collateral: 100 USDC

Leverage: 10x

Notional: 1,000 USDC

Pair min notional: 100 USDC

Close amount: 91 USDC

Closed notional: 910 USDC, passes

Remaining: 9 USDC

Remaining notional: 90 USDC, below min

``|
The residual position remains open, but a later normal market close of the full residual reverts because $9 * 10x$ is below the same minimum. The coin exposure close path has an equivalent minimum close check before its callback can round tiny remainders into a full close. Full TP, SL, or liquidation can still clear the position if their separate trigger conditions are met, but dust positions can be uneconomic to monitor and execute. This can leave stale below minimum exposure in storage and can turn into uneconomic bad debt cleanup if the dust later moves underwater.

— RECOMMENDATION

When a partial close would leave the remaining leveraged notional below `pairMinLevPosUSDC`, either reject the close or round it to a full close. Apply the same remainder rule to collateral based closes and coin exposure closes.

M-09 Partial closes can strand dust positions

CONTINUED

— RESOLUTION

Status: Acknowledged.

Guardian: Removing the minimum close-size checks lets an already-created residual be closed later, but neither market-close path checks whether the live remaining leveraged notional falls below `pairMinLevPosUSDC`. The collateral callback sends every non-full close to `registerPartialTrade`, which persists the remainder without validation. The coin-exposure callback rounds up only when the remaining collateral is below `COLLATERAL_DIFF_THRESHOLD` (`0.01 USDC`), which is unrelated to the configured pair minimum. A trader can therefore still create below-minimum live positions that are uneconomic to monitor or liquidate and can become dust bad debt.

****Recommendation****

Before settling either collateral-based or coin-exposure partial closes, calculate the live remaining leveraged notional. If it would fall below `pairMinLevPosUSDC`, reject the close or round it to a full close.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-10**

ACKNOWLEDGED

Stale prices allow unsafe margin withdrawals

CATEGORY Logical Error
 LOCATION LimitCallbackModule.sol
 REVIEW Main Review 1

DESCRIPTION

`updateMargin` lets the trader provide the oracle update used to decide whether collateral can be withdrawn. Pyth Core accepts any update no older than 30 seconds, while the backup feed permits deviations up to the pair's configured threshold. At high leverage, the position can cross liquidation after a price move far smaller than either window protects against.

```
``plain text
```

```
pyth.updatePriceFeeds{value: pyth.getUpdateFee(priceUpdateData)}(priceUpdateData);
priceData = pyth.getPriceNoOlderThan(r.job, 30);
```

```
`|
```

The callback calculates PnL and the withdrawal threshold entirely from that accepted price. A trader can therefore submit a still-valid favorable update after the market has moved against the position, withdraw collateral, and leave the position insolvent at the current price. For example, the PoC opens a 100 USDC, 100x long at 50,000. Twenty seconds later, the trader submits the old 50,000 update and withdraws 33 USDC. At the current price of 49,575, a 0.85% move, the position has lost 85 USDC but has only 67 USDC collateral left. It is immediately liquidatable and the vault absorbs the shortfall. The configured 2% backup-deviation bound does not reject the stale price.

RECOMMENDATION

Use a much tighter freshness bound for collateral withdrawals and evaluate the withdrawal at the conservative price across independent fresh sources. Require the post-withdrawal position to remain above liquidation with a buffer sized for oracle age, execution latency, spreads, and fees. Consider the two-step pattern used by other perps: the trader first requests the margin withdrawal, then a keeper executes it only with oracle timestamps valid for that request window.

M-10 Stale prices allow unsafe margin withdrawals

CONTINUED

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-11**

ACKNOWLEDGED

Dev gov fees can pull tranche funds

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/TradeRegistry.sol#L349 |
| REVIEW | Main Review 1 |

DESCRIPTION

Close settlement charges the close fee through `getTradeValue()`, then books the treasury share as dev and gov fees. The actual transfer of that treasury share uses

`VaultManager.sendUSDCToTrader(address(storageT), ...)`, the same helper used for trader payouts.

```plain text

```
storageT.incrementClosingFees(
 feeValues.feeAfterRebate - feeValues.referrerRebate - vaultAllocation - govFees,
 govFees
);
```

```
storageT.vaultManager().sendUSDCToTrader(
 address(storageT),
 feeValues.feeAfterRebate - feeValues.referrerRebate - vaultAllocation
);
```

`` `sendUSDCToTrader()` is not a plain fee transfer. It computes available liquid balance after reserved rewards and funding owed, and withdraws any shortfall from the junior tranche before sending funds.

`plain text

```
int256 balanceAvailable = int256(storageT.usdc().balanceOf(address(this))) -
 int256(totalRewards) - fundingOwed;
if (int256(_amount) > balanceAvailable) {
 uint256 difference = uint256(int256(_amount) - int256(balanceAvailable));
 junior.withdrawAsVaultManager(difference);
}
```

This is appropriate for trader payouts that LPs back, but not for dev and gov fee distributions. If VaultManager liquid balance is low, or if the raw balance is reserved as totalRewards or funding owed, a close can pull junior tranche assets into TradingStorage and book them as dev or gov fees. The protocol then lets the buyback keeper withdraw those amounts from TradingStorage, so LP principal can be used to fund treasury revenue rather

**M-11** Dev gov fees can pull tranche funds

CONTINUED

than only realized fee cash.

Illustration:

```
```vb.net
```

VaultManager raw balance: 50 USDC

Reserved totalRewards: 50 USDC

Funding owed: 0 USDC

Available balance: 0 USDC

Dev and gov close fee share: 5 USDC

```
sendUSDCToTrader(address(storageT), 5)
```

pulls 5 USDC from the junior tranche

and records 5 USDC as dev and gov fees.

```
```
```

**— RECOMMENDATION**

Do not use `sendUSDCToTrader()` for dev and gov fee transfers. Move treasury fee shares with a dedicated fee payout helper that can only use fee backed available balance and never pulls from tranche assets. If fee cash is not immediately available, accrue the dev and gov claim and defer the transfer until it is backed by available VaultManager balance.

**— RESOLUTION**

Status: Acknowledged.

MEDIUM **M-12**

ACKNOWLEDGED

# Open TWAP RFQ slices merge into recycled trade

|          |                                                                                                                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Logical Error                                                                                                                                                                                                                                                                                   |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/IntentCallbackModule.sol#L695">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/IntentCallbackModule.sol#L695</a> |
| REVIEW   | Main Review 1                                                                                                                                                                                                                                                                                   |

## — DESCRIPTION

Open TWAP/RFQ slices bind to the target position only by `(trader, pairIndex, index, direction)`, with no position-instance identity check. After the first open slice records `state.index`, every later slice increases whatever trade currently occupies that slot, as long as the direction matches:

```
``plain text
function _shouldOpenFreshSlice(address trader, uint256 pairIndex, uint256 index, bool buy)
private view returns (bool) {
ITradingStorage.Trade memory tracked = storageT.openTrades(trader, pairIndex, index);
return tracked.leverage == 0 || tracked.buy != buy;
}
`[
```

The close flow already guards against this by binding to the exact position via open timestamp (`existingTrade.timestamp != state.positionTimestamp` in `closeTwapCallback / closeRfqCallback`). `TwapState / RfqState` even carry a `positionTimestamp` field, but for opens it is set to `0` and never checked.

Trigger: while an open TWAP/RFQ is in progress, the trader closes the tracked position and opens a new same-direction position on the same pair. `firstEmptyTradeIndex` can reassign the freed slot to the new position, so the next operator slice calls `registerPositionIncrease` against that unrelated position. This blends its open price, recomputes leverage, resets its borrow/funding fee snapshots, and normalizes its TP/SL using the TWAP/RFQ escrow. It is the same trader and direction, so this is not fund theft, but it silently corrupts a position the trader never intended to modify, worsening its entry/leverage and raising liquidation risk.

## — RECOMMENDATION

Give open TWAP/RFQ the same identity binding as closes. When `state.index` is first assigned on the initial open slice, record the created position's `timestamp` into the intent

**M-12** Open TWAP RFQ slices merge into recycled trade

CONTINUED

state, and in `_shouldOpenFreshSlice` (or before the increase branch) treat a timestamp mismatch as "open fresh" rather than "increase," so a slice never merges into a recycled slot.

**— RESOLUTION**

Status: Acknowledged.

MEDIUM **M-13**

RESOLVED

# Legacy delegates retain TWAP cancel access

|          |                                   |
|----------|-----------------------------------|
| CATEGORY | DoS                               |
| LOCATION | DelegateModule.sol:removeDelegate |
| REVIEW   | Main Review 1                     |

## DESCRIPTION

The v1.5 `Trading` contract stored each trader's delegate in `delegations`. The upgrade preserves that slot as `__deprecated_delegations`, and `TwapTradingModule.isDelegate` still treats the stored address as authorized. However, the current `removeDelegate` function only clears `multipleDelegations`, so a trader cannot revoke their preserved v1.5 delegate:

```
```plain text
// TwapTradingModule.isDelegate
user == delegate ||
__deprecated_delegations[user] == delegate ||
multipleDelegations[user][delegate].isEnabled

// DelegateModule.removeDelegate
if (!multipleDelegations[_msgSender()][delegate].isEnabled)
  revert Errors.DelegateNotSet();
```
```

After the upgrade, a former delegate can therefore cancel any future open or close TWAP belonging to the trader indefinitely. The refund from an open TWAP still goes to the trader, so the delegate cannot steal it, but they can repeatedly disrupt execution or cancel scheduled closes while the trader's position remains exposed. This is distinct from L-07 because the trader cannot revoke the stale authority.

## RECOMMENDATION

Stop accepting `__deprecated_delegations` in `isDelegate`, or add a migration and revocation path that clears legacy delegate entries before enabling TWAP execution after the upgrade.

## RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-14**

RESOLVED

# Unsafe negative Pyth price cast

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/PriceAggregator.sol#L450>

REVIEW Main Review 1

## DESCRIPTION

`_formatPrice` converts the oracle price with `uint64(priceData.price)` where `priceData.price` is an `int64`. Solidity does not revert when casting a negative `int64` to `uint64`; it reinterprets the two's complement bits, so a negative price such as `-1` becomes `2**64 - 1`. The scaled result is therefore an astronomically large positive number rather than a rejected value.

This defeats the only downstream guard. `_validatedPrice` checks `price > 0`, which a wrapped negative passes, and the confidence bound `(conf * _PRECISION * 100) / price` collapses toward zero against the huge denominator, so it also passes. A negative Pyth or Avantis signed price is then consumed as a valid, enormous price and flows straight into open, close, and PnL settlement, mispricing trades and position health.

```
``plain text
```

```
// _formatPrice: negative int64 wraps to a huge uint64, no sign check
price = (uint64(priceData.price) * _PRECISION) / 10 ** uint32(-priceData.expo);
```

The Pyth Core `_formatPrice` overload and the Avantis signed overload both share this cast, and the Avantis path additionally skips the confidence bound entirely. Only the Lazer path guards this case, rejecting `rawPrice <= 0` in `_parsePythLazerPrice`.

This is not just theoretical for Avantis' supported markets: Avantis docs list WTI among supported commodities, and WTI has a demonstrated nonzero chance of printing below zero (April 2020). A negative WTI update routed through the vulnerable Pyth Core or Avantis signed path would therefore settle as an enormous positive price. Long positions can be credited up to the pair max gain, shorts can be pushed to maximum loss or liquidation, and the pair's settlement integrity depends on backup oracle rejection rather than the primary price parser.

A related assumption in the same function: for `expo` more negative than the `_PRECISION` scale (`1e10`), the division truncates low order digits and can round small prices toward zero, and a large positive `expo` risks overflow. Pyth feeds currently use small exponents, so this is latent, but it is unvalidated.

**M-14** Unsafe negative Pyth price cast

CONTINUED

**RECOMMENDATION**

Reject non positive prices before casting, matching the Lazer path: require `priceData.price > 0` in both `_formatPrice` overloads. Optionally bound the accepted `expo` range per onboarded feed so scaling cannot truncate to zero or overflow.

**RESOLUTION**

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-15**

ACKNOWLEDGED

# Most pairs lack effective oracle backups

CATEGORY Oracle  
LOCATION PriceAggregator.sol  
REVIEW Main Review 1

## DESCRIPTION

The Chainlink backup is optional, and a configured backup is silently ignored when its latest round is older than `chainlinkValidityPeriod`:

```
``plain text
if (backupFeed.maxDeviationP > 0 && backupFeed.feedId != address(0)) {
 (, int256 rawBkPrice, , uint updatedAt,) = chainlinkFeed.latestRoundData();
 if (updatedAt + chainlinkValidityPeriod >= block.timestamp) {
 // Revert when Pyth and Chainlink deviate too far.
 }
}
```

There is no `else` branch. A missing or stale backup therefore fails open and the primary Pyth or Lazer price continues to settlement without an independent price bound.

At Base block `48,864,654`, the deployed v1.5 PairStorage had 114 pair slots, of which 110 were listed. Only pair indexes 0, 1, 2, and 7, corresponding to ETH, WBTC, SOL, and OP, had nonzero Chainlink backup addresses. This leaves 106 of 110 listed pairs without any backup.

Even configured backups are frequently ineffective because the deployment uses one global 120 second validity period. At the same block, the ETH, WBTC, SOL, and OP Chainlink rounds were respectively 380, 208, 5,642, and 46 seconds old. Only OP was fresh enough to participate in validation at that instant. This is not simply caused by unavailable data. For example, AAVE, LINK, and DOGE are listed without configured backups even though Chainlink publishes [AAVE/USD] (<https://data.chain.link/feeds/base/base/aave-usd>), [LINK/USD] (<https://data.chain.link/feeds/base/base/link-usd>), and [DOGE/USD] (<https://data.chain.link/feeds/base/base/doge-usd>) feeds on Base. The deployment constants nevertheless set these backup addresses to zero.

Consequently, a faulty, malformed, or outlier primary price can settle trades across nearly the entire market without the advertised secondary check. This materially increases LP exposure during a primary oracle failure, including the omitted Lazer exponent path that a present and fresh Chainlink backup would reject.

**M-15** Most pairs lack effective oracle backups

CONTINUED

**— RECOMMENDATION**

Configure independent backup feeds for every material pair where a suitable source exists. Use a per feed maximum age consistent with that feed's update policy instead of one global 120 second window. Fail closed when a required backup is missing or stale for actions that create or increase protocol risk, including opens, position increases, margin withdrawals, and guaranteed SL updates. Do not blanket fail closed for exits and liquidations, since that can trap traders and leave insolvent positions open. Route those actions through a separately defined emergency source or conservative settlement policy while new exposure remains disabled.

**— RESOLUTION**

Status: Acknowledged.

MEDIUM **M-16**

RESOLVED

# Intent coin OI bypasses exposure caps

|          |                                                                              |
|----------|------------------------------------------------------------------------------|
| CATEGORY | Logical Error                                                                |
| LOCATION | IntentCallbackModule.sol:executeTwapOpenCallback(), executeRfqOpenCallback() |
| REVIEW   | Main Review 1                                                                |

## DESCRIPTION

Coin-denominated TWAP and RFQ opens check exposure using the coin amount valued at the raw oracle price. The callback then applies opening spread, and registration values the same coin amount at the spread-adjusted open price.

```
```plain text
```

```
uint256 posSize = slice.coinExposure.coinToUsdcAtPrice(price);  
newTrade.openPrice = uint256(pairInfos.applySpread(...));
```

```
if (!storageT.withinExposureLimits(..., posSize, ...)) {  
    revert Errors.ExposureLimitsBreached();  
}
```

```
``
```

For a long, opening spread raises the registered notional above the amount that passed the limit check. For example, 0.01 BTC at \$50,000 passes a \$500 cap, but a 5% spread registers \$525 of long OI. This can exceed pair-side, pair-total, group, global, and wallet exposure caps. Shorts are checked conservatively because their opening spread reduces the registered notional. The bypass is bounded by the configured spread, but it defeats the protocol's primary OI risk limits.

RECOMMENDATION

For coin-denominated slices, derive the exposure delta and minimum-position value from the final `newTrade.openPrice`. Pass that exact spread-adjusted delta to `withinExposureLimits()` and registration.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-17**

RESOLVED

Short increases enable triggered SL

CATEGORY Logical Error
LOCATION PairStorageExtension.sol:normalizeTpSl()
REVIEW Main Review 1

DESCRIPTION

Increasing a position recalculates its TP and SL around the blended open price. `normalizeTpSl()` treats `sl == 0`, which means no stop loss, as an actual price when calculating the old SL profit percentage.

```
``plain text
int256 slProfitPercent = currentPercentProfit(
eP.openPrice,
eP.sl,
eP.buy,
eP.leverage,
eP.pairIndex
);
`
```

For a short, price zero represents maximum positive profit. Normalization therefore replaces the disabled SL with a nonzero SL below the new open price. Since short SLs trigger when `price >= sl`, the current market price can already satisfy the new stop immediately after the increase.

Illustration: a trader opens a 10x short at 50,000 with `sl = 0`. When the position is increased at the same price, `currentPercentProfit(50,000, 0, false, 10x, pair)` treats the disabled SL as a profitable move and clamps it to the pair max gain, 500% in the local fixture. `_priceWithProfitPercent(50,000, 500%, 10x, false)` then writes 25,000 as the new SL. The market price is still 50,000, so the short SL predicate `price >= sl` is already true.

This affects standard and coin-exposure increases, including later TWAP and RFQ slices that merge into an existing position. An honest keeper can immediately close the trader's position and charge another close fee even though the trader never enabled an SL. Short open TWAPs with more than one slice are effectively broken.

RECOMMENDATION

M-17 Short increases enable triggered SL

CONTINUED

Preserve the sentinel explicitly: if the stored SL is zero, return `correctedSl = 0` without normalizing it. Apply the same sentinel handling to any optional trigger price.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-18**

ACKNOWLEDGED

Guaranteed stops enable cheap straddles

CATEGORY Gaming

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/LimitCallbackModule.sol#L189>

REVIEW Main Review 1

— DESCRIPTION

Guaranteed-SL pairs settle an adverse stop at the stored SL even when the executable oracle price has gapped beyond it. `correctSl` caps only the maximum permitted loss and does not enforce a minimum distance from entry.

```

`plain text
values.price = guaranteedSlEnabled && orderType == SL
? profitLockingSl ? pricePostSpread : existingTrade.sl
: pricePostSpread;
`

```

A trader can open equal long and short positions at high leverage and place both stops immediately around entry. The first price move closes only the losing leg near entry, regardless of how far the oracle moved, while the profitable leg captures the full move. If price reverses, the surviving leg's opposite near-entry stop similarly caps its loss. The trader has paid only fees and a very small bounded stop loss for direction-neutral exposure to a large move, while the vault funds the convex payout.

In the PoC, the trader opens two 100 USDC, 100x BTC positions at 50,000 with stops at 49,999 and 50,001. A move to either 50,500 or 49,500 closes the losing leg at its stored stop and the winning leg at market. Both directions return approximately 274 USDC, producing about 74 USDC net profit after fees on 200 USDC collateral.

— RECOMMENDATION

Do not provide unlimited gap protection for stops placed arbitrarily close to entry. Enforce a minimum guaranteed-SL distance that covers spreads, fees, and expected oracle latency, charge an explicit guarantee premium, cap the compensated gap, or settle at the worse of the stored stop and executable price.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/b1abe20a4d674cfc21a754742546adfc7b3a1b41>

M-18 Guaranteed stops enable cheap straddles

CONTINUED

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-19**

ACKNOWLEDGED

Builder can change fees after signing

CATEGORY Unexpected Behavior
LOCATION Eip7702Template.sol
REVIEW Main Review 1

DESCRIPTION

An EIP-7702 execution signature commits to the mode, calls, and nonce, but not the selected builder's fee configuration.

```
``plain text
bytes32 digest = _computeDigest(mode, calls, nonce);
`
```

During execution, `_chargeBuilderFees()` instead reads the builder code's current fee mode, amount, and collector. The builder owner may change all of these through `modifyBuilderCode()` after the user signs.

Consequently, a builder can observe a signed batch, raise its fee, change the recipient, and execute the unchanged signature. The PoC shows a user signing while the fixed fee is 1 USDC, after which the builder raises it to 10 USDC and the same signed payload charges 10 USDC. The loss is bounded by the deployed builder fee caps, currently 10 USDC fixed or 1% per eligible call, but it is not authorized by the user's signature.

RECOMMENDATION

Commit the builder code and a fee quote to the signed payload. At minimum, sign a maximum fee, expected collector, and deadline, then require the live configuration to remain within those terms.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-20**

RESOLVED

Position rolls omit funding settlements

CATEGORY	Logical Error
LOCATION	src/TradingStorage/TradingStorageCoreExtension.sol:positionRoll, _adjustRollCompute
REVIEW	Main Review 1

DESCRIPTION

`positionRoll` obtains accrued funding fees through `_adjustRollCompute` and applies them when recomputing the trade's collateral and leverage, but it never records that settlement. It calls `vaultManager.allocateRewards` for the borrow fees and then replaces the trade's funding snapshot, unlike the close, margin update, and position increase paths which call `pairInfos.recordFundingFees`.

```plain text

```
(r.newLeverage, r.newCollateral) = _calculateNewLeverage(
existingTrade.leverage,
existingTrade.initialPosToken,
existingTrade.openPrice,
r.marginFees,
r.fundingFees,
_newOpenPrice
);
```

```
pairInfos.storeTradeInitialAccFundingFees(
existingTrade.trader, existingTrade.pairIndex,
existingTrade.index, existingTrade.buy, _currentPrice
);
```

Resetting the snapshot makes the omission permanent, so `totalFundingPaid`, `totalFundingReceived`, and therefore `fundingOwed` no longer represent settled funding. `fundingOwed` is not audit only. It reduces the balance VaultManager treats as spendable in two places:

```plain text

```
int256 fundingOwed = storageT.pairInfos().fundingOwed();
int256 balanceAvailable = int256(storageT.usdc().balanceOf(address(this))) -
int256(totalRewards) - fundingOwed;
```

The same expression gates `withdrawFromBuffer`. If rolls deduct 10,000 USDC of positive funding from payer collateral, `fundingOwed` stays understated by 10,000 USDC, so

M-20 Position rolls omit funding settlements

CONTINUED

`balanceAvailable` is overstated by the same amount. VaultManager then pays traders out of buffer USDC that was reserved for funding receivers, and a later receiver payout draws the missing amount from the junior tranche instead. Governance can also withdraw more from the buffer than it should. The drift accumulates silently with every roll, since rolls are performed by a trusted keeper role rather than triggered by an attacker.

— RECOMMENDATION

Record the roll's funding fees before replacing the funding snapshot. Because `recordFundingFees` is currently `onlyCallbacks` and the roll executes in the storage context, add a narrowly authorized roll settlement entrypoint in PairInfos or route the update through the callback layer, rather than broadly widening access to `recordFundingFees`.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-21**

ACKNOWLEDGED

Backup deposits reset funding at zero

| | |
|----------|--|
| CATEGORY | Logical Error |
| LOCATION | src/PriceAggregator.sol:_backupOnlyPrice, src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback, src/PairInfos/PairInfos.sol:_storeAccFundingFees, src/PairInfos/PairInfos.sol:storeTradeInitialAccFundingFees |
| REVIEW | Main Review 1 |

DESCRIPTION

Single-order fulfillment permits backup-only mode, and `_backupOnlyPrice` returns zero for every non-liquidation order type. `updateMarginCallback` only rejects a zero price on the withdrawal branch:

```
``plain text
if (pendingMarginUpdate._type == ITradingStorage.updateType.DEPOSIT) {
    storageT.transferUSDC(existingTrade.trader, address(storageT), pendingMarginUpdate.amount);
} else {
    if (price == 0) revert Errors.ZeroPrice();
}
...
pairInfos.storeTradeInitialAccFundingFees(..., price);
`
```

Margin deposits therefore proceed with `price == 0`. The callback computes borrow and funding fees at a current price of zero and then writes both the pair-level and trade-level funding snapshots at that price, while `lastUpdateBlock` advances to the current block.

During backup-only mode a trader can deposit a trivial amount of margin purely to force this path. Doing so erases funding accrual that was pending for the pair since the last update, because the accumulator advances over that block range using a zero price, and it resets the trade's funding baseline to a price the market can never reach. The trader escapes accrued funding they owed, and the loss is absorbed by the counterparty side and the vault. The trigger is permissionless and repeatable for as long as backup-only mode is active, which is precisely the degraded state where accounting integrity matters most.

RECOMMENDATION

Reject zero prices for all margin update types, not only withdrawals. If backup-only mode is intended to keep supporting deposits, derive and validate a real backup price for `UPDATE_MARGIN` rather than returning zero, and never advance funding accumulators or write funding snapshots from a zero price.

M-21 Backup deposits reset funding at zero

CONTINUED

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-22**

RESOLVED

Stale close notional inflates rebates

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | src/Trading/RfqTradingModule.sol:_validateCloseReq, src/Trading/TwapTradingModule.sol:closeTwap, src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallbackWithCoinExposure, src/TradingCallbacks/TradeRegistry.sol:unregisterTrade |
| REVIEW | Main Review 1 |

DESCRIPTION

RFQ and TWAP close orders never set `beingMarketClosed`, so the guard that is meant to prevent overlapping close lifecycles is one-directional: a market close blocks another market close, but an RFQ or TWAP reduction can settle underneath a queued market close.

A trader can queue a near-full RFQ or TWAP reduction and then queue a full coin-exposure market close before the reduction executes. If the reduction settles first, the market-close callback reloads the now-reduced trade but still derives the close notional from the stale coin exposure captured in the pending order:

```
``plain text
```

```
values.levPosToken = pendingOrder.coinExposure.coinToUsdcAtPrice(existingTrade.openPrice);
uint256 closeFee = values.levPosToken *
aggregator.pairsStorage().pairCloseFeeP(existingTrade.pairIndex, pendingOrder.coinExposure,
existingTrade.buy) /
100 /
PRECISION;
```

Only `closeCollateral` is subsequently clamped to the remaining collateral; `values.levPosToken` and `closeFee` are not. The close fee is therefore computed against the full pre-reduction notional while the position being settled is only a dust remainder.

Because the referral rebate is derived from that inflated fee, a self-referring trader can mint an oversized rebate against a near-empty position. The rebate is paid out of settlement accounting, so the excess is extracted from the vault and honest traders. The setup requires no privileged access and both legs are trader-controlled.

RECOMMENDATION

Lock a position against all overlapping close lifecycles rather than market closes alone, by setting and honouring `beingMarketClosed` (or an equivalent flag) for RFQ and TWAP closes as well. Independently, recompute the close notional, fee tier, and referral rebate from the clamped live position after `closeCollateral` is capped, so fees can never be charged on size that no longer exists.

M-22 Stale close notional inflates rebates

CONTINUED

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-23**

ACKNOWLEDGED

LPs Can Dodge Trader Wins

CATEGORY Gaming
LOCATION GLOBAL
REVIEW Main Review 1

— DESCRIPTION

LP share accounting is based on the tranche's current asset balance and reserved liquidity, but it does not reflect pending / unrealized trader profits before those profits are settled.

If traders are in big profit, but didn't realize it yet the vault already has an economic liability to that trader. However, until the close is executed, that liability is not pulled from the tranche. LPs can redeem at the pre-loss share price, avoid the upcoming trader payout, and then re-enter the system after the loss has been realized by the remaining LPs.

This gaming vector will avoid loss for the LPs who abuse it and socialize more loss on the honest passive LPs.

Open / exit fees on the LP side reduce the profitability and frequency of this behavior, but they do not fully mitigate it. Fees only add friction; they do not make LP shares reflect pending trader PnL or prevent an LP from exiting before a large unrealized trader win is booked.

— RECOMMENDATION

Consider to ensure that LP shares always reflect the current economic state of the system.

Otherwise make sure that LP exit fees are priced by the system's state in a way that drastically reduces the likelihood of this gaming vector being profitable.

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-24**

RESOLVED

Close Intents Consume Open Slots

CATEGORY DoS

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/70f16f3e40d0032e075ae19ed9690be143d817ee/src/Trading/MarketTradingModule.sol#L196-L201>

REVIEW Main Review 1

DESCRIPTION

The open-trade validation enforces `maxTradesPerPair` using `openTradesCount + pendingMarketOpenCount + openLimitOrdersCount`. However, `openLimitOrdersCount` is also incremented for RFQ/TWAP close intents, not only for orders that can create new positions.

As a result, active close intents can consume the same per-pair slots used to decide whether a trader may open a new trade. A trader with open positions and pending RFQ/TWAP close intents can therefore hit `MaxTradesPerPair` and be blocked from opening new trades, even though those close intents do not create additional open positions.

This occurs because `storeTwapCloseOrder` and `storeRfqCloseOrder` increment `openLimitOrdersCount`, and the market/RFQ/TWAP open paths include that counter in the open-slot limit.

RECOMMENDATION

Consider to separate counters for opening intents and closing intents.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-25**

ACKNOWLEDGED

One-sided funding creates phantom liability

CATEGORY Logical Error
 LOCATION src/PairInfos/PairInfos.sol:getPendingAccFundingFees
 REVIEW Main Review 1

DESCRIPTION

`getPendingAccFundingFees` charges funding to the existing side even when the receiving side has zero open interest:

```

`plain text
if (openInterestCoinLong > 0) {
valueLong += (fundingFeesPaidByLongs + hedgingFeePaidByLongs) / int256(openInterestCoinLong);
}
if (openInterestCoinShort > 0) {
valueShort += (fundingFeesPaidByLongs + hedgingFeePaidByLongs) * (-1) /
int256(openInterestCoinShort);
}
`

```

A later counterparty cannot receive that funding. During registration, `TradeRegistry` materializes and snapshots the current funding accumulator before `storeTrade` inserts the new position into coin OI. The new position therefore accrues only funding created after it opened. The following sequence reproduces the resulting accounting drift:

1. A long is the only position for 50,000 blocks, so its funding accumulator increases.
2. A short opens and snapshots the current short accumulator.
3. Both positions close without another block of accrual.

```

`vb.net
one-sided long funding: 633982
total funding paid: 633982
total funding received: 0
funding owed after both sides close: 633982
long coin OI: 0
short coin OI: 0
`

```

The long's settlement increases `totalFundingPaid`, but no position can settle the corresponding historical credit into `totalFundingReceived`. `fundingOwed()` consequently treats

M-25 One-sided funding creates phantom liability

CONTINUED

the residual as a permanent global trader liability even after no position can claim it.

`VaultManager` subtracts this phantom liability from the balance available for trader payouts and governance withdrawals. Payouts therefore pull from the junior tranche earlier than necessary, and the retained funding remains unavailable to LPs or governance. The drift can accumulate naturally whenever a pair has only one open side. A trader can accelerate it as a grieving action, but must pay the funding they create.

— RECOMMENDATION

Do not create trader to trader funding while the receiving side has zero open interest. If one-sided funding is intended as compensation for LPs or the protocol, account for it in an explicit revenue bucket rather than `fundingOwed`. Ensure that no funding liability remains once all positions that could receive it have closed.

— RESOLUTION

Status: Acknowledged.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-26**

RESOLVED

Pair updates leave usdOI unsynced

| | |
|----------|--|
| CATEGORY | Logical Error |
| LOCATION | <code>\`src/PairStorage/PairStorage.sol\`, \`updatePair\` (\`groupIndex\`, \`isUSDCAligned\`)</code> |
| REVIEW | Main Review 1 |

— DESCRIPTION

`TradingStorage` only updates global `usdOI` when the pair's current `groupIndex` is `2` (FOREX), and it books long/short notional into `usdOI[0]` / `usdOI[1]` using the current `isUSDCAligned` flag.

`PairStorage.updatePair` can change both fields while open interest still exists, and there is no migration or rebucketing of `usdOI`.

`\`plain text`

```
if (pairsStored.pairGroupIndex(_pairIndex) == 2) {
  bool isUSDCAligned = pairsStored.isUSDCAligned(_pairIndex);
  // books into usdOI[index] or usdOI[inverseIndex]
}
```

After a live reconfiguration:

1. Flip `isUSDCAligned` while positions are open. Opens were credited to one USD bucket; later closes debit the opposite bucket. That underflows `usdOI` and reverts OI updates, so affected FOREX opens, closes, and liquidations can brick.
2. Move a pair out of group `2` with residual OI. Later closes no longer touch `usdOI`, so the old notional stays stuck and permanently distorts blended skew / borrow fees for every FOREX market.
3. Move a pair into group `2` with existing OI. Closes try to subtract from `usdOI` that never received the open, again underflowing.

The failure mode does not require a malicious governor: an operational pair reclass or alignment fix is enough, and there is no onchain repair for the desynced buckets.

Illustration: open `100k` USDC long on an aligned FOREX pair (`usdOI[0] += 100k`), then set `isUSDCAligned = false`. A full close executes `usdOI[1] -= 100k`. If `usdOI[1] < 100k`, settlement reverts.

— RECOMMENDATION

M-26 Pair updates leave usdOI unsynced

CONTINUED

Keep `usdOI` consistent with current pair classification: either forbid `groupIndex` / `isUSDCAligned` changes while the pair has open interest, or migrate `usdOI` atomically when those fields change so every open notional remains in the bucket that later closes will debit.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-27**

RESOLVED

Coin RFQ opens exceed OI caps and underflow

CATEGORY Informational

LOCATION `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:openRfqCallback():L396-L427` `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerTradeWithCoinExposure():L120-L158` `packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695`

REVIEW Main Review 1

— DESCRIPTION

For coin-sized RFQ opens, `openRfqCallback` values `coinExposure` at the raw oracle price for `withinExposureLimits`, but `TradeRegistry` derives leverage and canonical OI from the higher spread-adjusted long execution price. An order whose raw notional consumes the remaining cap therefore passes validation yet stores and reserves more OI than the pair, side, wallet, and global amounts actually checked. Once pair OI exceeds `pairMaxOI` at one-sided skew, `getPendingAccRolloverFees` subtracts `utilization\*skew` greater than 100% from `1e12` and reverts by underflow.

Impact: A routine operator fill of an unprivileged coin RFQ can push a pair above its configured solvency cap and make rollover-dependent settlement revert. Market closes, liquidations, margin updates, and subsequent opens/increases on the pair can remain blocked, trapping users and allowing losing positions to accrue uncovered loss until governance changes configuration or deploys a fix.

— RECOMMENDATION

Compute the spread-adjusted execution price before exposure validation and pass the exact OI that `TradeRegistry` will store and reserve to every total, side, wallet, group, and global cap check. Apply the same fix to all coin-sized intent paths. As defense in depth, bound or saturate rollover multiplier math when `utilization` reaches or exceeds 100%.

— RESOLUTION

Status: Resolved.

MEDIUM **M-28**

RESOLVED

Stale close fee fabricates profit and erases funded LP PnL rewards

| | |
|----------|---|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:_executeCloseTwapSlice():L268-L300 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:_closeTwapSliceUnregister():L326-L344 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeValuePure():L865-L885 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:unregisterTrade():L332-L370 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:decrementPnlRewards():L304-L309 |
| REVIEW | Main Review 1 |

DESCRIPTION

After a concurrent close reduces a TWAP target, the full-close clamp corrects collateral and coin size but retains the original leveraged notional. The dust close therefore supplies an uncollectible fee far larger than live collateral. At zero market PnL, PairInfos' liquidation-threshold branch changes the returned PnL to `fees - collateral`, and TradeRegistry treats that artificial positive value as realized profit and decrements the global `pnlRewards` bucket. The passing PoC first funds that bucket through a real losing trade, then reduces a 1,000-USDC 150x position to 1 USDC and executes its stale full-size TWAP. Although the trader receives zero from the dust close, the 142.5-USDC post-discount stale fee erases exactly 141.5 USDC of funded PnL-reward accounting. The previously demonstrated referral rebate remains claimable. Impact: A trader can repeatedly erase LP reward accounting funded by real trader losses while sacrificing only dust. The backing USDC is not immediately transferred or burned by `decrementPnlRewards`, but it becomes de-reserved for ordinary settlement and LPs can no longer receive it through normal PnL-reward distribution. At the demonstrated allowed fee policy and maximum leverage, each 1-USDC dust close erases about 141.5 USDC of reward entitlement, in addition to the same root cause's already proven profitable referral extraction and pooled-liquidity harm.

RECOMMENDATION

Whenever a close amount is clamped or dust-promoted, recompute leveraged notional from the final collateral and coin size, and enforce that fee notional cannot exceed the notional actually unregistered. PairInfos should not report positive economic PnL merely because fees exceed collateral, and VaultManager should decrement `pnlRewards` only for profit actually paid to the trader.

RESOLUTION

Status: Resolved.

MEDIUM **M-29**

ACKNOWLEDGED

TWAP race cashes replacement exposure out at a pre-existence price

| | |
|----------|---|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/Trading/TwapTradingModule.sol:_validateTwapCloseOrder():L404-L421 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageIntentExtension.sol:storeTwapCloseOrder():L282-L304 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:closeTwapCallback():L203-L255 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallback():L324-L384 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_pythCorePrice():L284-L307 |
| REVIEW | Main Review 1 |

— DESCRIPTION

A TWAP close admitted before beingMarketClosed is set neither acquires the mutex nor rechecks it during settlement. It can delete P1 after P1's market close is pending, leaving the generation-blind market order alive. P2 can then reuse the tuple, and the stale order can settle P2 using a still-valid Pyth observation published before P2 existed. The paired fork PoC proves the observation predates P2 while satisfying the 60-second freshness and 2% Chainlink-deviation checks.

Impact: From identical post-P2 state, settlement at the stale pre-P2 price pays over 2,000 USDC more than a legitimate contemporaneous close. The paired balance assertions show the incremental payout is an equal incremental loss of pooled VaultManager/tranche/TradingStorage USDC. The extraction can be repeated subject to oracle, OI, gain-cap, and liquidity limits.

— RECOMMENDATION

Acquire and honor a generation-bound exclusive lock for TWAP close lifecycles, and release it only on completion or cancellation. Bind pending market closes to the original trade timestamp/generation and cancel on occupant mismatch. Reject oracle observations older than the target position generation.

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-30**

ACKNOWLEDGED

Newest Pyth Lazer report can predate RFQ exposure and strip enlarged-position collateral

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageIntentExtension.sol:storeRfqOpenOrder():L360-L383 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:openRfqCallback():L425-L451 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncrease():L215-L225 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_pythLazerPrice():L310-L318 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_validateTimestamp():L343-L348 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L469

REVIEW Main Review 1

— DESCRIPTION

A residual RFQ increase checkpoints a new blended entry, leverage, collateral, OI, and mutation timestamp, but UPDATE_MARGIN accepts any age-valid Lazer observation without requiring it to postdate that checkpoint. The PoC first accepts P0 on chain, retains a strictly newer P0 payload, then creates 90,000 of the final 100,000 USDC OI through a later RFQ slice. Although the retained report is newest by feed timestamp, it predates most of the liability. The callback combines current enlarged OI with that pre-exposure valuation and permits withdrawal; the identical action using the actual post-mutation price reverts its withdrawal-equity gate.

Impact: The passing PoC strips 333 USDC while preserving 100,000 USDC OI, leaves the losing leg near 150x, and turns matched exposure into over 300 USDC attacker profit with corresponding protocol-liquidity loss. The mismatch scales across staged RFQs up to OI and available liquidity limits.

— RECOMMENDATION

Use the already-checkpointed position mutation time (or a dedicated monotonic nonce) as a lower bound on oracle publish time for collateral-removing updates, or serialize increases and margin withdrawals under a generic position mutex. Keep a feed high-water mark as defense in depth, but it cannot replace position-epoch binding.

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-31**

ACKNOWLEDGED

RFQ close lets a stale market close cash out a replacement position

| | |
|----------|--|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/Trading/RfqTradingModule.sol:_validateCloseReq():L326-L337 packages/avantis-contracts-team2-1783453751962/src/Trading/RfqTradingModule.sol:executeCloseRfqSlice():L268-L295 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:closeRfqCallback():L479-L524 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallback():L324-L384 |
| REVIEW | Main Review 1 |

— DESCRIPTION

An RFQ close checks beingMarketClosed only when its lifecycle is created, but does not acquire the mutex or recheck it when slices execute. A market close can consequently be queued after the RFQ. The RFQ can then delete the original trade and its TradeInfo while leaving the market close pending. After a new trade reuses the tuple, the market callback has no timestamp binding and settles the old collateral request against the replacement trade's live side, leverage, and entry price.

Impact: A normally validated stale market-close price can realize PnL on high-leverage replacement exposure that did not exist at the price observation, forcibly deleting the replacement and paying unearned profit from pooled VaultManager/tranche USDC. The passing fork PoC uses a fresh Pyth report within the Chainlink deviation band and pays more than twice the replacement collateral; the race can scale subject to oracle freshness/deviation, pending-order, OI, gain-cap, and liquidity limits.

— RECOMMENDATION

Serialize all close lifecycles with one mutex, acquired at market/RFQ/TWAP close creation and released only on completion or cancellation. Bind pending market closes to the target trade timestamp/generation and cancel without settlement on mismatch; recheck close ownership before each RFQ slice.

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-32**

ACKNOWLEDGED

LP withdrawals retroactively reprice aged rollover and instantly liquidate healthy traders

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getBlendedUtilizationRatio():L612-L617 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:pairMaxOI():L482-L489 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:_withdraw():L292-L308

REVIEW Main Review 1

— DESCRIPTION

For dynamically capped pairs, pairMaxOI derives from current tranche assets. LP withdrawals can therefore shrink capacity without checkpointing rollover. PairInfos then applies the post-withdrawal utilization multiplier to every block since the prior checkpoint, manufacturing historical borrow debt. The PoC uses live economic parameters and realistic two-second block progression: after 33 days at 19% utilization the victim is healthy, but a strategic LP owning 80% of shares withdraws 76.25% of TVL, raises current utilization to 80%, and increases the fee for the unchanged interval by over sixfold without another block passing. Impact: A sufficiently large LP can withdraw redeemable liquidity and instantly move fee-paying positions' liquidation boundaries at an unchanged price. The PoC completes an ordinary liquidation immediately after the withdrawal. The manipulation can affect an entire position side and becomes permanent when settlement checkpoints the retroactively inflated accumulator. Exploitation requires control of enough redeemable shares and reserve headroom; the demonstrated case uses 80% share ownership.

— RECOMMENDATION

Checkpoint rollover before every mutation that changes dynamic OI capacity, or time-integrate historical capacity/utilization rather than pricing the full interval from the current endpoint. Alternatively, decouple rollover capacity from instantaneous permissionless tranche assets. Apply checkpoint-before-change consistently to OI, cap configuration, and vault-derived capacity inputs.

— RESOLUTION

Status: Acknowledged.

M- 32 LP withdrawals retroactively reprice aged rollover and instantly liquidate healthy traders CONTINUED

Guardian: Rollover borrowing fees accrue lazily from the last checkpoint using the capacity read at the next update. An LP withdrawal reduces `junior.totalAssets()`, which lowers pair capacity and raises utilization, but `AvantisTranche._withdraw` does not checkpoint rollover accumulators first.

The next accumulator update therefore applies the post-withdrawal utilization and multiplier to every block since the previous checkpoint, including blocks before the withdrawal. An LP can abruptly increase aged borrowing fees and potentially make previously healthy traders liquidatable. The new saturation logic prevents arithmetic failure at extreme utilization but does not prevent this retroactive repricing.

****Recommendation****

Checkpoint rollover fees for every affected pair before tranche assets or another capacity input changes.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-33**

RESOLVED

Ol-cap reduction retroactively confiscates aged positions' collateral

CATEGORY Informational

LOCATION `packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:setAssetOIParams():L219-L224` `packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695` `packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:_storeAccRolloverFees():L905-L912`

REVIEW Main Review 1

— DESCRIPTION

`setAssetOIParams` replaces the hard-cap denominator without checkpointing `PairInfos`' lazy rollover accumulator. `getPendingAccRolloverFees` then applies the new utilization/skew multiplier to every block since `lastUpdateBlock`, including blocks before the cap change. The production-consistent PoC ages a normal position for 15,768,000 blocks and 365 days at 1% utilization, then reduces the cap to a value still above live OI. The fee rises by over 1,000x and a flat-price market close pays zero; checkpointing before the cap change preserves over 99% of collateral.

Impact: A legitimate hard-cap reduction can manufacture historical rollover debt, wipe healthy positions at an unchanged price, move liquidation boundaries, and allocate the repriced charge upon settlement. The effect can apply across the pair's aged live OI; increasing the cap can inversely erase accrued LP compensation.

— RECOMMENDATION

Checkpoint `PairInfos` rollover under the old cap before changing asset OI parameters, or version cap parameters by activation block and accrue piecewise. Reject unsafe caps and make the nonlinear denominator total at all utilization values.

— RESOLUTION

Status: Resolved.

MEDIUM **M-34**

RESOLVED

Unbounded rollover curve freezes at cap and confiscates just below it

| | |
|----------|--|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getBlendedUtilizationRatio():L612-L617 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:minMaxRollOverFee():L785-L801 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:pairMaxOI():L482-L489 |
| REVIEW | Main Review 1 |

— DESCRIPTION

The rollover multiplier uses an unbounded quotient whose denominator is `1e12 - blendedUtilization * skew`. At exactly full one-sided utilization the denominator is zero, and above it subtraction underflows, freezing every settlement path that checkpoints rollover. More severely, immediately below the cap the expression remains finite but becomes arbitrarily large. A permissionless trader can fill the remaining exposure capacity, checkpoint a previously healthy victim's long historical interval at the ordinary rate, and then let one realistic block accrue at the near-singular multiplier. The victim's flat-price market close succeeds but the annual fee clamp consumes the entire collateral, while the fresh filler pays negligible fees because it was exposed only for the final block. The revised end-to-end PoC advances block and timestamp consistently, uses ordinary exposure-valid opens and oracle settlement, asserts the pre-attack victim payout exceeds 90%, and then proves the same position closes for zero after the cap fill.

Impact: An unprivileged cap filler can confiscate the full collateral of older same-side positions without adverse price movement while bearing only negligible rollover cost itself. At the exact or negative denominator boundary, all borrow-fee-dependent closes, liquidations, margin updates, increases, and forced closes instead revert, creating the previously identified pair-wide settlement freeze. The direct, successful zero-payout settlement just below the cap is strictly higher impact than the recoverable denial of service alone.

— RECOMMENDATION

Replace the asymptotic quotient with one bounded, continuous rollover curve shared by authoritative settlement and quoting code. Explicitly handle inputs at or above 100% utilization/skew without division, underflow, or a fallback to a low baseline. Cap the per-block multiplier/rate before fee accumulation so a single block cannot exhaust collateral, and preserve risk-reducing settlement under every legacy or over-cap state.

M-34 Unbounded rollover curve freezes at cap and confiscates just below it**CONTINUED**

— RESOLUTION

Status: Resolved.

MEDIUM **M-35**

RESOLVED

Legacy PnL positions create repeatable unfunded funding withdrawals

| | |
|----------|--|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:updatePairParams():L153-L170 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeFundingFee():L932-L951 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccFundingFees():L963-L992 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L500 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:_calculateNewLeverage():L512-L529 |
| REVIEW | Main Review 1 |

DESCRIPTION

Disabling PnL order mode affects only future orders, leaving existing PnL positions and their coin OI intact. Funding accrual includes that OI when calculating and socializing side-wide payments, but `getTradeFundingFee` always exempts the persistent PnL position from settlement. After a normal PnL-to-fixed transition, an attacker can retain large legacy PnL OI on the nominal paying side, open a smaller fixed-fee receiver on the other side, and crystallize its unbacked negative funding through repeated margin withdrawals. The PoC does not use partial closes and proves the PnL payer remains fully open and unchanged, so this is not a duplicate of partial-close debt forgiveness.

Impact: The receiving position can repeatedly withdraw real USDC while retaining essentially the same collateral, leverage, and OI. Once manager-held cash is insufficient, settlement pulls from the LP tranche. The drain scales with legacy skew, funding rate, elapsed time, and OI limits and persists until the exempt position is closed or converted; external hedging can reduce the attacker's directional exposure.

RECOMMENDATION

Use the same population for funding contribution and settlement: either charge PnL positions whose OI contributes to funding, or exclude their OI from funding calculations. Prevent fee-mode transitions while incompatible positions remain live, or atomically checkpoint and convert or close them during migration.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-36**

RESOLVED

Stale funding reserve enables ERC4626 share dilution and LP asset theft

CATEGORY Informational

LOCATION `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbac
kModule.sol:updateMarginCallback():L445-L480 packages/avantis-contracts-team2-17
83453751962/src/VaultManager.sol:_sendUSDCToTrader():L488-L501 packages/avantis-
contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:recordFundingFees():L3
43-L365 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:wi
thdrawAsVaultManager():L144-L147 packages/avantis-contracts-team2-1783453751962/
src/AvantisTranche.sol:previewDeposit():L172-L174`

REVIEW Main Review 1

DESCRIPTION

In a legitimate two-sided market, `updateMarginCallback` pays a funding recipient through `sendUSDCToTrader` before recording the in-flight negative funding fee. `fundingOwed` therefore overstates the liability during `balanceAvailable` calculation, causing an excessive immediate tranche draw despite sufficient raw manager cash. The later record releases the reserve but does not restore ERC4626 NAV. Passing PoCs demonstrate live reservations exceeding assets and direct extraction: with no pending `pnlRewards` at entry, an untrusted depositor mints excess shares at the artificially depressed NAV; matched zero-sum settlement later returns the displaced surplus through ordinary rewards after all OI closes; the attacker redeems 9,314.826241 USDC from a 9,000 USDC deposit.

Impact: The payout-before-recording defect creates an attacker-timed ERC4626 NAV decrease, excess share minting, and capture of returned assets from existing LPs. At high utilization it also breaks reserve backing and freezes LP withdrawal capacity. Unlike the legacy-PnL variant, this occurs with fully backed ordinary two-sided funding and remains exploitable if payer-population compatibility is fixed.

RECOMMENDATION

Record the current recipient funding credit before calling `sendUSDCToTrader`, relying on transaction atomicity to revert the audit if payout fails; alternatively pass the in-flight funding delta and compute liquidity using `postSettlementFundingOwed`. Add a two-sided invariant that recipient settlement cannot draw from the tranche when raw manager cash net of post-settlement liabilities covers the payout.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-37**

ACKNOWLEDGED

Low-leverage longs can promise more profit than the tranche reserves

| | |
|----------|---|
| CATEGORY | Informational |
| LOCATION | <code>packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerTrade():L55-L106</code> <code>packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerTradeWithCoinExposure():L114-L167</code> <code>packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:correctTp():L228-L237</code> <code>packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:_reserveBalance():L248-L251</code> <code>packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:maxWithdraw():L332-L337</code> |
| REVIEW | Main Review 1 |

— DESCRIPTION

Opening settlement reserves post-fee collateral multiplied by leverage, while take-profit correction and PnL capping permit `maxGainP` percent of collateral independently of leverage. A long whose allowed leverage is below `maxGainP/100` can therefore create a maximum profit liability greater than the liquidity locked for it. The passing fork PoC uses normal router, oracle, callback, and vault paths: two allowed 2x longs reserve 600 USDC although each can earn 500% of collateral, LPs redeem to the protocol-advertised reserve boundary, the first winner receives its full corrected-TP payout, and the tranche then holds less than the second trade's remaining reservation. The second equally valid winning settlement reverts and remains live. Fork state also confirms deployed pair configuration permits 1x leverage with 2,500% maximum profit, so the mismatch is not limited to an artificial test parameter.

Impact: Legitimate long-side appreciation can make maximum trader payouts exceed the assets protected from LP withdrawal. Early winners can consume manager collateral and tranche principal, leaving later winning positions unclosable, remaining LP withdrawals frozen, and live reservations greater than tranche assets. This creates a settlement race and potential protocol-wide insolvency under deployed risk parameters.

— RECOMMENDATION

Align liquidity reservation and withdrawal capacity with maximum LP-funded payout. Reserve at least $\max(\text{collateral} \times \text{leverage}, \text{collateral} \times \text{maxGainP} / 100)$, accounting consistently for collateral already held by VaultManager, or enforce $\text{leverage} \geq \text{maxGainP} / 100$. Apply the invariant to ordinary, fixed-coin, limit, TWAP/Rfq, and increase registration paths, and add tests that all maximum-profit liabilities remain payable after maximum permitted LP withdrawals.

— RESOLUTION

Status: Acknowledged.

MEDIUM **M-38**

RESOLVED

Pause allows OI migration race

CATEGORY Warning

LOCATION Contract Upgrade & State Migration – Auditor Review Doc

REVIEW Main Review 1

DESCRIPTION

The migration runbook pauses `TradingRouter` to freeze position state while OI is scanned and written. The operator-only `executeMarketOrder` entypoint has no `whenNotPaused` check, so a market open or close submitted before the pause can still settle during the supposed freeze.

OI migration writes absolute assignments:

```
``plain text
_walletOI[_users[i]][_pairIndexes[i]] = _values[i];
_openInterestCoinOI[_pairIndexes[i]][_indexes[i]] = _values[i];
``
```

If a pending order settles between the off-chain scan and the risk-manager write, the absolute migration assignment overwrites the post-settlement truth. Undercounted wallet OI later underflows on close because its decrement has no clamp. Incorrect coin or wallet OI also distorts exposure limits and funding. A pending ZFP open can additionally settle after the force-close scan and survive the product migration.

This is sharper than a generic pause gap: the migration procedure both assumes a frozen OI snapshot and uses assignment semantics that make any intervening settle corrupt the upgrade.

RECOMMENDATION

Drain or cancel all pending market orders before scanning, and keep operator fulfillment halted throughout the pause, migration, and verification window. Enforce the pause when market orders settle, or require zero pending orders and prevent new fulfillment until reconciliation succeeds.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-39**

PENDING

Recursive entry-price flooring converts losing DCA legs into LP-funded profit

| | |
|----------|---|
| CATEGORY | Informational |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncrease():L169-L235 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:_computeNewOpenPrice():L432-L442 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallback():L324-L384 |
| REVIEW | Main Review 1 |

— DESCRIPTION

Each increase feeds the already-floored aggregate entry back into the next harmonic merge, so downward remainders compound rather than preserving the exact cost/coin basis. The revised passing PoC configures a PEPE/SHIB-magnitude exponent -10 market through the real PairStorage path, charges 0.10% maker/taker fees on every open and close, applies a nonzero 1 bp execution spread, and executes 453 separately authorized public open/increase legs. The exact fee-net legs lose 25.405146 USDC at close, while the recursively rounded aggregate reports 199.302284 USDC profit. The 224.707430 USDC rounding delta exceeds 90.147000 USDC of open/close fees, and the ordinary market close leaves the trader net-positive. Impact: An unprivileged DCA/TWAP-style trader can realize positive net PnL even though its actual fills are collectively losing, with the excess paid from AvantisTranche. The improved PoC directly answers the fee/granularity objection: it proves trader profit and tranche principal loss after conservative 0.10% open and close fees plus nonzero spread at a realistic listed-token price magnitude. The setup requires many independently fulfilled legs and a suitable price path, while OI and operational limits bound scale, supporting Medium severity.

— RECOMMENDATION

Maintain a canonical high-precision aggregate coin/cost accumulator and derive display entry price only after aggregation. Never use a previously rounded entry as the basis for the next merge. Compute close PnL from conserved notional and coin exposure with full-precision arithmetic and non-compounding rounding.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/8730149e359462678bf92eec15553d4abeeb69d4>

— RESOLUTION

Status: Pending.

MEDIUM **M-40**

RESOLVED

Zero-value deposits can move liquidatable positions outside liquidation

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L483 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:_calculateNewLeverage():L512-L529 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeLiquidationPricePure():L840-L854 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:_updateTrade():L548-L558

REVIEW Main Review 1

— DESCRIPTION

Margin-update validation follows the requested enum rather than the actual post-fee collateral delta. For a nominal DEPOSIT, accrued borrow/funding fees can exceed the deposited amount, so the callback reduces collateral and raises leverage while taking the DEPOSIT branch and skipping withdrawal-only equity checks. Capitalizing fee F changes the liquidation-distance numerator from $liqThreshold * C - F$ to $liqThreshold * (C - F)$, moving the liquidation boundary away by the non-threshold fraction of F despite no new backing. The PoC proves through the real LIQ callback that an identical control is liquidated, while DEPOSIT(0) contributes no USDC and makes the same price revert InvalidLiq.

Impact: A trader can front-run a valid keeper liquidation with a zero-value nominal deposit and keep the leveraged position open without adding margin. This denies an immediately valid liquidation and reduces the protocol's liquidation buffer, allowing additional adverse price movement before liquidation becomes valid again. The maneuver can repeat as fees accrue until the maximum-leverage check binds, increasing insolvency exposure during volatile markets.

— RECOMMENDATION

Choose validation based on the actual post-fee collateral delta. If collateral decreases, apply live-price withdrawal/equity checks and reject updates to already-liquidatable positions. Alternatively require every DEPOSIT to cover accrued debits and strictly increase collateral. Preserve accrued fees in liquidation accounting so capitalization cannot relax the liquidation boundary.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/1476425c88707fa184d9f06a448767e955613077>

— RESOLUTION

Status: Resolved.

M-40 Zero-value deposits can move liquidatable positions outside liquidation **CONTINUED**

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

MEDIUM **M-41**

ACKNOWLEDGED

Limit orders can open with crossed TP or SL and wipe trader collateral

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitOpenOrderCallback():L81-L161 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/MarketCallbackModule.sol:openTradeMarketCallback():L72-L125 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:correctTp():L228-L242 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L166-L205

REVIEW Main Review 1

— DESCRIPTION

A trader-supplied nonzero TP/SL can be valid relative to the requested standing-limit price but become wrong-side after a gapped spread-adjusted fill. Limit-open settlement validates trigger direction, exposure, and entry slippage, then registers the original target/stop without checking it against the actual fill; correctTp/correctSl can preserve it. The market-open callback already rejects the identical state. This is independent of accepted position-increase normalization findings: no increase or normalizeTpSl call occurs, and preserving zero sentinels or using final leverage does not add the missing limit-fill invariant. The revised PoC also contains no close-only transition, stale leverage, or index reuse.

Impact: A resting order can be born economically inverted and normal keeper automation can immediately close it at a loss. The revised passing fork PoC fills a valid 150x reversal long at 51,000 while preserving its 50,400 TP; the equivalent market open reverts. Guaranteed TP settlement values the close at 50,400, clamps PnL to -100%, returns zero collateral, and credits a keeper reward. Loss is bounded to affected traders' escrow but can wipe it completely.

— RECOMMENDATION

After spread is applied at LIMIT_OPEN fulfillment, enforce the same TP/SL-versus-final-entry invariants as MarketCallbackModule, leaving a crossed order unfilled or canceling/refunding it. If rebasing is desired, require explicit trader-authorized semantics.

— RESOLUTION

Status: Acknowledged.

LOW **L-01**

ACKNOWLEDGED

Market fulfill burns intent orders

CATEGORY Validation

LOCATION `src/Trading/MarketTradingModule.sol:executeMarketOrder`

REVIEW Main Review 1

DESCRIPTION

`executeMarketOrder` forwards any `orderId` to `PriceAggregator.fulfill` without checking that the stored request is a market-style order. If an operator passes a TWAP or RFQ slice `orderId`, `_executeCallback` has no matching branch and no final revert, so it returns without calling the intent callback. `fulfill` then deletes the oracle request anyway.

```
`plain text`
```

```
_executeCallback(orderId, r, price, _spreadParams);  
delete orders[orderId];  
delete ordersAnswers[orderId];  
`
```

The intended TWAP/RFQ path is `fulfillIntents`, which passes the parent intent id and dispatches to the TWAP/RFQ callbacks. A burned open slice does not automatically seize collateral, because unused TWAP escrow can be refunded by trader cancellation and unused RFQ escrow can be refunded by operator cancellation when cancellation is available. The impact is still a silent availability failure: the affected slice does not execute, the oracle request cannot be retried under the same `orderId`, RFQ users depend on operator action to recover or progress, close intents leave the position exposed until retried or canceled, and stale slice storage remains orphaned.

RECOMMENDATION

Reject unsupported order types in `_executeCallback`, or validate in `executeMarketOrder` that the stored order type is one of the callback types handled by `fulfill`. Keep TWAP and RFQ order types restricted to `fulfillIntents`.

RESOLUTION

Status: Acknowledged.

LOW **L-02**

ACKNOWLEDGED

Expired market closes lock exits

| | |
|----------|---------------------------------|
| CATEGORY | DoS |
| LOCATION | TradingStorageCoreExtension.sol |
| REVIEW | Main Review 1 |

— DESCRIPTION

Market-close requests set `beingMarketClosed = true` when the pending order is stored. The flag is cleared only when `unregisterPendingMarketOrder` successfully unregisters that pending order.

```
``plain text
_openTradesInfo[_order.trade.trader][_order.trade.pairIndex]
[_order.trade.index].beingMarketClosed = true;
...
require(_order.block > block.number - 15, "ORDER_EXPIRED");
`
```

The close callback calls `unregisterPendingMarketOrder(orderId, false)` after settlement logic. If an operator executes the close after the 15-block window, `unregisterPendingMarketOrder` reverts with `ORDER_EXPIRED`. This reverts the callback, so the close does not settle and the stored close flag remains true.

The trader cannot clear this state directly. A later `closeTradeMarket` call reads `beingMarketClosed` and reverts with `ALREADY_BEING_CLOSED`. The only cleanup path is `Trading.cancelPendingMarketOrder`, which is `onlyOperator`.

Example: a trader requests a market close, the operator execution is delayed by 16 blocks, and `executeMarketOrders` reverts with `ORDER_EXPIRED`. The position remains open, but every new market-close request from the trader reverts until an operator explicitly cancels the expired pending order.

Impact

An ordinary execution delay can leave a trader unable to close a position through the normal market-close path. During that lock, the position remains exposed to price movement, liquidation, and fees. Recovery depends on operator intervention rather than the trader.

— RECOMMENDATION

L-02 Expired market closes lock exits**CONTINUED**

Do not make successful cleanup depend on the 15-block freshness check. Expired pending orders should be cancellable by the trader or should be automatically unregistered without reverting settlement cleanup. At minimum, allow the trader to clear their own expired pending close and reset `beingMarketClosed`.

— RESOLUTION

Status: Acknowledged.

LOW **L-03**

ACKNOWLEDGED

Margin deposits leave stale max TP

| | |
|----------|--|
| CATEGORY | Logical Error |
| LOCATION | src/Trading/PositionUpdatesModule.sol:_executeUpdateMargin, src/PairStorage/PairStorageExtension.sol:correctTp |
| REVIEW | Main Review 1 |

DESCRIPTION

`updateMargin` re-corrects the current TP after a margin change by passing the already-stored `existingTrade.tp` back into `_updateTp`. `_updateTp` then calls `correctTp` using the trade's updated leverage. However, `correctTp` only recomputes the max-gain TP when the supplied TP is zero or already reaches the pair max gain at the current leverage. After a margin deposit reduces leverage, the old max-gain TP is below the new max-gain TP, so it is treated as an explicit below-max TP and left unchanged.

Illustration with `openPrice = 2000` and `maxGainP = 500`:

```
<table header-row="true">
```

```
<tr>
```

```
<td>Path</td>
```

```
<td>Leverage</td>
```

```
<td>Stored TP</td>
```

```
<td>Meaning</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Open long with tp = 0</td>
```

```
<td>25x</td>
```

```
<td>2400</td>
```

```
<td>20% move at 25x equals 500% profit</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Deposit margin, leverage drops</td>
```

```
<td>10x</td>
```

```
<td>2400</td>
```

```
<td>old TP is only 200% profit at 10x, so correctTp leaves it unchanged</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Fresh long with same open price and tp = 0</td>
```

```
<td>10x</td>
```

```
<td>3000</td>
```

L-03 Margin deposits leave stale max TP**CONTINUED**

<td>50% move at 10x equals 500% profit</td>

</tr>

</table>

The margin-updated trade and the fresh trade can end with the same open price and leverage, but the margin-updated trade is capped at `2400` while the fresh trade can use `3000`. This creates an asymmetric outcome for traders who add margin: their TP can remain stale and close the position at a lower profit target than the protocol would allow for an otherwise equivalent fresh position.

— RECOMMENDATION

Preserve max-TP intent across margin updates. Before changing leverage, detect whether the stored TP is the old leverage's auto-max or max-gain TP. If it is, recompute the TP with the new leverage after the margin update. Explicit below-max TP values should remain unchanged.

— RESOLUTION

Status: Acknowledged.

LOW **L-04**

ACKNOWLEDGED

Position roll disables liquidations

CATEGORY	Unexpected Behavior
LOCATION	LimitTradingModule.sol, PairStorage.sol
REVIEW	Main Review 1

DESCRIPTION

`executeLimitOrder`, the only entry point for liquidations (and TP/SL closes), carries the `whenMarketNotPaused` modifier:

```

`plain text
) external payable whenNotPaused onlyOperator whenMarketNotPaused(_pairIndex) {
`
`plain text
modifier whenMarketNotPaused(uint _pairIndex) {
if(pairInfos.pairsStorage().isPositionRollOngoing(_pairIndex)){
revert Revert_PositionRollOngoing();
}
_ ;
}
`

```

While a pair has `isPositionRollOngoing(_pairIndex) == 1`, every `executeLimitOrder` call for that pair reverts, so liquidations cannot run for the entire duration of the roll.

`isPositionRollOngoing` is toggled by governance via `PairStorage.setPositionRollOngoing`, and the roll itself is performed by the position-roll keeper.

During a position roll the market's price basis is being adjusted while liquidations are frozen. A position that becomes insolvent inside the roll window cannot be liquidated until the roll completes, so the protocol carries bad debt and the LPs absorb the shortfall for that period. This is not attacker-triggerable (it depends on the trusted roll keeper's operation), but it is an availability gap that couples a routine maintenance action to the safety-critical liquidation path.

RECOMMENDATION

Allow liquidations to proceed during position rolls, or bound the roll window tightly and ensure positions are re-margined or protected before a roll starts. If liquidations must be paused, document the window and its bad-debt exposure, and keep it as short as possible.

L-04 Position roll disables liquidations

CONTINUED

— RESOLUTION

Status: Acknowledged.

LOW **L-05**

RESOLVED

Signed TWAP RFQ bypass pause gates

CATEGORY	Unexpected Behavior
LOCATION	TradingRouter.sol
REVIEW	Main Review 1

— DESCRIPTION

Direct TWAP/RFQ initiation goes through the router's pause and pair-roll checks, but the signed batch initiation entrypoints only require `onlyOperator`. When the router is paused, an operator can still submit a signed TWAP/RFQ open or close intent through `executeTwapBatched` or `executeRfqBatched`. Open intents can transfer the trader's USDC into `TradingStorage` and create a TWAP/RFQ state while slice execution and cancellation remain pause-gated.

This weakens the pause control and can leave newly created signed-intent orders stuck until the protocol is unpaused. The same path also bypasses the position-roll initiation gate for signed TWAP/RFQ orders.

— RECOMMENDATION

Apply the same pause and pair-roll checks used by direct TWAP/RFQ initiation before executing signed batch initiation. For open orders, validate the target pair before escrow transfer and state creation.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-06**

ACKNOWLEDGED

Signed intents cannot settle in backup mode

CATEGORY Unexpected Behavior
LOCATION PriceAggregator.sol
REVIEW Main Review 1

DESCRIPTION

`PriceAggregator.fulfillIntents` resolves the settlement price with `_validatedPrice(orderId, r, priceUpdateData, _priceSourcing, false)`, passing `allowBackupOnly = false`. In `_validatedPrice`, the backup only branch is taken only when `allowBackupOnly && useBackupOnly`, so while governance has enabled backup only oracle mode (main oracle dysfunctional), all TWAP and RFQ open and close slices fail to settle.

This is an availability gap for the signed intent order types during the exact window backup mode exists to cover. Traders with pending TWAP/RFQ opens or closes cannot progress or exit through the intent path until main oracle service resumes, even though the equivalent limit close path is designed to keep working on the backup feed. Triggering requires governance to enable backup only mode.

RECOMMENDATION

Allow the intent settlement paths that can safely rely on the Chainlink backup feed to pass `allowBackupOnly = true`, consistent with the limit close handling, so TWAP/RFQ orders can settle while backup only mode is active.

RESOLUTION

Status: Acknowledged.

LOW **L-07**

ACKNOWLEDGED

Loss protection is inert

CATEGORY	Unexpected Behavior
LOCATION	PairStorage.sol, PairInfos.sol
REVIEW	Main Review 1

— DESCRIPTION

Loss protection cannot activate in the current code. `PairInfos.lossProtectionTier` always returns tier `0`, and `PairStorage.lossProtectionMultiplier` returns `100` whenever the selected multiplier slot is unset. There is no production setter for `PairStorage.lossProtection[pairIndex][tier]`, so protected PnL is always equal to raw PnL.

`setLossProtectionConfig` only stores skew thresholds in `PairInfos`; it does not set multipliers in `PairStorage`, and the tier calculation does not read those thresholds. As a result, the loss-protection feature is effectively dead. Losing trades do not receive a loss rebate at close, and the margin-withdrawal path also does not currently become more permissive because the multiplier remains `100`.

— RECOMMENDATION

Either remove the unused loss-protection path, or implement the full configuration flow: set multipliers with bounds, compute tiers from the skew config, and keep withdrawal solvency checks based on raw PnL.

— RESOLUTION

Status: Acknowledged.

LOW **L-08**

ACKNOWLEDGED

ERC4626 tranche first deposit inflation

CATEGORY Warning
LOCATION AvantisTranche.sol
REVIEW Main Review 1

DESCRIPTION

Tranche (AvantisTranche.sol) extends OpenZeppelin 4.9.3 ERC4626Upgradeable, does not override `_decimalsOffset()` (so offset is 0), and is not seeded on deployment. This is the setup usually flagged for the ERC4626 first depositor inflation attack, but OZ 4.9.x already applies the virtual shares/assets mitigation that the tranche inherits:

```
``solidity
shares = assets.mulDiv(totalSupply() + 10 ** _decimalsOffset(), totalAssets() + 1, rounding);
assets = shares.mulDiv(totalAssets() + 1, totalSupply() + 10 ** _decimalsOffset(), rounding);
`
```

The `+1` virtual asset and `+10**offset` virtual shares form a phantom position that captures part of any donation. Running the attack at offset 0 on an empty vault: the attacker takes 1 share for 1 wei, donates `D`, and only zeroes out a victim depositing `V` if `D > 2V`. The attacker's net on redemption is $(V - D)/2$, which is always negative, and half the victim's loss is stranded in the vault. So the profitable theft variant is not viable.

The residual is a bounded, self funded grieving action: a griever can cost a first depositor `V` only by burning more than $V/2$ of their own funds, only while total supply is near zero (fresh deployment or a freshly unified v2.0 vault), and only if the depositor ignores receiving zero shares. No protocol funds are at risk; this is defense in depth at the weakest offset setting.

RECOMMENDATION

Override `_decimalsOffset()` to a non zero value such as 6, which scales both the required donation and the attacker loss by `10**offset` and makes grieving infeasible.

RESOLUTION

Status: Acknowledged.

LOW **L-09**

ACKNOWLEDGED

Large payouts hit tranche pull cap

CATEGORY Logical Error
LOCATION `AvantiTranche.sol`
REVIEW Main Review 1

DESCRIPTION

`VaultManager._sendUSDCToTrader` covers any payout shortfall with a single `junior.withdrawAsVaultManager(difference)` call. The tranche rejects any manager withdrawal `>= 500,000e6`, while its deposit cap is governance-configurable and OI limits scale with tranche assets. After the vault grows beyond the initial cap, a legitimate large profitable close, margin withdrawal, or execution reward that needs to pull at least 500,000 USDC from the tranche reverts with `MaxWithdrawalAmountExceeded`.

This blocks settlement for otherwise valid exits and leaves the position or payout pending until liquidity is manually reshuffled or the transaction is split through a different path. The trigger is config-gated by vault size and available buffer, but the fixed pull cap is not tied to the configured tranche cap or protocol OI limits.

RECOMMENDATION

Remove the fixed per call cap for vault manager pulls, raise it with the tranche deposit cap, or split shortfall pulls into bounded chunks inside `VaultManager` so valid payouts cannot be blocked by an internal transfer limit.

RESOLUTION

Status: Acknowledged.

LOW **L-10**

RESOLVED

Clamped TWAP RFQ close overcharges fee

CATEGORY	Logical Error
LOCATION	https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/TradingCallbacks/IntentCallbackModule.sol#L332 ,
REVIEW	Main Review 1

DESCRIPTION

In `_executeCloseTwapSlice` and `_executeCloseRfqSlice`, the close notional `v.levPosToken` is derived once from the requested slice `coinSize`. When the requested slice meets or exceeds the remaining position, the code clamps `v.collateral` to `existingTrade.initialPosToken` and recomputes `v.coinSize` to the real remaining size, but it does not recompute `v.levPosToken`. The close fee in `_closeTwapSliceUnregister` is then charged as `v.levPosToken * pairCloseFeeP(...)`, using the pre clamp oversized notional.

As a result, whenever a final or oversized close slice is clamped down to the actual position, the trader pays a close fee on more notional than is actually closed. The overcharge is bounded by how much the requested slice exceeds the remainder and is operator influenced, so impact is limited to excess fees on the closing slice.

RECOMMENDATION

After clamping `v.collateral` and `v.coinSize`, recompute `v.levPosToken` from the clamped values before computing the close fee so the fee matches the notional actually closed.

RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-11**

ACKNOWLEDGED

Low price TP rounds to entry

CATEGORY Rounding
LOCATION PairStorageExtension.sol:correctTp
REVIEW Main Review 1

DESCRIPTION

`correctTp` treats `tp == 0` as a request to set the trade's TP to the pair max-gain price, but the max-gain price delta is computed with integer division. For very low-price pairs or very high leverage, `tpDiff` can round down to zero:

```
``plain text
uint256 tpDiff = ((openPrice *
uint256(pairs[pairIndex].values.maxGainP)).unscaleByLeverage(leverage)) / 100;
return buy ? openPrice + tpDiff : tpDiff <= openPrice ? openPrice - tpDiff : 0;
`
```

The condition is roughly `openPrice * maxGainP < leverageX * 100`, where `openPrice` is scaled by `1e10` and `leverageX = leverage / 1e10`. With `maxGainP = 500` and `1000x` leverage, tokens priced below about `\$0.00000002` can produce `tpDiff == 0`. A market open with `tp == 0` passes the pre-registration TP check, then registration corrects TP to exactly `openPrice`, so the stored position can have an entry-price TP instead of no TP or a meaningful max-gain TP.

RECOMMENDATION

When clamping TP, require the computed `tpDiff` to be nonzero, round it up to at least one price unit, or preserve `tp == 0` as no TP. Also validate pair configuration so supported pair prices, max gain, and max leverage cannot produce a zero TP distance.

RESOLUTION

Status: Acknowledged.

LOW L-12

ACKNOWLEDGED

Operator spread can set arbitrary prices

CATEGORY	Validation
LOCATION	PairInfosExtension.sol
REVIEW	Main Review 1

DESCRIPTION

Avantis accepts an operator supplied signed `spreadP` during order execution and applies it directly to the authenticated oracle price. The value is not bounded below and is not checked against the pair's configured spread before it becomes the execution price used for slippage, registration, and PnL.

```plain text

```
finalPrice = _marketExecutionPrice(price, _spreadParams.spreadP, _buy, _isOpen);
```

```
function _marketExecutionPrice(int256 _price, int256 _spreadP, bool _long, bool _isOpen) private pure
returns (int256) {
```

```
int256 priceDiff = (_price * _spreadP) / 100 / int256(_PRECISION);
```

```
return _isOpen
```

```
? _long ? (_price + priceDiff) : (_price - priceDiff)
```

```
: _long ? (_price - priceDiff) : (_price + priceDiff);
```

```
}
```

```
``
```

A negative spread improves every trade direction. For example, if the authenticated oracle price is 100, an operator supplied `spreadP = -90%` lets a colluding long open at 10; the favorable slippage check passes because it runs on the adjusted open price. Closing near 100 can then realize profit, bounded by the pair's maximum gain cap. This is reported as Low because the operator is a trusted role. If that role or its spread input were treated as untrusted or compromised, the same issue would be much more severe because it can manufacture arbitrary settlement prices.

## RECOMMENDATION

Reject negative `spreadP` values unless they are an intentional, separately capped rebate mechanism. Enforce an on-chain range for operator supplied spread parameters, compare them against pair configuration, and clamp both explicit spread and impact adjusted spread to the configured bounds before applying them to execution prices.

---

**L-12** Operator spread can set arbitrary prices

---

**CONTINUED****— RESOLUTION**

Status: Acknowledged.

LOW **L-13**

ACKNOWLEDGED

# USDC Blacklisted Traders Are Not Liquidatable

|          |                                                                                                                                                                                                                                                                                                                                             |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | DoS                                                                                                                                                                                                                                                                                                                                         |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/70f16f3e40d0032e075ae19ed9690be143d817ee/src/TradingCallbacks/TradeRegistry.sol#L355">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/70f16f3e40d0032e075ae19ed9690be143d817ee/src/TradingCallbacks/TradeRegistry.sol#L355</a> |
| REVIEW   | Main Review 1                                                                                                                                                                                                                                                                                                                               |

## — DESCRIPTION

Liquidation and force-close settlement can revert when the trader is blacklisted by USDC.

`LimitCallbackModule.executeLimitCloseOrderCallback` routes liquidations through `_unregisterTrade`. Inside `TradeRegistry.unregisterTrade`, the protocol first calculates `usdcSentToTrader` and, if it is nonzero, calls `vaultManager.sendUSDCToTrader(_trade.trader, usdcSentToTrader)`. `VaultManager._sendUSDCToTrader` then performs `require(storageT.usdc().transfer(_trader, _amount))`.

Therefore, if there are USDC transfers to the blacklisted addresses, tx will revert and the liquidation does not complete.

## — RECOMMENDATION

Consider to escrow the residual amount in a claimable balance that the trader can withdraw later to safely navigate this edge case.

## — RESOLUTION

Status: Acknowledged.

LOW **L-14**

ACKNOWLEDGED

## Pending opens bypass risk controls

|          |                                                   |
|----------|---------------------------------------------------|
| CATEGORY | Warning                                           |
| LOCATION | MarketCallbackModule.sol, LimitCallbackModule.sol |
| REVIEW   | Main Review 1                                     |

### — DESCRIPTION

Open-order risk checks run when the trader submits the request but are not consistently repeated when the position is registered. Consequently, orders accepted before an emergency or configuration change can still create exposure afterward:

- `executeMarketOrder` has no pause modifier, and the market-open callback does not check the global pause. A pending market open can execute while `TradingRouter` is paused.
- Limit-open fulfillment checks price and exposure, but not `closeOnlyMode`. A resting limit order can open after the pair is switched to close-only mode.
- Standard market and limit callbacks do not recheck current pair leverage bounds. An order accepted before governance lowers `pairMaxLeverage` can open above the new maximum.

Attackers can preposition resting orders before a volatile event or risk intervention, then add exposure after governance has attempted to stop or reduce it.

### — RECOMMENDATION

Immediately before registering every market, limit, TWAP, or RFQ open, recheck the global pause, pair close-only state, position-roll state, and current pair leverage limits. Cancel and refund orders that no longer satisfy current risk configuration.

### — RESOLUTION

Status: Acknowledged.

**Guardian:** Open orders can be fulfilled after governance changes a pair's risk controls. Limit opens now recheck close-only mode and leverage, and TWAP/RFQ callbacks recheck leverage, but validation remains incomplete.

Market-open callbacks do not recheck the pair's current close-only state, and the collateral-denominated market callback does not recheck current min/max leverage. TWAP and RFQ open callbacks also omit close-only validation. An order accepted before a pair becomes close-only or its leverage range is tightened can therefore add exposure under stale permissions.

**L-14** Pending opens bypass risk controls**CONTINUED****\*\*Recommendation\*\***

Immediately before registering every market, TWAP, or RFQ open or increase, recheck the pair's current close-only state and applicable min/max leverage. Cancel and refund pending orders or slices that no longer satisfy the current controls, and add tests that change each setting between initiation and callback.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-15**

ACKNOWLEDGED

# Reserve accounting omits liabilities

|          |                                                                                                                                                                                                                                   |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Logical Error                                                                                                                                                                                                                     |
| LOCATION | src/VaultManager.sol:distributePnlRewardsFraction(), src/VaultManager.sol:distributeRewardsWithThreshold(), src/VaultManager.sol:_sendUSDCToTrader(), src/VaultManager.sol:allocateGasFees(), src/VaultManager.sol:claimGasFees() |
| REVIEW   | Main Review 1                                                                                                                                                                                                                     |

## DESCRIPTION

`VaultManager` tracks several liabilities against one pooled USDC balance, but each consumer subtracts a different subset of them before spending. Two omissions follow.

**\*\*1. Reward distribution ignores `fundingOwed`.**

Reward distribution can move funds into the junior tranche without preserving `fundingOwed`, even though trader payouts and governance buffer withdrawals both treat it as reserved:

``plain text

```
int256 currentBalance = int256(IERC20(junior.asset()).balanceOf(address(this)));
```

```
if (currentBalance > int256(_thresholdAmount)) {
```

```
...
```

```
if (pnlRewardsToDistribute > 0) {
```

```
 distributePnlRewardsFraction(pnlRewardsToDistribute * 100 / pnlRewards);
```

```
}
```

```
if (totalRewardsToDistribute > 0) {
```

```
 totalRewards -= totalRewardsToDistribute;
```

```
 _distributeRewards(address(junior), totalJuniorRewards);
```

```
}
```

```
}
```

```
``
```

The PnL leg routes through `distributePnlRewardsFraction()`, which only subtracts `totalRewards` before moving `pnlRewards`. The `totalRewards` leg transfers after checking only the raw balance threshold. Neither ensures the remaining balance still covers funding owed.

If `VaultManager` holds 100 USDC, `totalRewards` is 80 USDC, and `fundingOwed` is 50 USDC, the threshold function can distribute 80 USDC and leave 20 USDC against 50 USDC of funding liabilities.

**\*\*2. Trader payouts ignore `gasFeesCollected`.**

The mirror-image omission exists in the payout path. `_sendUSDCToTrader()` reserves two buckets and forgets the third:

``plain text

```
int256 fundingOwed = storageT.pairInfos().fundingOwed();
```

**L-15** Reserve accounting omits liabilities

CONTINUED

```
int256 balanceAvailable = int256(storageT.usdc().balanceOf(address(this))) - int256(totalRewards) -
fundingOwed;
```

```
``
Keeper gas fees accrue into a separate gasFeesCollected bucket via allocateGasFees(), but are
never excluded from payout-available balance. Trader payouts therefore spend USDC backing
accrued gas fees, and the shortfall is pulled from the junior tranche through
withdrawAsVaultManager(), so LP assets end up funding keeper compensation.
claimGasFees() then validates only the accounting bucket, never actual liquidity:
`plain text
if (gasFeesCollected < _amount) revert Errors.InsufficientGasFeesCollected();
IERC20(junior.asset()).safeTransfer(gov, _amount);
``
```

So a claim can pass its own check and still fail on transfer, or succeed only by having already drained the tranche.

**— RECOMMENDATION**

Introduce a single helper that computes spendable balance as the raw USDC balance minus every tracked liability ( `totalRewards`, `fundingOwed`, and `gasFeesCollected` ), and route reward distribution, trader payouts, buffer withdrawals, and gas-fee claims through it.

Specifically: subtract `fundingOwed()` before any reward distribution and cap or reject the combined `pnlRewards` and `totalRewards` transfer so the post-distribution balance still covers it; exclude `gasFeesCollected` from `_sendUSDCToTrader()`'s `balanceAvailable`; and check actual USDC availability, not just the bucket, in `claimGasFees()`.

**— RESOLUTION**

Status: Acknowledged.

LOW **L-16**

ACKNOWLEDGED

# Validator selection is not signed

|          |                                         |
|----------|-----------------------------------------|
| CATEGORY | Signatures                              |
| LOCATION | src/EIP7702Template/Eip7702Template.sol |
| REVIEW   | Main Review 1                           |

## DESCRIPTION

Validator signatures are encoded as an unsigned 20 byte validator address followed by the inner signature. ERC-4337 excludes `userOp.signature` from `userOpHash`, while the direct execution digest commits only to the mode, calls, and nonce.

```
``plain text
(IValidator validator, bytes calldata innerSignature) =
_decodeValidator(userOp.signature);
return validator.validate(calls, msg.sender, userOpHash, innerSignature) ? 0 : 1;
``
```

A bundler or relayer can replace only the validator prefix without invalidating the inner signature. If two enabled validators accept the same signer but enforce different target or spending policies, a payload intended for the restrictive validator can be redirected to the permissive validator and execute actions the signer did not authorize under the selected policy. The same issue affects direct execution and ERC-1271 validation.

## RECOMMENDATION

Bind the selected validator address into the data authenticated by the validator. Use a validator specific wrapped digest for ERC-4337 and ERC-1271, and include the validator address in the direct execution typed data.

## RESOLUTION

Status: Acknowledged.

LOW **L-17**

ACKNOWLEDGED

## Post execution context can be overwritten

CATEGORY Validation  
LOCATION src/EIP7702Template/Eip7702Template.sol:validateUserOp  
REVIEW Main Review 1

### — DESCRIPTION

`IValidator` expects validators to preserve context between `validate` and `postExecute`, but `postExecute()` receives no digest or operation identifier. `EntryPoint` does not decide whether the account signature is valid itself. It first loops over the bundle and calls each account's `validateUserOp()`, which invokes the selected validator, then begins a separate execution loop. Two operations from the same account using the same validator can therefore overwrite operation A's context while validating operation B.

```
```plain text
```

```
function postExecute() external;
```

```
// Called later during execution without userOpHash.
```

```
validator.postExecute();
```

```
``
```

For a bundle containing operations A and B:

```
```vb.net
```

```
validation: validateUserOp(A) -> validator.validate(A) -> context[account] = A
```

```
validation: validateUserOp(B) -> validator.validate(B) -> context[account] = B
```

```
// A is overwritten
```

```
execution: executeUserOp(A) -> validator.postExecute() consumes B, then clears it
```

```
execution: executeUserOp(B) -> validator.postExecute() receives empty context
```

```
```
```

Execution A then consumes B's context and clears it, while execution B receives empty context. Direct nested execution can produce the same ambiguity. Validators that enforce spending limits, balance changes, or other post execution conditions can consequently apply checks to the wrong operation or skip them entirely.

— RECOMMENDATION

Pass the operation digest or another unique context identifier to `postExecute`, and require validators to key context by account and operation. Apply the interface and call site change to both delegation implementations.

L-17 Post execution context can be overwrittenCONTINUED

— RESOLUTION

Status: Acknowledged.

LOW **L-18**

ACKNOWLEDGED

Revoked validator executes in same bundle

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/baseMainnet/src/EIP7702Template/Eip7702Template.sol#L443>

REVIEW Main Review 1

— DESCRIPTION

`validateUserOp` checks that a validator is enabled and caches it for execution. `EntryPoint v0.8` validates every operation in a bundle before executing any of them. An owner operation with nonce `n` can therefore remove a compromised validator, while a validator signed operation with nonce `n + 1` still passes validation earlier in the same bundle.

```
```plain text
```

```
if (!_getStorage().validatorEnabled[validator]) {
 revert InvalidValidator();
}
_storeValidator(userOpHash, validator);

// executeUserOp
_executeCalls(calls);
IValidator validator = _loadValidator(userOpHash);
`
```

The `removal` executes first, but `executeUserOp`` performs the authorized calls without rechecking the validator. The compromised validator can consequently transfer assets after its revocation has executed. Exploitation requires a previously enabled validator and same bundle ordering, but it defeats account recovery and can cause direct asset loss.

### — RECOMMENDATION

Store the validator's authorization epoch during validation and require the current epoch to match before `_executeCalls`. Increment the epoch whenever validator authority changes. At minimum, recheck `validatorEnabled` immediately before execution.

### — RESOLUTION

Status: Acknowledged.

LOW **L-19**

ACKNOWLEDGED

## Delegate can execute any trader action

CATEGORY	Improper access control
LOCATION	src/Trading/DelegateModule.sol:delegatedAction
REVIEW	Main Review 1

### — DESCRIPTION

`DelegateModule.delegatedAction` accepts arbitrary `call_data` from any approved delegate and executes it through `address(this).delegatecall(call_data)` after setting `senderOverride = trader`, so every downstream `__msgSender()` resolves to the trader. There is no restriction on which selectors a delegate may invoke.

An approved delegate can therefore reach any trader facing action, including `updateMargin` with a `WITHDRAW` type and unbounded opens or increases against the trader's USDC allowance. This is broader than the intended one click trading scope, which is documented to exclude withdrawals.

Because withdrawn margin is sent to the trader through `sendUSDCToTrader`, a malicious delegate cannot directly steal funds. It can still grief the trader by forcing margin withdrawals that raise leverage and liquidation risk, or by consuming the trader's allowance with unwanted positions. Exploitation requires a trader authorized delegate to act maliciously, which bounds likelihood.

### — RECOMMENDATION

Restrict `delegatedAction` to an explicit allowlist of trading selectors reflecting the intended delegate scope, and exclude margin withdrawal and any other fund moving or account configuration actions.

### — RESOLUTION

Status: Acknowledged.

LOW **L-20**

RESOLVED

# Nested actions bypass builder fees

CATEGORY	Logical Error
LOCATION	src/EIP7702Template/Eip7702Template.sol:_executeCalls, src/Trading/DelegateModule.sol:delegatedAction
REVIEW	Main Review 1

## DESCRIPTION

The EIP-7702 fee scanner unwraps at most one `delegatedAction()` call. It charges a fee only when that immediate inner selector is fee eligible.

```
``plain text
if (selector == DELEGATED_ACTION_SELECTOR) {
 bytes4 innerSelector;
 assembly {
 let bytes0ff := calldataload(add(cd.offset, 36))
 innerSelector := calldataload(add(add(cd.offset, 36), bytes0ff))
 }
 if (_isFeeEligible(innerSelector)) {
 _chargeBuilderFees(selector, collateral, trader);
 }
}
```

`DelegateModule.delegatedAction()` permits another `delegatedAction()` as its inner call, and because the inner dispatch is a `delegatecall` the original `msg.sender` is preserved, so the delegate approval check passes again at the second level. A trader's approved delegate can therefore wrap an eligible trading action twice. `_isFeeEligible` returns false for the delegated action selector itself, so the scanner sees the second wrapper, skips the fee, and the recursive delegated execution still reaches the terminal trading action.

A proof of concept opens a real position through two wrappers while the configured builder collector receives no fee. Impact is limited to builder revenue; protocol and trader accounting remain intact.

## RECOMMENDATION

Recursively unwrap delegated actions until reaching the terminal selector, with a strict depth bound, and compute the fee from that terminal call. Alternatively, reject nested `delegatedAction()` calls.

**L-20** Nested actions bypass builder fees

CONTINUED

**— RESOLUTION**

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW L-21

ACKNOWLEDGED

## Zero target bypasses validator policies

CATEGORY Validation

LOCATION src/EIP7702Template/Eip7702Template.sol:validateUserOp, \_execute, \_executeCalls

REVIEW Main Review 1

### — DESCRIPTION

The account passes each call's raw target to the selected validator, but `_executeCalls` later replaces `address(0)` with `address(this)`. A restricted validator can therefore approve a call because its target is zero even though execution turns it into an account self call.

```
```plain text
```

```
validator.validate(calls, msg.sender, userOpHash, innerSignature);
```

```
address to = call.to == address(0) ? address(this) : call.to;
```

```
(bool success, bytes memory data) = to.call{value: call.value}(call.data);
```

```
``|
```

For example, a session validator that rejects calls targeting the account will accept a zero target containing `addValidator(attackerValidator)`. Execution aliases the target to the account, satisfies `onlyThis`, and installs the attacker's unrestricted validator. A restricted session key can consequently obtain complete account authority. Both ERC-4337 and direct validator execution are affected, and `Original7702.sol` contains the same aliasing.

Severity is Medium because exploitation requires an enabled validator whose policy is target based and which does not itself normalize zero targets. The repository ships only the `IValidator`` interface, with every implementation living in tests, so no such validator is deployed today. The mismatch remains in the account template that third party and future validators will be written against, and any target restricted validator added later inherits the bypass silently.

— RECOMMENDATION

Normalize zero targets to the account before passing calls to a validator, so validation and execution use identical destinations. Alternatively, reject zero targets for validator authorized executions. Apply the change to both delegation implementations.

L-21 Zero target bypasses validator policiesCONTINUED

— RESOLUTION

Status: Acknowledged.

LOW L-22

ACKNOWLEDGED

Dust closes avoid sweep taker fee

CATEGORY	Logical Error
LOCATION	src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallbackWithCoinExposure, src/PairStorage/PairStorageExtension.sol:pairCloseFeeP, src/PairStorage/PairStorageExtension.sol:_makerOrTakerFee
REVIEW	Main Review 1

DESCRIPTION

Coin-exposure market closes select the fee tier before dust remainders are rounded up into a full close.

`pairCloseFeeP` charges the taker fee when a close fully sweeps a pair side to zero open interest, and the maker fee otherwise:

```
``plain text
uint256 closeFee = values.levPosToken *
aggregator.pairsStorage().pairCloseFeeP(existingTrade.pairIndex, pendingOrder.coinExposure,
existingTrade.buy) /
100 /
_PRECISION;
...
if (existingTrade.initialPosToken - closeCollateral < PositionMath.COLLATERAL_DIFF_THRESHOLD) {
closeCollateral = existingTrade.initialPosToken;
}
`
```

The fee tier is chosen from `pendingOrder.coinExposure`, which still leaves nominal dust on the book, so the cheaper maker fee applies. The dust clamp then runs afterwards and closes the full remaining collateral without recomputing the close coin size or the fee.

A trader who is the sole position on a pair side can therefore quote a close that leaves only dust, pay the maker fee on almost the entire notional, and still have the position fully swept. `COLLATERAL_DIFF_THRESHOLD` is $1e4$, or one hundredth of a USDC at $1e6$ precision, so the dust required is negligible relative to the fee saved.

Severity is limited to Low because it requires the closing position to be the sole contributor to that pair side, which bounds how often the setup is available.

RECOMMENDATION

Perform the dust canonicalisation before calling `pairCloseFeeP`, so that fee-tier selection sees the same size that settlement will actually close. Use the final clamped coin size and collateral consistently for the fee tier, the fee amount, and settlement.

L-22 Dust closes avoid sweep taker feeCONTINUED

— RESOLUTION

Status: Acknowledged.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-23**

ACKNOWLEDGED

Backup mode consumes close orders

CATEGORY	Logical Error
LOCATION	src/PriceAggregator.sol:_backupOnlyPrice, src/PriceAggregator.sol:_executeCallback, src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallback, src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallbackWithCoinExposure, src/TradingCallbacks/LimitCallbackModule.sol:executePartialTpSlCallback
REVIEW	Main Review 1

DESCRIPTION

In backup-only mode `_backupOnlyPrice` resolves a real backup price only for liquidations, and returns zero for every other order type:

```
``plain text
if ((r.orderType == OrderType.LIMIT_CLOSE || r.orderType == OrderType.LIMIT_CLOSE_PNL) &&
    backupFeed.feedId != address(0)) {
    if (pendingOrder.orderType == ITradingStorage.LimitOrder.LIQ) {
        price = _validBackupPrice(orderId, r.pairIndex, backupFeed.feedId);
    }
}
```

Single-order fulfillment nevertheless dispatches that zero price into the market close and partial TP/SL callbacks. Market-close callbacks interpret `price == 0`` as a cancellation and clear the pending close. Partial TP/SL callbacks skip settlement but still unregister both the trigger and the pending order.

The consequence is that a user's exit is silently consumed without executing, during the exact emergency mode that backup feeds exist to keep settlement available. The trader is not told the close failed for lack of a price; the order simply disappears and the position stays open and exposed. Meanwhile liquidations continue to work, so the asymmetry is strictly against the trader. Recovery requires resubmitting once the primary feed returns, but position-level TP/SL triggers must also be reinstalled.

RECOMMENDATION

Do not dispatch zero prices to non-liquidation callbacks. Either resolve backup prices for the affected close paths so they can settle, or leave the pending order and its trigger intact and revert with an explicit error while backup-only mode cannot price the order.

RESOLUTION

Status: Acknowledged.

LOW **L-24**

RESOLVED

Bad leverage bounds lock increases

CATEGORY	Validation
LOCATION	src/Trading/PositionUpdatesModule.sol:_increasePositionSizeWithCoinExposure, src/TradingCallbacks/MarketCallbackModule.sol:increasePositionSizeCallbackWithCoinExposure
REVIEW	Main Review 1

— DESCRIPTION

Coin-exposure position increases accept `_minLeverage` and `_maxLeverage` without ever requiring `minLeverage <= maxLeverage`. The request path validates the trade, escrows collateral, and stores the pending order, and only the callback evaluates the interval:

```
`plain text
storageT.transferUSDC(_trader, address(storageT), _updateInfo.initialPosToken);
...
storageT.storePendingMarketOrderWithCoinExposure(..., _coinExposure, _maxLeverage,
_minLeverage);
...
finalTrade.leverage >= pendingOrder.minLeverage &&
finalTrade.leverage <= pendingOrder.maxLeverage
`
```

In `_increasePositionSizeWithCoinExposure` the checks cover an existing position, ownership, pending-order count, minimum position size, close-only mode, and slippage, but not the ordering of the two leverage bounds. Collateral is transferred to storage before the pending order is stored, so an inverted interval escrows funds against an order whose settlement condition is unsatisfiable by construction.

A mistaken trader, or an approved delegate acting maliciously on the trader's behalf, can create an increase that can never settle. The collateral and a pending-order slot stay consumed until an operator or governance cleanup path force-unregisters the order. Impact is limited because the funds are recoverable through the existing force-unregister path, so this is availability and operational overhead rather than loss.

— RECOMMENDATION

Reject unordered leverage bounds at request time. Validate `_minLeverage <= _maxLeverage`, and that both fall within the pair's configured leverage limits, before transferring collateral or storing the pending order.

L-24 Bad leverage bounds lock increases

CONTINUED

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-25**

ACKNOWLEDGED

Delegations compose across cancels

CATEGORY Access Control

LOCATION `src/Trading/DelegateModule.sol:delegatedAction, src/Trading/TwapTradingModule.sol:cancelTwap, src/Trading/TwapTradingModule.sol:isDelegate`

REVIEW Main Review 1

DESCRIPTION

`delegatedAction` sets the mutable global `senderOverride` to the principal and then `delegatecall`s arbitrary calldata against the same contract. Downstream authorization reads that global rather than the true external caller:

```
``plain text
senderOverride = trader;
(bool success, bytes memory result) = address(this).delegatecall(call_data);
...
if (!isDelegate(state.trader, __msgSender())) revert Errors.NotAuthorized();
`
```

Because the delegated call target is unrestricted, delegation composes transitively. If trader B has delegated trader A, and trader A has delegated caller D, then D can invoke `delegatedAction(A, cancelTwap(B_twap))`. Inside the nested call `__msgSender()` resolves to A, `isDelegate(B, A)` returns true, and the cancellation is authorized, even though B never granted D any authority.

The refund from a cancelled open TWAP still goes to the TWAP owner, so this is disruption rather than theft: an indirectly related delegate can repeatedly terminate another trader's scheduled TWAPs at chosen moments, halting a staged entry midway or cancelling a staged exit while the remaining position stays exposed. This is distinct from the legacy-delegate issue, in that here the authority is live rather than unrevokable, but the trader has no way to see or bound the transitive chain.

RECOMMENDATION

Do not use a mutable global such as `senderOverride` as the identity for nested authorization decisions. Either forbid nested delegated calls, for example by rejecting reentrant `delegatedAction` invocations, or thread the original external caller and the principal explicitly into downstream authorization so that delegation cannot chain.

L-25 Delegations compose across cancels

CONTINUED

— **RESOLUTION**

Status: Acknowledged.

LOW L-26

ACKNOWLEDGED

Pooled ETH funds underpaid Pyth updates

CATEGORY Oracle

LOCATION `src/PriceAggregator.sol:_pythCorePrice`, `src/PriceAggregator.sol:fulfill`, `src/PriceAggregator.sol:fulfillIntents`

REVIEW Main Review 1

— DESCRIPTION

`PriceAggregator._pythCorePrice` forwards `pyth.getUpdateFee(priceUpdateData)` out of the aggregator's own balance, but never checks that the current fulfillment actually supplied that ETH:

```
``plain text
```

```
pyth.updatePriceFeeds{value: pyth.getUpdateFee(priceUpdateData)}(priceUpdateData);  
priceData = pyth.getPriceNoOlderThan(r.job, 30);  
`
```

There is no `msg.value` accounting on this path. Any native token retained by `PriceAggregator`, whether from earlier overpayments, ignored fulfillments, or reverted callbacks, is therefore available to pay for a later trader's `PYTH_CORE` update.

A trader holding a valid pending order can submit fresh, well-formed Pyth data with too little `msg.value` and consume the pooled balance instead of funding the oracle fee themselves. Every price check still passes because the oracle data itself is valid; only the fee funding is wrong. The cost is socialised onto whichever party's ETH is sitting in the contract, and the pool is drained until it is empty, at which point unrelated fulfillments begin reverting for insufficient balance.

— RECOMMENDATION

Require `msg.value >= pyth.getUpdateFee(priceUpdateData)` for each Pyth Core fulfillment and refund any excess to the payer after the oracle call. A request should never be able to spend ETH left behind by earlier requests.

— RESOLUTION

Status: Acknowledged.

LOW L-27

ACKNOWLEDGED

Forced close corrupts pair funding

CATEGORY	Oracle
LOCATION	src/PriceAggregator.sol:forceCloseTradeMarket,_forceClose; src/TradingCallbacks/MarketCallbackModule.sol:closeTradeMarketCallback; src/TradingCallbacks/TradeRegistry.sol:unregisterTrade; src/PairInfos/PairInfos.sol:getTradeValue,_storeAccFundingFees,getPendingAccFundingFees
REVIEW	Main Review 1

DESCRIPTION

`PriceAggregator.forceCloseTradeMarket` is restricted to the governance-selected `riskManager`, but its caller-supplied `_price` is not checked against oracle data, freshness or deviation bounds. `_forceClose` forwards any usable nonzero value directly to `closeTradeMarketCallback`.

The callback uses a spread-adjusted version for the target trade's PnL, but also passes the raw value through `TradeRegistry.unregisterTrade` as `ctx.currentPrice`. `PairInfos.getTradeValue` then calls `_storeAccFundingFees(pairIndex, currentPrice)` before removing the position from OI.

Funding and hedging accrued since `lastUpdateBlock` are linear in `currentPrice`. The force-close price therefore determines the pair-wide funding increment for the elapsed interval, not only the selected trade's settlement. The resulting long/short accumulators, update block and unvalidated `lastUpdatePrice` are persisted. With unequal OI, a price ten times the honest market price records approximately ten times the correct funding: the paying side inherits excessive debt and the receiving side excessive credits. A near-zero value instead removes almost all expected accrual. Later valid updates cannot reconstruct the corrupted historical interval.

The intended operating model materially limits impact. A dedicated funding keeper calls `updateFundingAccumulators` with oracle-validated prices approximately every five minutes. A force close should therefore normally corrupt only the short interval since the previous keeper update, rather than funding over the position's complete lifetime. The next keeper update starts a correctly priced interval, although it does not repair the stored delta. The five-minute cadence is operational rather than enforced on-chain: the contract authenticates the keeper and validates its prices but imposes no update interval. A compromised risk manager could still overcharge unrelated traders or create excessive credits, but exploitation requires control of a governance-appointed privileged signer and the affected interval is normally bounded by frequent keeper updates. The finding is therefore rated Low.

L-27 Forced close corrupts pair funding

CONTINUED**— RECOMMENDATION**

Do not use a caller-selected emergency-close value as a funding-accounting price. Obtain a fresh pair-bound price through the same `validatePrice` path used by ordinary execution and pass the validated value consistently to PnL settlement and funding accounting. If force close must support a fallback when the normal oracle is unavailable, require a quorum-signed or timelocked price with explicit pair binding, freshness, and deviation bounds. As defense in depth, make funding-accumulator updates consume a validated-price context rather than a plain integer so no callback or maintenance path can persist unauthenticated prices. Add regression tests showing that abnormal force-close inputs cannot change unrelated positions' funding fees or the stored pair funding price.

— RESOLUTION

Status: Acknowledged.

LOW **L-28**

ACKNOWLEDGED

RFQ close slices uncapped against authorized

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/RfqTradingModule.sol:executeCloseRfqSlice():L268-L295 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:closeRfqCallback():L479-L524 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:_executeCloseRfqSlice():L526-L600 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:_closeTwapSliceUnregister():L326-L345 packages/avantis-contracts-team2-1783453751962/src/TradingCallback/TradeRegistry.sol:unregisterTrade():L310-L375
REVIEW	Main Review 1

— DESCRIPTION

RFQ close slices are not capped to the remaining authorized or live coin size. Settlement derives leveraged notional from the arbitrary submitted amount, then clamps collateral and coin size to a full close without recomputing that notional. Close fees and referral rebates are consequently based on operator-unbounded phantom exposure.

Impact: A colluding operator, trader, and referrer can make the phantom rebate exceed all sacrificed position collateral and withdraw a net profit from pooled VaultManager liquidity. The fork PoC proves controlled balances rise by exactly the pool's loss versus a legitimate full close.

— RECOMMENDATION

Cap slices to the remaining lifecycle authorization and live exposure. After any clamp or dust promotion, recompute leveraged notional and every fee-dependent value from the final settled amount, including in the shared TWAP close path.

— RESOLUTION

Status: Acknowledged.

LOW L-29

ACKNOWLEDGED

Collateral TWAP slices retain stale leverage after a live risk-cap reduction

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/TwapTradingModule.sol:_validateTwapOpenOrder():L372-L400 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:openTwapCallback():L123-L170
REVIEW	Main Review 1

DESCRIPTION

Collateral-denominated open-TWAP lifecycles validate leverage only at creation. When an honest operator later submits a slice using the lifecycle-authorized leverage, the fresh-position callback registers it without comparing it to the pair's current leverage ceiling. A lifecycle admitted at 150x can therefore create new 150x exposure after the risk role has reduced the live cap to 5x, even though an identical new lifecycle is rejected.

Impact: Dormant collateral TWAP lifecycles can defeat a later leverage reduction and create thinner-collateralized liabilities after the risk response. The end-to-end PoC stages paired 150x lifecycles, reduces the cap to 5x, settles both through the normal operator/oracle/callback graph, and closes after a 1% authentic price move. The asymmetric +150%/-100% PnL bounds leave the trader with net profit and reduce aggregate protocol USDC by the same amount. Current OI and liquidity caps bound aggregate extraction.

RECOMMENDATION

Before each fresh collateral TWAP registration, validate the slice leverage against the current pair minimum and maximum (respecting only the applicable current override), and provide safe cancellation/refund for lifecycles invalidated by risk changes. Recheck close-only and other live risk controls at slice execution as defense in depth.

RESOLUTION

Status: Acknowledged.

LOW **L-30**

RESOLVED

Signed partial TP/SL closes stale-basis positions during active rolls

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/LimitTradingModule.sol:executePartialTpSl():L259-L303 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executePartialTpSlCallback():L327-L404 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:unregisterTrade():L240-L266
REVIEW	Main Review 1

— DESCRIPTION

executePartialTpSl checks only the global pause and can be newly authorized after a pair roll begins. Its callback does not reject the active roll, so a full-coin request can close an unprocessed position against the replacement feed, delete it, and mutate OI, reserves, and pairTraders before positionRoll rebases its obsolete entry.

Impact: A trader can convert a feed/instrument basis cutover into artificial leveraged PnL paid from protocol liquidity. The PoC pays the stale position over five times collateral while an identically opened, correctly rolled control closes near flat, and it mutates the supposedly frozen migration set during the roll.

— RECOMMENDATION

Enforce the pair-specific roll guard at signed partial execution and callback settlement. Bind signed closes to a per-pair roll epoch and make roll processing robust to deleted or invalidated entries.

— RESOLUTION

Status: Resolved.

<https://github.com/Avantis-Labs/avantis-contracts/tree/audit/guardianFixes>

LOW **L-31**

RESOLVED

Unchecked fixed-collateral TWAP leverage can extract LP principal

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/TwapTradingModule.sol:_validateTwapOpenOrder():L372-L400 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/IntentCallbackModule.sol:openTwapCallback():L126-L152
REVIEW	Main Review 1

— DESCRIPTION

TWAP admission bounds the trader's defaultLeverage and maxLeverage, but fixed-collateral fresh-slice settlement copies the operator-supplied slice leverage into the trade and registers it without checking either the lifecycle maximum or pair leverage bounds. The passing PoC shows a 10x-authorized TWAP opening at 500x despite a 150x pair maximum, while the ordinary market route rejects the same leverage. The resulting trade creates 50 times the authorized OI and reserve; after an authentic 1% price move, normal close settlement commits a 600 USDC payout, increasing the trader's balance by 500 USDC and reducing tranche assets by exactly 500 USDC.

Impact: A colluding trader and compromised operator can bypass pair collateralization policy and extract LP principal after small genuine price movements, limited only by separate exposure caps. A malicious operator can also violate an honest trader's explicit maximum and impose unauthorized liquidation sensitivity. This proves third-party LP fund loss rather than only the trader escrow stranding and bookkeeping denial demonstrated by the prior same-root-cause finding.

— RECOMMENDATION

For every fixed-collateral slice, require the supplied leverage to be at least the current pair minimum and no greater than the lifecycle maximum, which is itself capped by the pair maximum. Apply these checks to both fresh and increase branches and explicitly reject zero leverage. If TWAP must necessarily average over time, require at least two slices as a separate policy hardening.

— RESOLUTION

Status: Resolved.

LOW L-32

ACKNOWLEDGED

Position rolls bypass OI caps and can freeze pair-wide settlement

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:positionRoll():L401-L460 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:withinExposureLimits():L859-L870 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L696
REVIEW	Main Review 1

DESCRIPTION

positionRoll() can increase a trade's USD OI when the instrument is redenominated upward, but it updates trade, aggregate, pair/group, and wallet OI and reserves the delta without applying withinExposureLimits(). Thus an otherwise honest scheduled roll can push live OI above the global, pair, side, group, or wallet caps. Once a sufficiently skewed pair exceeds its pair cap, rollover multiplier math subtracts an over-100% utilization/skew product from its fixed denominator and panics.

Impact: An adversary can prefill capacity before a scheduled roll so legitimate keeper rolls create an over-cap state. The resulting rollover underflow blocks remaining rolls and borrow-fee-dependent closes, liquidations, margin changes, and increases across the pair, retaining unwanted or underwater positions and locking reserves until privileged configuration/accounting intervention. The PoC uses real market opens and keeper roll/callback paths, explicitly proves the ordinary open is cap-rejected, the roll crosses both caps while remaining physically reservable, and market close and liquidation revert atomically.

RECOMMENDATION

Before any OI-increasing roll mutation, validate the delta with withinExposureLimits() or an equivalent complete global/pair/side/group/wallet check. Also make rollover utilization math saturating or otherwise safe at and above 100% so legacy or misconfigured exposure states cannot prevent risk-reducing settlement.

RESOLUTION

Status: Acknowledged.

Guardian: The new rollover-multiplier saturation prevents over-cap utilization from underflowing in the fee calculation and freezing pair settlement, but it only handles the consequence of exceeding a cap.

positionRoll can still increase a trade's USD OI when the new open price is higher, then writes the positive delta through _updateOpenInterestUSDC, updateGroupOI, and reserveBalance without calling withinExposureLimits. A roll can

L-32 Position rolls bypass OI caps and can freeze pair-wide settlement**CONTINUED**

therefore exceed the pair-total, pair-side, group, protocol-wide, and per-wallet caps even though an ordinary position increase with the same delta would revert. The keeper restriction limits who can trigger the path but does not preserve the exposure invariant.

****Recommendation****

Before allocating fees or changing any trade or OI state, consider to calculate any positive roll delta (`newTradeOi - oldTradeOi`) and require `withinExposureLimits(trader, pairIndex, delta, buy)`, or equivalent checks covering the protocol, pair total and side, group, and wallet caps. Then update the trade, every OI aggregate, group OI, and vault reserves atomically.

LOW L-33

ACKNOWLEDGED

Persistent phantom PnL rewards let LPs extract future traders' collateral

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:_sendUSDCToTrader():L488-L501 packages/avantis-contracts-team2-1783453751962/src/VaultManager.sol:distributePnlRewardsFraction():L539-L548 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:unregisterTrade():L332-L370 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:withdrawAsVaultManager():L144-L147
REVIEW	Main Review 1

DESCRIPTION

VaultManager excludes totalRewards but not live-position principal or pnlRewards backing when computing payout liquidity. An early profitable close can consume another live position's principal; a later losing close replenishes the manager and increments pnlRewards. When the displaced position closes at break-even, its principal repayment spends those reward-backing tokens while TradeRegistry decrements pnlRewards by zero. The resulting false counter persists after the originating trades are gone. A later trader's collateral makes it distributable, allowing a transient LP to deposit before the keeper transfer and redeem afterward. The passing fork PoC reconciles the LP's exact post-fee gain to the loss of combined manager/tranche assets and proves the selected trader's close requires a tranche pull at or above the hard per-call ceiling. Example:

1. **A** and **B** each open 1000 → balance 2000.
2. **B** closes +500 → pay 1500 from the shared buffer → balance 500. `pnlRewards` stays 0. A's claim is still 1000 (500 hole vs LPs/tranche).
3. **C** opens 1000 → balance 1500, then closes -500 → pay C 500, `pnlRewards += 500`, balance 1000.
4. That 1000 is economically **A's remaining principal**, but the counter also claims 500 of it as PnL rewards.
5. **A** closes break-even → `balanceAvailable` allows paying 1000 (ignores `pnlRewards`) → balance 0, `pnlRewards` still 500.
6. Later opens refill the manager with new collateral. Keeper `distributePnlRewardsFraction` only checks `balance - totalRewards`, so it can ship that phantom 500 into the tranche.

Profits drawing other traders' buffer first is normal pooling. The bug is **double-claiming** the same dollars as (a) remaining principal and (b) `pnlRewards`, then leaving the counter up after principal is repaid.

Impact: A persistent false reward counter can redirect unrelated future traders' collateral into the tranche as pseudo-yield. An unprivileged transient LP can extract a conserved amount of protocol/victim USDC, and the selected trader can become unable to settle when

L-33 Persistent phantom PnL rewards let LPs extract future traders' collateral **CONTINUED**

the resulting tranche shortfall reaches the 500,000-USDC per-call limit. Repeated normal settlement cycles can compound the unbacked counter and insolvency.

— RECOMMENDATION

Track live-position principal and each reward liability separately. Exclude pnlRewards backing and position collateral from payout liquidity, debit the matching PnL pool atomically with profitable payouts, and permit distribution only when pnlRewards is backed by genuinely unencumbered USDC. Use cooldowns only as defense in depth.

— RESOLUTION

Status: Acknowledged.

LOW **L-34**

ACKNOWLEDGED

Reentrant consecutive nonces bypass validator lifetime limits and drain the account

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/EIP7702Template/Eip7702Template.sol:_execute():L268-L319 packages/avantis-contracts-team2-1783453751962/src/EIP7702Template/Eip7702Template.sol:_executeCalls():L443-L482 packages/avantis-contracts-team2-1783453751962/src/EIP7702Template/Eip7702Template.sol:_getAndUseNonce():L597-L600

REVIEW Main Review 1

DESCRIPTION

OP_DATA consumes a keyed nonce before validator validation and then forwards value and all remaining gas to arbitrary targets without an account execution lock. A contract payee invoked by signed nonce n can therefore reenter execute with signed nonce $n+1$, nesting multiple validate/call/postExecute lifecycles. An account-scoped transient reservation following IValidator's documented pattern is overwritten by each nested validation and cleared by the innermost postExecute, so all validations observe stale persistent lifetime usage. The passing PoC's validator permits only one-ether native payments to one fixed payee under a one-ether lifetime cap; the payee's receive hook recursively executes five individually signed consecutive nonces, draining five ether while only one ether is recorded. The serial control rejects nonce one after nonce zero completes.

Impact: A restricted session credential can spend its lifetime allowance once per recursion level and drain the delegated account's entire native or token balance without the owner key. The validator sees only individually in-policy payments to its configured payee; reentrancy occurs implicitly in that payee's callback. Prepared consecutive nonces can scale the loss to the full account balance while validator accounting records only one permitted spend.

RECOMMENDATION

Add an account-wide transient reentrancy lock covering authorization, calls, and postExecute, and prevent any later nonce/key from executing before the active lifecycle finishes. As defense in depth, validators can maintain per-account context stacks, but Delegation should guarantee non-interleaving because postExecute receives no digest or operation identifier.

RESOLUTION

Status: Acknowledged.

LOW L-35

ACKNOWLEDGED

Missed funding snapshots waive fees

CATEGORY Warning

LOCATION Contract Upgrade & State Migration – Auditor Review Doc

REVIEW Main Review 1

DESCRIPTION

After the upgrade, funding for an existing trade is charged only when its migration snapshot has `openedAfterUpdate == true`. `setTradeInitialAccFundingFees` sets that flag per trade. Any live position omitted from Step 6.3(c) keeps the default record:

```
``plain text
if (!fundingRecord.openedAfterUpdate || isPnl) {
return 0;
}
`
```

Those trades pay no funding for the rest of their life, even though they continue to occupy coin OI and drive the pair funding accumulator for everyone else. Wallet OI, coin OI, and funding snapshots are written in separate batches, but the runbook has no completeness check tying every live non-PnL trade to a successful funding snapshot. A missing or failed funding batch can therefore waive funding even when the OI migration succeeded.

The asymmetry transfers value from the vault / paying side to the missed positions and permanently skews `totalFundingPaid` / `totalFundingReceived` style accounting for those markets.

RECOMMENDATION

Before unpause, require every live non-PnL trade to have `openedAfterUpdate == true` and a funding snapshot equal to the applicable side accumulator. The verification must fail if any expected funding row was not written.

RESOLUTION

Status: Acknowledged.

LOW L-36

ACKNOWLEDGED

Coin OI batches can split pair funding

CATEGORY Warning

LOCATION `\`src/TradingStorage/TradingStorage.sol\`, \`migrateCoinOI\``

REVIEW Main Review 1

— DESCRIPTION

`migrateCoinOI` overwrites each `(pairIndex, side)` aggregate without checkpointing funding and without requiring both sides of a pair in the same transaction. The migration script emits nonzero sides in pair ascending, long then short order and batches every 100 entries, so one pair's long and short rows can land in adjacent transactions.

Funding's pending math always applies the `**current**` long/short coin OI to every block since `lastUpdateBlock`. While only one side is restored, any call that runs `_storeAccFundingFees` freezes that half written book into `accPerOi*`. Trading pause blocks public paths such as `updateMargin`, but several pause independent entypoints still checkpoint, including `updateFundingAccumulators`, manager/keeper funding param setters, risk manager `setTradeInitialAccFundingFees` or `forceCloseTradeMarket`, operator fulfillment of leftover pending market orders, and position rolls.

Under a strictly ordered pause, complete coin OI, then funding snapshot, then unpause sequence with those entypoints idle, no permissionless extraction is shown. The residual risk is operational: a funding update in the batch gap can distort pair accumulators before both sides and per trade funding snapshots are finalized, and can complicate later settlement accounting if a trade is armed against the intermediate state.

— RECOMMENDATION

During coin OI migration, do not exercise any path that can call `_storeAccFundingFees` on an affected pair until both sides of that pair are written and funding snapshots are complete. In particular, keep `updateFundingAccumulators`, funding param updates, force closes, pending market fulfillments, and funding init off half migrated pairs. Prefer migrating both sides of each pair atomically so a mid migration checkpoint cannot observe a split book.

— RESOLUTION

Status: Acknowledged.

LOW **L-37**

RESOLVED

Risk manager forges funding rebates

CATEGORY Validation

LOCATION `src/PairInfos/PairInfos.sol:setTradeInitialAccFundingFees (L1098-L1121), _storeAccFundingFees (L915-L923), getPendingAccFundingFees (L963-L992), getTradeFundingFeePure (L1017-L1026); src/TradingCallbacks/TradeRegistry.sol:427`

REVIEW Main Review 1

— DESCRIPTION

L-37 Risk manager forges funding rebates

CONTINUED

Funding transfers value from the crowded side of a market to the thin side. Rather than charge every position every block, `PairInfos.sol` keeps a counter per pair and side (`accPer0iLong` / `accPer0iShort`) holding cumulative funding per unit of open interest. Each trade stores that counter at open; settlement is $(\text{counterNow} - \text{baseline}) * \text{coinSize}$ (:1017-1026), and a negative result is a credit

`TradeRegistry.sol:427` adds to collateral, paid from the tranche.

The counter advances by $\text{imbalance} * \text{blocksElapsed} * \text{price} * \text{rate}$ (:963-992). The price is a straight multiplier: one ten times too high moves the counter ten times too far. Every normal path takes it from

`PriceAggregator`, signature- and freshness-checked.

`setTradeInitialAccFundingFees` (:1098-1121, `onlyRiskManager`) does not. It takes `uint256[]` `calldata prices` and passes each entry straight to `_storeAccFundingFees(pairIndex, prices[i])` — no oracle, signature, deviation or staleness check — then writes the named trade's baseline from the counter it just moved. It resolves that trade with `storageT.openTrades(...)`, a mapping read returning a zero-filled struct for an unwritten key rather than reverting.

Order matters: each iteration first moves the pair-wide counter using the supplied price, then restamps one named trade's baseline with the new value. An attacker with the risk-manager key holds a non-PnL position on the receiving side of a skewed pair and calls the helper with a price far above market. Naming their own trade would reset its baseline to the poisoned counter and net zero, so they name a nonexistent trader or index; that write lands on a slot nothing reads and their baseline stays honest. A normal close at a validated price pays $(\text{poisonedCounter} - \text{baseline}) * \text{size}$ from the tranche; validating that price does not help, the corruption already landed in global state.

`_storeAccFundingFees` also advances `lastUpdateBlock`, so honest updates start from the forged value and never recompute the interval; every position on the pair then settles against it.

Same root cause as 6a69d55c34dea27378ae45c5, which reaches `_storeAccFundingFees` with an unvalidated price via `forceCloseTradeMarket`; this entry point needs no live trade at all. Medium because it requires the risk-manager key: an operational role, not timelocked governance, and an EOA on the fork carrying the pending upgrade. An honest operator passing a wrong price causes the same irreversible corruption.

— RECOMMENDATION

Route the price through the same authenticated oracle path used by every normal funding update, so

`_storeAccFundingFees` can never be reached with a caller-supplied number. Apply this to

`forceCloseTradeMarket` as well, since it is the second entry point of the same defect.

Separate the two jobs the function performs. Migrating a global accumulator and repairing one trade's snapshot should not happen in the same call: checkpoint pair-wide funding once

L-37 Risk manager forges funding rebates**CONTINUED**

with a validated price, then write per-trade baselines without touching global state.

Require each tuple to resolve to a live trade, since `storageT.openTrades` returns a zero-filled struct for a key that was never written rather than reverting, and include `prices` in the `LengthMismatch` check, which currently validates only `traders`, `pairIndexes` and `indexes`. Finally, restrict the helper to timelocked governance and remove it from production deployments once migration is complete.

— RESOLUTION

Status: Resolved.

INFORMATIONAL I-01

ACKNOWLEDGED

PnL liquidations skip vault fee

CATEGORY	Logical Error
LOCATION	LimitCallbackModule.sol:executeLimitCloseOrderCallback, TradeRegistry.sol:unregisterTrade, TradingStorageCoreExtension.sol:applyReferralAndPnlFee
REVIEW	Main Review 1

DESCRIPTION

PnL liquidations compute a total zero-fee-perp liquidation fee, but settlement discards the vault/LP portion and charges only the liquidator reward.

In `executeLimitCloseOrderCallback`, the PnL liquidation branch selects `zfpLiqFeeP` as the liquidator reward and `zfpLiqTotalFeeP` as the total liquidation fee. It then derives the LP-side fee as:

```
``plain text
values.levPosToken = values.reward * ( totalLiquidationP - liquidatorFeeP) / liquidatorFeeP;
```

That LP-side fee is passed to `_unregisterTrade` as `_lpFeeToken`. However, `_unregisterTrade` forwards it to `TradingStorage.applyReferralAndPnlFee`, where `_isPnl == true` replaces the supplied fee with `getPnlBasedFee`. For losing PnL trades, including liquidations, `getPnlBasedFee` returns zero.

As a result, a PnL liquidation configured with `zfpLiqFeeP = 5` and `zfpLiqTotalFeeP = 10` only charges the 5% liquidator reward. The intended additional 5% vault/LP liquidation fee is ignored.

Impact

Every PnL liquidation under-collects the configured total liquidation fee. The keeper still receives the liquidator reward, but the vault/LP fee component is not charged or allocated, so distressed zero-fee-perp positions pay less than governance configured and LPs lose liquidation-fee revenue.

RECOMMENDATION

Preserve the liquidation fee when `_isPnl` is true. For liquidation closes, do not overwrite `_lpFeeToken` with `getPnlBasedFee`, or pass an explicit close reason into `applyReferralAndPnlFee` so PnL profit fees are only substituted for normal profitable PnL closes.

Kartik:

I-01 PnL liquidations skip vault fee

CONTINUED

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-02

ACKNOWLEDGED

Stale decrease size order type

CATEGORY	Informational
LOCATION	src/interfaces/IPriceAggregator.sol:OrderType
REVIEW	Main Review 1

— DESCRIPTION

`IPriceAggregator.OrderType` still includes `DECREASE_SIZE`, and `PriceAggregator.maxDeviationP` classifies it as a close-side order type. No live creation path appears to request prices with `OrderType.DECREASE_SIZE`, and `_executeCallback` has no matching callback branch for it. Position decreases are implemented through the market-close path instead: `closeTradeMarket` stores a `MARKET_CLOSE` or `MARKET_CLOSE_PNL` request with the collateral amount to close, and `closeTradeMarketCallback` performs full or partial close settlement.

— RECOMMENDATION

Remove `DECREASE_SIZE` if it is no longer part of the design, or add an explicit unsupported-order revert in the aggregator callback dispatch. If a dedicated decrease-size flow is planned, add the matching callback path before exposing any creation route.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-03

ACKNOWLEDGED

Voluntary closes forfeit liquidation buffer

CATEGORY	Unexpected Behavior
LOCATION	src/PairInfos/PairInfos.sol:getTradeValuePure
REVIEW	Main Review 1

DESCRIPTION

`getTradeValuePure()` applies the liquidation floor inside shared close-settlement math, not only inside the liquidation callback:

```
``plain text
int256 fees = int256(rolloverFee) + int256(closingFee) + fundingFee;
int256 value = int256(collateral) + lossProtectedPnl - fees;
if (value <= (int256(collateral) * int256(100 - liqThreshold)) / 100) {
    value = lossProtectedPnl - pnl;
    lossProtectedPnl = fees - int256(collateral) + value;
}
```

With the default `liqThreshold = 85` and inert loss protection, any close whose post-PnL, post-fee value is at or below 15% of collateral returns zero to the trader. This applies to normal market closes, TP, and SL closes because they all route through the same `getTradeValuePure()` settlement path, even when no liquidation order was executed and no liquidation validity check was performed.

Example: a voluntary market close with 1,000 USDC collateral and 149 USDC remaining after PnL, rollover, funding, and close fees returns 0. If the trader first deposits enough margin to move above the floor, the same position can close with residual value returned. This creates a sharp payout cliff and makes pre-close margin top-ups materially change proceeds.

RECOMMENDATION

Apply the liquidation-buffer forfeiture only for liquidation closes, or pay voluntary closes their net value after fees. If the intended rule is that any below-floor close forfeits the buffer, expose and document that behavior clearly so users know to top up or expect zero payout.

I-03 Voluntary closes forfeit liquidation buffer**CONTINUED**

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-04

ACKNOWLEDGED

overwriteTrade silently skips missing slots

CATEGORY Warning
LOCATION TradingStorageCoreExtension.sol
REVIEW Main Review 1

DESCRIPTION

`overwriteTrade()` returns when the target slot has no live trade instead of reverting:

``plain text

```
ITradingStorage.Trade storage existingTrade = _openTrades[_trade.trader][_trade.pairIndex][_trade.index];  
if (existingTrade.leverage == 0) {  
    return;  
}
```

The position increase registry assumes this write succeeds. `_applyPositionIncreaseStorage()` increments group OI, calls `overwriteTrade(existingTrade)`, and then increments per pair and wallet OI. If a stale or inconsistent increase path ever reaches this branch, the merged trade would not be stored while collateral movement, reward allocation, pending order cleanup, and OI accounting may still continue. The normal empty slot stale increase paths tend to revert earlier through direction or zero state arithmetic, so this is primarily a defense in depth issue. The storage primitive should not silently ignore a missing position when callers rely on it as a required state transition.

RECOMMENDATION

Revert with `NoTrade` when `overwriteTrade()` is called for an empty slot. Position increase callbacks should also bind pending orders to the original position timestamp or nonce before doing any accounting.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-05

ACKNOWLEDGED

Update mode writes Pyth state unnecessarily

CATEGORY	Informational
LOCATION	PriceAggregator.sol
REVIEW	Main Review 1

DESCRIPTION

`_pythCorePrice` has two branches. Open, close, limit, and TWAP order types use `parsePriceFeedUpdates`, which verifies the submitted payload and returns the price without writing storage. Margin, SL, and increase order types instead call `updatePriceFeeds` followed by `getPriceNoOlderThan(job, 30)`, which persists the price into the Pyth contract before reading it back.

For these order types the price is only needed transiently to run one health check in the same transaction, so the storage write serves no protocol purpose and adds gas on top of the update fee that both modes already pay. `getPriceNoOlderThan` alone cannot replace it, because in a pull oracle the fresh price is not on chain until something writes it, but `parsePriceFeedUpdates` already provides the same freshness guarantee without the write and is used elsewhere in the same contract.

The two modes also differ in which price is consumed. `parsePriceFeedUpdates` returns exactly the submitted payload. `updatePriceFeeds` only overwrites when the submitted price is newer and then `getPriceNoOlderThan` reads the freshest stored value, so a newer third party push in the same block is used instead of the caller's payload. This is generally harmless and slightly reduces stale price selection, but it means the settled price is not deterministically the one the caller supplied.

RECOMMENDATION

Use `parsePriceFeedUpdates` (or `parsePriceFeedUpdatesUnique`) for the margin, SL, and increase order types to avoid the storage write and match the open and close paths, unless persisting the canonical Pyth price for other consumers is intended. If update mode is kept deliberately, document that the consumed price is the freshest stored value rather than the submitted payload.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I - 06

ACKNOWLEDGED

RWA oracle states lack an action policy

CATEGORY	Suggestion
LOCATION	PriceAggregator.sol
REVIEW	Main Review 1

DESCRIPTION

Avantis has no explicit action policy for Pyth Lazer's RWA market sessions. The settlement path requests and parses only `price`, `exponent`, and `confidence`, validates the bundle timestamp, and then treats the price as usable. `closeOnlyMode` is the only pair-level market control. `marketSession` and oracle health describe different risks and must not be conflated:

```
<table header-row="true">
<colgroup>
<col>
<col width="735">
</colgroup>
<tr>
<td>Signal</td>
<td>Meaning</td>
</tr>
<tr>
<td>regular </td>
<td>The primary market is in its regular trading session.</td>
</tr>
<tr>
<td>preMarket </td>
<td>The pre-market session is active.</td>
</tr>
<tr>
<td>postMarket </td>
<td>The post-market session is active.</td>
</tr>
<tr>
<td>overNight </td>
<td>An overnight venue session is active.</td>
</tr>
```

I-06 RWA oracle states lack an action policy

CONTINUED

```

<tr>
<td> closed </td>
<td>No supported venue session is active. A price may still be present because Pyth carries the last
aggregate forward.</td>
</tr>
<tr>
<td> feedUpdateTimestamp == timestampUs </td>
<td>The feed produced a new aggregate in this bundle.</td>
</tr>
<tr>
<td> feedUpdateTimestamp < timestampUs </td>
<td>The price and its related properties were carried forward. This can occur during closures or when too
few publishers contribute.</td>
</tr>
</table>

```

A fresh price is not necessarily safely tradable. Pre-market, post-market, and overnight prices may be current while the protocol's hedge venue has less depth, larger spreads, or is unavailable. Likewise, `confidence` measures publisher disagreement and is not a standalone liquidity test. `publisherCount`, confidence, session, feed freshness, and actual hedge availability should therefore be evaluated independently.

— RECOMMENDATION

Request, authenticate, and parse `marketSession`, `feedUpdateTimestamp`, and `publisherCount` onchain. Configure the allowed actions per pair and session rather than relying on a global schedule or backend-only enforcement.

```

<table header-row="true">
<colgroup>
<col>
<col width="751">
</colgroup>
<tr>
<td>State</td>
<td>Recommended protocol behavior</td>
</tr>
<tr>
<td> regular with a fresh, healthy feed</td>
<td>Permit normal trading subject to existing risk limits.</td>
</tr>
<tr>

```

I-06 RWA oracle states lack an action policy

CONTINUED

<td> preMarket , postMarket , or overNight with a fresh, healthy feed and live hedge venue</td>
<td>Use pair-specific lower leverage and OI caps, wider spreads, tighter oracle-quality bounds, and smaller order-size limits. Disable the session entirely when Avantis cannot hedge it.</td>
</tr>
<td>Any active session with a carried or degraded feed</td>
<td>Allow collateral deposits and cancellations. Block new exposure, increases, withdrawals, and price-dependent settlement.</td>
</tr>
<td>closed </td>
<td>Allow collateral deposits, cancellations, and close requests, but queue price-dependent closes, TP/SL, and liquidations until a fresh tradable price exists. An independently live and hedgeable RFQ path may use a separate policy.</td>
</tr>
<td>Reopening</td>
<td>Require a fresh feed with sufficient publishers and acceptable confidence. Apply conservative limits for an initial reopening window before restoring normal parameters.</td>
</tr>
</table>

Define pair-specific thresholds for maximum feed age, minimum publisher count, confidence-to-price ratio, and optionally disagreement with a fresh backup feed. A market action should proceed only if both its session policy and oracle-health policy pass.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-07

ACKNOWLEDGED

Tranche pull cap is per call only

CATEGORY Warning
LOCATION AvantisTranche.sol
REVIEW Main Review 1

DESCRIPTION

`AvantisTranche.withdrawAsVaultManager` limits each manager pull to less than `500,000e6`, but it does not track how much has been pulled over time. `VaultManager._sendUSDCToTrader` calls it whenever a trader payout exceeds the vault manager's available liquid balance:

```
``plain text
if (int256(_amount) > balanceAvailable) {
uint256 difference = uint256(int256(_amount) - int256(balanceAvailable));
junior.withdrawAsVaultManager(difference);
}
`
```

As a result, the fixed limit only rejects a single large shortfall. Several settlements with 499,999 USDC shortfalls can cumulatively pull much more than 500,000 USDC from the tranche. This does not bypass the tranche `depositCap`, which is checked against aggregate `totalAssets() + assets`, but it means the withdrawal cap is not useful as an aggregate loss or liquidity drain limiter.

RECOMMENDATION

If the cap is intended only as a per call safety check, document that limitation. If it is intended to limit cumulative tranche pulls, replace it with epoch based or rolling aggregate accounting, or enforce the limit at the vault manager payout layer. Consider enforcing rate limits at the Keeper level instead.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-08

ACKNOWLEDGED

Multicall upgrade bricks trade settlement

CATEGORY	Logical Error
LOCATION	script/testnet/upgradeMulticall.s.sol:_upgradeMulticall(), script/testnet/upgrade.s.sol:_upgradeMulticall()
REVIEW	Main Review 1

DESCRIPTION

Both supplied upgrade scripts deploy a new `Multicall` implementation but pass the Referral proxy to `ProxyAdmin.upgrade()`:

```
```plain text
function _upgradeMulticall() internal {
 vm.startBroadcast(deployer);
 Multicall newImpl = new Multicall();
 vm.stopBroadcast();

 _proxyAdminUpgrade(address(referral), address(newImpl));
}
``
```

A valid execution therefore replaces Referral's implementation with `Multicall` while leaving the actual `Multicall` proxy unchanged. Transparent proxy upgrades do not check ABI compatibility, so the upgrade succeeds and Referral's state remains stored but its functions become inaccessible. Every close, including market closes, TP/SL execution, partial closes, and liquidations, calls `Referral.traderReferralDiscount()` through `TradeRegistry.unregisterTrade()` and `TradingStorageCoreExtension.applyReferralAndPnlFee()`. Because that selector does not exist on `Multicall`, settlement reverts and the callback cannot clear its pending order. Normal opens and position increases that charge an opening fee fail through the same missing Referral call. Users cannot exit positions and keepers cannot liquidate unhealthy positions until governance restores the Referral implementation, allowing bad debt risk to grow during the outage. The replacement also exposes `Multicall13`'s permissionless aggregate functions at the Referral address. Arbitrary callers can consequently make external calls with the Referral proxy as `msg.sender`, which can abuse any balance, allowance, or external permission assigned to that address.

---

**I-08** Multicall upgrade bricks trade settlement

---

**CONTINUED****— RECOMMENDATION**

Change both `_upgradeMulticall()` functions to upgrade `address(multicall)`, and ensure the deployment and impersonation RPC select the same intended environment.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-09

ACKNOWLEDGED

# Avantis price source trusts publisher

CATEGORY Trust Assumptions

LOCATION src/PriceAggregator.sol:\_validatedPrice, \_avantisPrice, \_validateBackupPrice

REVIEW Main Review 1

## DESCRIPTION

The AVANTIS price source depends on the whitelisted publisher as a trusted actor. `_avantisPrice` verifies that the submitted payload was signed by an enabled publisher, but `_validatedPrice` deliberately skips the confidence ratio bound for this source and relies only on an optional Chainlink backup check afterward.

```

`plain text
require(
price > 0 &&
(_priceSourcing == PriceSourcing.AVANTIS ||
((conf * _PRECISION * 100) / price) <= maxDeviationP(r.pairIndex, r.orderType)),
"PRICE_DEVIATION_TOO_HIGH"
);
`

```

The publisher signature is the only gate on the submitted price and confidence values:

```

`plain text
!storageT.whitelists().verifySignerIsPublisher(
keccak256(priceUpdateData[0]).toEthSignedMessageHash().recover(priceUpdateData[1])
)
`

```

The backup deviation check does not close the gap. `_validateBackupPrice` runs only when a pair has both a configured backup feed and a nonzero deviation bound, and it returns early without reverting when the backup price is zero or stale, so a missing or stale anchor disables the bound instead of blocking settlement.

If the same operational backend controls both the execution operator and the Avantis publisher key, or if the publisher key is compromised, that component can sign and submit arbitrary price and confidence values for settlement. This is an operational trust assumption rather than a permissionless vulnerability, but privileged key compromise is a realistic protocol risk and should be explicit. It compounds with the separately reported issue that most pairs lack an effective oracle backup, since without a fresh backup the publisher is entirely unchecked.

---

**I-09** Avantis price source trusts publisher

---

**CONTINUED****— RECOMMENDATION**

Document whether the execution operator and Avantis publisher are independent. Keep the publisher key separated from operator infrastructure, preferably behind quorum signing or an HSM, and require backend sanity checks against independent sources before signing. For pairs using the Avantis source, consider requiring a fresh backup price or another onchain bound for high impact settlement paths instead of silently skipping the anchor when it is unavailable.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-10

ACKNOWLEDGED

# Builder code hijack diverts user trade fees

CATEGORY	Improper access control
LOCATION	src/EIP7702Template/BuilderCode.sol:registerBuilderCode, script/deploy7702.s.sol:run
REVIEW	Main Review 1

## DESCRIPTION

`BuilderCode.registerBuilderCode` is unauthenticated. Any address can claim any unused `bytes32` identifier, become its permanent `builderCodeOwner`, and set an arbitrary `_feeCollector`. The ownership check in `modifyBuilderCode` applies only after that first registration, so it does not protect the initial claim, and the fee caps bound only the size of the fee, not who receives it.

The delegate template makes the identifier claimable in practice. `script/deploy7702.s.sol` derives the builder code deterministically from the public string `avantis.integrator`, producing `0x2583d8796bcd325b6f6587f3c682f70d4e3d7abc9e5c7ea7b31ca85b3de6b00`, bakes it into the `Delegation` bytecode as an immutable, initializes the registry with a 10 USDC fixed fee cap and a 1 percent fee cap, and never registers the code. Anyone can compute that value before deployment and claim it first.

The claimed configuration is charged against user balances on every fee eligible action, so this diverts user funds rather than builder revenue:

```
```plain text
uint256 fee = isPercentFee ? (_collateral * feePercent) / 1e12 : fixedFee;

if (fee > 0 && _collateral > 0) {
if (_outerSelector == DELEGATED_ACTION_SELECTOR) {
bool ok = USDC.transferFrom(_trader, feeCollector, fee);
require(ok, "FEE_TRANSFER_FAILED");
} else {
(bool success) = USDC.transfer(feeCollector, fee);
require(success, "FEE_TRANSFER_FAILED");
}
}
```
```

Under EIP-7702 the account performing `transfer` is the user's own EOA, and the `delegatedAction` branch pulls from the trader's allowance. At the deployed 1 percent cap, a 10,000 USDC open sends 100 USDC to the attacker's collector, repeated for every open, position increase, TWAP open, and RFQ open.

**I-10** Builder code hijack diverts user trade fees

CONTINUED

Two factors make the race near certain. `_chargeBuilderFees` reverts with `BUILDER_CODE_NOT_REGISTERED` while the owner is zero, so the template cannot execute any fee eligible action until somebody registers the code. Registration is therefore a mandatory launch step that the script omits, and the first caller wins it while also making the template appear to begin working normally. Separately, nothing in the user's signed calls binds the builder configuration, so an attacker can register at zero fee, wait for adoption, then raise the fee to the cap through `modifyBuilderCode`.

`overrideBuilderCode` allows governance to reassign the code, so losses are bounded by detection and response time rather than by any protocol level limit.

**— RECOMMENDATION**

Authenticate registration, for example by restricting a code to a governance approved registrant, and register the template's code in the same deployment transaction that publishes the `Delegation` bytecode. Confirm on chain that the embedded code resolves to the intended owner and collector before advertising the delegate. Consider skipping the fee instead of reverting when a code is unregistered, so an unclaimed code cannot block trading.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I - 11

ACKNOWLEDGED

# Extreme Funding Freezes Settlement

CATEGORY Validation

LOCATION `avantis-contracts-team2-1784553551238/src/PairInfos/PairInfos.sol`, `avantis-contracts-team2-1784553551238/src/TradingCallbacks/TradeRegistry.sol`

REVIEW Main Review 1

## DESCRIPTION

Funding transfers value between long and short traders according to the imbalance between both sides of a pair. Avantis stores a per-pair funding rate and periodically adds the pending amount to cumulative funding values. Closing or liquidating a position must perform this calculation before the trade can be removed and its collateral paid.

`PairInfos.sol` allows the configured keeper to replace the complete funding parameters through `setFundingParamsByKeeper`. Unlike the rollover-rate setters, the funding setter accepts any `uint256` for `fundingFeePerBlockP`. The calculation later casts it to `int256` and multiplies it by the coin-OL imbalance, elapsed blocks, price and precision factors using checked arithmetic. A value can therefore be accepted but make every later accrual revert because an intermediate multiplication overflows.

To reproduce on a local deployment, configure a test account as keeper and create unequal long and short coin OL on one pair. Call `setFundingParamsByKeeper` with `fundingFeePerBlockP` equal to the maximum positive `int256` value; the transaction succeeds because no range is checked. Advance one block and request `getPendingAccFundingFees` for that pair. The calculation reverts. Then execute a normal close or liquidation through the callback-authorized path. `TradeRegistry.sol` calls `getTradeValue`, which stores pending funding before unregistering the trade, so the same arithmetic revert rolls back the complete settlement. The position remains open and no collateral is paid.

The manager's ordinary funding setters do not immediately solve the condition because they also accrue the poisoned rate before assigning its replacement. A migration helper can first move funding `lastUpdateBlock` to the current block; a manager can use it and replace the rate before another block elapses. Recovery therefore requires privileged diagnosis and a correctly ordered intervention while users cannot close or liquidate affected positions.

A compromised or malfunctioning keeper could repeat this on every pair and prevent exits, including liquidations, until a manager repairs them. Positions could accumulate losses and create bad debt after settlement resumes. During the review, the team stated that keepers are trusted. Because exploitation requires failure of this explicitly trusted role, the issue is retained as a trust-assumption observation and its criticality is reduced to Info.

---

**I-11** Extreme Funding Freezes Settlement

---

**CONTINUED****— RECOMMENDATION**

Apply an explicit upper bound to every funding parameter before storing it, using a protocol-approved economic maximum that remains arithmetically safe under the largest supported OI, price and elapsed-block values. Reject values above `type(int256).max` before any cast and structure the calculation with bounded intermediate values or checked mul-div helpers. Use one internal validation routine from the keeper, manager and initial configuration setters so no privileged path bypasses the limit. Keep an emergency manager function that can replace a bad rate without first accruing it, but make it reset the timestamp and emit a dedicated recovery event atomically. Add boundary tests for maximum OI, maximum price, long inactivity and both OI directions.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-12

ACKNOWLEDGED

# Roll Keeper Forges Entry Price

CATEGORY Trust Assumptions

LOCATION `avantis-contracts-team2-1784553551238/src/TradingStorage/TradingStorage.sol`, `avantis-contracts-team2-1784553551238/src/TradingStorage/TradingStorageCoreExtension.sol`

REVIEW Main Review 1

## DESCRIPTION

Position rolling adjusts trades when a market moves to a new contract or reference price. It should preserve economic position: the reference-price change determines the replacement entry, while current authenticated data settles accrued fees. Avantis delegates this operation through `positionRollKeepers` while governance enables rolling for a pair.

`TradingStorage.sol` restricts `positionRoll` to keepers but forwards their values to `TradingStorageCoreExtension.sol`. The extension obtains an authenticated market price through `PriceAggregator.validatePrice`, but uses it only for borrowing and funding fees. Keeper-supplied `_newOpenPrice` is never compared with that price, a signed adjustment, a bounded factor or a governance-approved reference.

The value replaces `existingTrade.openPrice` and determines new open interest and leverage. Future closes calculate profit from it and transfer a positive result from `VaultManager`. A compromised keeper can choose a low nonzero entry for a colluding long, roll using valid oracle data, then close at the real price. Pair maximum gain limits the amount, but it remains artificial protocol-funded profit. Exposure and reserves are recalculated from the forged value and do not prevent payment.

To reproduce, enable rolling for a test pair and authorize a test keeper. Open a controlled long at the market price. Call `positionRoll` with fresh oracle data but a much lower nonzero `_newOpenPrice` that leaves valid leverage. `RollAdjustmentDone` and the trade getter show the forged entry. Close normally and compare the payout with a legitimate control roll. The validated test stored 1,000 while the oracle reported 50,000 and produced a larger USDC payout.

A compromised roll keeper could collude across an active pair and convert fabricated PnL into real vault USDC. During the review, the team stated that keepers are trusted. Because exploitation requires failure of this explicitly trusted role, the issue is retained to document the trust assumption and its criticality is reduced to Info.

## RECOMMENDATION

Do not accept an unconstrained replacement entry price from the keeper. Supply authenticated before-roll and after-roll market references, calculate the adjustment on-chain, and

**I-12** Roll Keeper Forges Entry Price**CONTINUED**

bind them to the pair, roll identifier and validity window. If an external roll index is unavoidable, require a quorum-signed value from an independent authorised source and enforce a governance-configured maximum deviation from the current oracle price. Reject zero or out-of-range results before changing collateral, leverage, open interest or fee snapshots.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-13

ACKNOWLEDGED

# TWAP Schedule Can Be Bypassed

CATEGORY Business logic errors

LOCATION `avantis-contracts-team2-1784553551238/src/Trading/TwapTradingModule.sol`, `avantis-contracts-team2-1784553551238/src/TradingCallbacks/IntentCallbackModule.sol`

REVIEW Main Review 1

## DESCRIPTION

A time-weighted average price order, or TWAP, divides one large trade into smaller slices executed across a trader-selected period. This reduces the risk of entering or exiting the complete position at one price. When Avantis accepts a TWAP, `TwapTradingModule.sol` validates runtime against the pair's minimum, maximum and frequency. It calculates the expected slice count and checks that the order is large enough, making runtime an apparent part of the trader's enforced instruction.

The execution path does not enforce this commitment. `TwapState` stores total size and the last slice identifier, but not the start time, deadline, frequency or required slice count. The operator-only execution functions merely require a `sliceId` greater than the previous value. They do not require the next identifier, wait for its interval or reject execution after the selected runtime.

The operator can therefore execute the intended slices in consecutive transactions immediately after creation, compressing a multi-hour plan into one block. It can also skip identifiers or wait until after the selected runtime. Closing has the equivalent weakness. Runtime and cadence are emitted for off-chain consumers, but the scheduler cannot constrain an authorised operator that calls the contract directly. Slice-sizing weaknesses are covered by a separate finding and are not required for this timing bypass.

To reproduce, create an open TWAP for 600 USDC over six five-minute slices. From an enabled test operator, immediately submit six execution transactions with consecutive identifiers, the intended 100 USDC per slice and valid price data. All transactions can succeed before the first five-minute interval has elapsed, completing what should have been a thirty-minute plan. A fresh plan's slice can also execute after its declared runtime because no expiry is checked.

A compromised execution operator could compress or delay the schedule, exposing all collateral to one price point and increasing slippage, adverse timing, liquidation risk and loss. During the review, the team stated that keepers, including the relevant execution operator, are trusted. Because exploitation requires failure of this explicitly trusted role, the issue is retained as a trust-assumption observation and its criticality is reduced to Info.

## I-13 TWAP Schedule Can Be Bypassed

CONTINUED

### — RECOMMENDATION

Persist the TWAP creation time, runtime, frequency and total slice count in `TwapState`. Require exactly the next slice identifier and derive its earliest execution time from the stored start and frequency. Reject early execution, skipped identifiers and any slice submitted after the final deadline. Apply the same temporal rules to opening and closing TWAPs. Keep the independent on-chain sizing constraints described by the slice-size finding, and test early, back-to-back, skipped, late and final-deadline executions.

### — RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-14

ACKNOWLEDGED

# Close slices exceed authorized amount

CATEGORY Trust Assumptions

LOCATION `src/Trading/RfqTradingModule.sol:executeCloseRfqSlice; src/Trading/TwapTradingModule.sol:executeCloseTwapSlice; src/TradingCallbacks/IntentCallbackModule.sol:closeTwapCallback, closeRfqCallback`

REVIEW Main Review 1

## DESCRIPTION

TWAP and RFQ close orders allow a trader to authorize settlement of only part of a leveraged position. The EIP-712 close-order hashes commit to `coinSizeToClose`, and the value is persisted as `state.coinSize`. Execution happens later in operator-controlled slices whose `coinSize` is not part of the trader's signature. `executeCloseTwapSlice` and `executeCloseRfqSlice` verify that the intent exists, is a close, and uses a newer slice ID, but never require `slice.coinSize <= state.coinSize - state.coinSizeClosed`. The callbacks do not restore this boundary. They only revert when the amount closed before the current callback is already greater than the authorized total:

```
``plain text
```

```
if (state.coinSizeClosed > state.coinSize) revert ...;
bool isComplete = state.coinSizeClosed + slice.coinSize >= state.coinSize;
`
```

For example, if a trader signs a close for 10 coins and nothing has yet closed, an operator can submit a 100-coin slice. The check evaluates `0 > 10` as false, while completion evaluates `0 + 100 >= 10` as true. The callback therefore processes 100 coins instead of rejecting the excess. It converts that slice to collateral; if the result covers the live position or leaves only dust, it clamps the amount to all remaining collateral, calls `_unregisterTrade`, and deletes the intent as complete. A valid partial-close authorization can consequently close the trader's complete position.

RFQ slippage limits only the price, `sliceld` prevents stale slices, and `positionTimestamp` prevents targeting a replacement position; none restrict quantity. The outcome is unauthorized realization of the complete PnL, full close fees, and removal of exposure when only a partial close was authorized. During the review, the team indicated that keepers, including these execution operators, are considered trusted. Since exploiting the issue requires an authorized operator to submit an excessive slice, the finding is reduced to Info and retained as a trust-assumption and defense-in-depth observation.

**I-14** Close slices exceed authorized amount

CONTINUED

**— RECOMMENDATION**

Before requesting a price and again inside each callback, compute `remaining = state.coinSize - state.coinSizeClosed` ; reject zero-sized slices and any `slice.coinSize > remaining` . The callback check is required because multiple asynchronous requests may be pending against the same pre-callback state. After validation, complete the intent only when the slice exactly consumes the authorized remainder. Use the validated effective size consistently for collateral conversion, fees, accounting, and events. Dust handling may close the live position only when the trader-authorized remaining amount itself covers that residual position. Apply the same helper to TWAP and RFQ closes and test exact final fills, one-unit overfills, full-position overfills, and concurrent callbacks.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-15

ACKNOWLEDGED

# Forex skew test misses production branch

|          |                                                 |
|----------|-------------------------------------------------|
| CATEGORY | Logical Error                                   |
| LOCATION | test/fixtures/Base.t.sol\`, \`__setupForexPairs |
| REVIEW   | Main Review 1                                   |

## DESCRIPTION

The original shared fixture created one crypto group before `FOREX`, assigned its FX pairs to group 1, and therefore differed from live v1.5, where `CRYPTO01`, `CRYPTO02`, and `FOREX` are groups 0, 1, and 2. V2 only records category USD OI and applies the 75% category plus 25% pair skew calculation to group 2:

```
``plain text
if (pairsStorage.pairGroupIndex(_pairIndex) == 2) {
 uint256[2] memory usdOI = storageT.getUsdOI();
 // ...
 return (75 * usdSkew + 25 * assetSkew) / 100;
}
```

`test_getBlendedSkew_forex_group`` only asserted that the result was nonzero. The group 1 fallback also returns a nonzero pair skew, so the test passed without exercising the production forex branch. This could hide regressions in forex USD direction mapping and blended borrow fee calculations.

## RECOMMENDATION

Mirror the production group ordering in the shared fixture and assert the exact blended result using nonzero category USD OI, so the pair only fallback cannot satisfy the test.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-16

ACKNOWLEDGED

# FOREX util skips category blend

|          |                                                                 |
|----------|-----------------------------------------------------------------|
| CATEGORY | Logical Error                                                   |
| LOCATION | \`src/PairInfos/PairInfos.sol\`, \`getBlendedUtilizationRatio\` |
| REVIEW   | Main Review 1                                                   |

## DESCRIPTION

FOREX docs and the trader SDK define dynamic margin (borrow) inputs as:

```
``vb.net
```

```
Blended Utilization = 0.75 * Category Utilization + 0.25 * Asset Utilization
```

```
Blended Skew = 0.75 * Category Skew + 0.25 * Asset Skew
```

```
`|
```

On chain, `getBlendedSkew` implements a FOREX category blend for group 2, but `getBlendedUtilizationRatio` returns only asset utilization:

```
`plain text
```

```
pairUtilization = (pairLongOI + pairShortOI) * 1e10 / pairMaxOI(_pairIndex);
```

```
`|
```

`pairMaxOI` may be derived from `groupMaxOI`, so the group cap sizes the pair limit, but category utilization (`groupOI / groupMaxOI`) is never computed or mixed in.

`getPendingAccRolloverFees` feeds this value straight into the dynamic borrow multiplier, so FOREX borrow fees diverge from the documented and SDK rate whenever category util and pair util differ.

Illustration: FOREX category nearly full, USDJPY almost empty.

```
<table header-row="true">
```

```
<tr>
```

```
<td>Bucket</td>
```

```
<td>OI</td>
```

```
<td>Cap</td>
```

```
<td>Utilization</td>
```

```
</tr>
```

```
<tr>
```

```
<td>FOREX category (EURUSD + GBPUSD + USDJPY + ...)</td>
```

```
<td>\\\ $120,000</td>
```

```
<td>\\\ $150,000</td>
```

```
<td>80%</td>
```

```
</tr>
```

```
<tr>
```

**I-16** FOREX util skips category blend

CONTINUED

`<td>USDJPY alone</td>``<td>\\$5,000</td>``<td>\\$50,000</td>``<td>10%</td>``</tr>``</table>```` vb.net``Docs / SDK blended U = 0.75 * 80% + 0.25 * 10% = 62.5%``On chain U = 10%`````

If the pair utilization threshold is 20%, on chain stays on the base borrow multiplier while the documented formula is already deep into the dynamic region (therefore charging less fees). The inverse case (quiet category, crowded USDJPY) overstates utilization and can overcharge USDJPY borrowers relative to docs/SDK.

**— RECOMMENDATION**

For FOREX (group 2), compute category utilization as `group0I / groupMax0I`, blend it 75/25 with pair utilization, and use that value in rollover accrual so on chain borrow fees match the documented FOREX margin formula. If asset only util is intentional, remove the blended naming and align docs/SDK.

**— RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-17

ACKNOWLEDGED

# Returning governance source can replay a consumed upgrade and downgrade the Pyth proxy

CATEGORY	Informational
LOCATION	packages/onchain-base-type4/src/__onchain/pythupgradable-41c9e39574f40ad3/project:/contracts/pyth/PythGovernance.sol:verifyGovernanceVM():L42-L59 packages/onchain-base-type4/src/__onchain/pythupgradable-41c9e39574f40ad3/project:/contracts/pyth/PythGovernance.sol:AuthorizeGovernanceDataSourceTransfer():L117-L169 package s/onchain-base-type4/src/__onchain/pythupgradable-41c9e39574f40ad3/project:/contracts/pyth/PythSetters.sol:setLastExecutedGovernanceSequence():L44-L46
REVIEW	Main Review 1

## DESCRIPTION

Pyth stores one sequence watermark for the active governance source.

AuthorizeGovernanceDataSourceTransfer overwrites it with the incoming claim VAA's sequence and retains no per-source high-water mark. If source B returns using a valid unused higher-index claim emitted before later actions from B's prior tenure, the transfer lowers B's watermark and makes already-consumed authentic actions executable again. The PoC keeps Wormhole and governance honest: B authentically upgrades to V1, patches to V2, and rotates to C. After C validly returns to B through an older claim, an unprivileged caller replays the consumed V1 upgrade and downgrades the live proxy.

## RECOMMENDATION

See description.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-18

RESOLVED

# Retired PnL positions manufacture funding credits that drain LP principal

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:\_increasePositionSize():L277-L342 packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:\_increasePositionSizeWithCoinExposure():L421-L483 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncrease():L190-L213 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncreaseWithCoinExposure():L258-L280 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeFundingFee():L932-L951 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccFundingFees():L963-L990

REVIEW Main Review 1

## DESCRIPTION

Both increase paths rely only on a position's historical posType and do not require the pair to still permit PnL exposure. After PnL mode is disabled, a retained PnL anchor can therefore absorb new collateral and OI while remaining opening-fee and funding-liability exempt. The added coin OI nonetheless enters pair-wide funding accrual, creating funding credits for ordinary counterpositions without corresponding payer debt. The revised PoC passes through normal requests and callbacks, exercises both increase variants, and proves all pending-order counters balance.

Impact: An attacker can turn a small pre-transition PnL anchor into OI-cap-sized exempt exposure and pair it with an ordinary funding receiver. At a flat price, the PoC realizes over 400 USDC net attacker profit and removes over 400 USDC from AvantisTranche LP principal. The loss scales with OI and accrual time and can be directionally hedged.

## RECOMMENDATION

Reject increases when the stored position type differs from the pair's currently enabled type, or freeze/convert legacy positions during transitions. Independently exclude funding-exempt PnL OI from funding calculations or charge it the corresponding liability so funding remains conserved.

## RESOLUTION

Status: Resolved.

## INFORMATIONAL I-19

ACKNOWLEDGED

# Signed TP/SL update for a closed position can destroy a replacement at the reused index

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/library/SignatureHelpers.sol:\_hashUpdateTpSlReq():L222-L237 packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:updateTpSlWithSig():L486-L517 packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:\_updateTp():L525-L541 packages/avantis-contracts-team2-1783453751962/src/TradingCallback/s/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L166-L315

REVIEW Main Review 1

## DESCRIPTION

UpdateTpSlReq authenticates the trader, pair, reusable numeric index, new TP/SL, deadline, and nonce, but not the position timestamp/incarnation, side, open price, or expected update version. A valid unconsumed request therefore survives full closure and applies to an unrelated replacement that later reuses the index. \_updateTp loads the replacement and does not require the proposed TP to remain on its profitable side. The passing fork PoC signs a 55,000 TP for a position opened at 50,000, closes it, opens a five-times-larger replacement at 100,000 in the reused index, and submits the old signature. At an unchanged authenticated price of 100,000, TP execution in supported guaranteed-price mode substitutes the stale 55,000 TP and realizes the capped 100% loss on the replacement.

Impact: An operator holding a still-valid update signature for a closed position can mutate and close a later, materially larger replacement that the trader never authorized the signature to affect. The PoC destroys all collateral of a five-times-larger replacement at an unchanged market price. No signature forgery is required, and the router exposes no nonce-cancellation path for the trader to invalidate the delayed request.

## RECOMMENDATION

Include the position timestamp and side, and preferably an expected TP/SL or incarnation version, in UpdateTpSlReq and reject mismatches against the current trade. Increment a version on slot reuse and expose nonce cancellation. Validate TP direction against the current position before storage, and bind guaranteed settlement to the authenticated incarnation and trigger.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-20

ACKNOWLEDGED

# Cross-mode raw-flow netting can turn an unfunded maker credit into same-price LP loss

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:\_decayVol():L780-L815 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:getDecayedVol():L818-L828 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfosExtension.sol:applySpread():L55-L121 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorage.sol:updatePairParams():L153-L170

REVIEW Main Review 1

## DESCRIPTION

The canonical accumulator nets raw notional across PnL and legacy flow even though current PairStorage assigns those modes independent impact/skew economics. Before a mode transition, an attacker opens a zero-impact PnL short. After transition, each legacy long is sized to the seed's exact next-block decayed value. applySpread sees near-zero finalVol and waives the configured 1.2% legacy impact, while the demonstrated official/configured mode-skew policy supplies the balancing maker quote. Closing the long restores a negative seed, allowing shrinking repetitions. The passing PoC reads current coefficients, reproduces the canonical exponential skew calculation from actual OI, advances every fulfillment two seconds, charges 0.1% open and close fees, and proves every cycle profitable at an unchanged oracle price. After full unwind the attacker gains 5,644.921331 USDC and tranche assets fall. The unresolved question is whether operator responsibility for off-chain spread inputs intentionally includes preventing this round trip despite canonical cross-mode netting.

Impact: If mode-specific coefficients are meant to make historical flow economically non-fungible, a visible fee-mode transition permits direct same-price extraction from ERC4626 LP liquidity using honest oracle data and configured operator parameters. Raw flow created under a zero coefficient becomes reusable credit against sensitive-mode impact even though it never funded that credit. The demonstrated \$1m seed and three decaying cycles net over \$5.6k; loss scales with OI, accounts, pairs, and cycles inside the flow half-life. If operators are instead intended to own all maker-rebate and anti-round-trip policy, this result is expected off-chain pricing behavior.

## RECOMMENDATION

Clarify and document the trust boundary. If canonical state must preserve mode-specific economics, partition or coefficient-weight the accumulator by mode, pass isPnl into applySpread, isolate old-mode flow during transitions, and validate submitted mode-specific parameters on chain. If operators intentionally own the policy, specify and test backend rules that prevent round-trippable maker credits across mode transitions.

---

**I -** Cross-mode raw-flow netting can turn an unfunded maker credit into  
**20** same-price LP loss **CONTINUED**

---

— **RESOLUTION**

Status: Acknowledged.

## INFORMATIONAL I-21

ACKNOWLEDGED

# Partial guaranteed stop losses settle at the gap price

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L166-L322 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executePartialTpSlCallback():L327-L404 packages/avantis-contracts-team2-1783453751962/src/Trading/LimitTradingModule.sol:\_transformTpSlIntent():L382-L410

REVIEW Main Review 1

## DESCRIPTION

On a pair with guaranteedSlEnabled, ordinary full SL settlement substitutes the persistent stop price for the gapped oracle price. The signed PARTIAL\_SL callback has no analogous branch: it ignores the pending order type and signed stop when selecting the settlement price and always applies spread to the current oracle price. The passing PoC opens identical 10x longs with \ \$48,000 stops and supplies an authentic, correctly triggering \ \$45,000 gap price. The ordinary full SL returns over \ \$40 from \ \$100 collateral, while an economically full-size signed partial SL settles at the gap and returns zero.

Impact: If guaranteed-stop protection is intended to cover signed partial stops, a normal oracle gap can wipe all collateral authorized for a partial stop even though an equivalent full stop receives protected-price settlement. At high leverage, a modest gap beyond the signed trigger can turn a bounded loss into total loss of the affected slice without oracle forgery or untriggered execution.

## RECOMMENDATION

If signed PARTIAL\_SL orders inherit guaranteed-stop protection, branch explicitly on PARTIAL\_SL and, for an ordinary loss-side stop on a guaranteed-SL pair, settle at pendingOrder.trade.sl rather than the spread-adjusted gap price. Preserve any intended profit-side/trailing exception and add proportional full-versus-partial equivalence tests.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-22

ACKNOWLEDGED

# Collateral increases can retroactively unlock previously capped PnL

CATEGORY Informational

LOCATION `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:registerPositionIncrease():L169-L225` `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:_computeNewOpenPrice():L432-L442` `packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateMarginCallback():L409-L501` `packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:currentPercentProfit():L309-L320`

REVIEW Main Review 1

## DESCRIPTION

maxGainP is reapplied to mutable live collateral and leverage without crystallizing profit accrued by earlier collateral. A mark-price increase preserves the seed's full raw dollar gain through the blended entry while enlarging its denominator, and updateMargin(DEPOSIT) can similarly lower leverage at fixed OI/open price. A trader can add transient collateral and close immediately, with no intervening price move, to realize PnL that a direct close immediately beforehand would cap. The PoC shows a 100-USDC seed directly limited to 500 USDC profit, while a 200-USDC mark-price increase unlocks roughly another 500 USDC despite the added leg earning zero PnL. Duplicate review confirmed this is independent of accepted rollover, oracle replay, legacy-funding, leverage-validation, and zero-SL findings: those affect pair-wide fee accrual, price freshness, funding populations, rejected above-cap deposits, or trigger sentinels, whereas this occurs with zero fees/funding, an honest current price, valid leverage, and irrelevant TP/SL solely because the per-position payout denominator changes.

Impact: If maxGainP is intended to bound profit earned by the collateral actually exposed during the move, transient added collateral can convert previously unpayable mark-to-market gain into an immediate LP-funded payout. The excess scales with seed OI and appreciation and is limited by exposure and vault liquidity. If the cap is intentionally defined only against mutable live collateral, this behavior is expected.

## RECOMMENDATION

Clarify and document the cap invariant. If profit must remain anchored to exposed collateral, crystallize capped PnL before every increase or margin deposit, or maintain per-lot/capped-accrual accounting. If mutable live-collateral repricing is intended, document that deposits can raise the dollar payout ceiling and size LP risk controls accordingly.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-23

ACKNOWLEDGED

# Per-holder rollback bypasses reserve limits before aggregate OI handoff

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:migrate():L399-L414 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:unSwallowReservedOI():L388-L392 packages/avantis-contracts-team2-1783453751962/src/AvantisTranche.sol:maxRedeem():L322-L328
REVIEW	Main Review 1

## DESCRIPTION

Rollback transfers each holder's previewRedeem assets into the legacy senior tranche and burns that holder's unified shares, but migrate() neither enforces the source reserve-adjusted withdrawal limit nor transfers a proportional portion of totalReserved. The entire swallowed reservation is returned only by a separate unSwallowReservedOI() transaction. A holder whose full redemption is rejected by the source utilization guard can therefore be migrated and immediately redeem the same value from the still-unreserved destination. Repeating this during the multi-transaction migration window extracts backing that neither tranche would permit once paired with its liabilities.

Impact: On the production fork, the normal reserve gate blocks the tested holder cohort, whereas the rollback route lets that same cohort remove more than \$18 million. After the aggregate handoff, even pooling every asset still present in both tranches produces utilization above the configured 90% ceiling. Active OI is therefore assigned to insufficient backing, socializing the deficit to slower LPs and risking failed or underfunded trader-profit settlement.

## RECOMMENDATION

Make rollback a globally locked transition across both tranches, preventing deposits, transfers, withdrawals, and redemptions until all assets and OI are handed over. Alternatively, migrate each holder's assets together with a conservative proportional reservation atomically and keep destination shares non-redeemable until finalization. Never expose migrated backing before its corresponding liabilities are effective.

## RESOLUTION

Status: Acknowledged.

HIGH **H-01**

RESOLVED

# Floored DCA entry mints long close PnL

CATEGORY Rounding

LOCATION `\src\TradingCallbacks\TradeRegistry.sol:_computeNewOpenPrice\`

REVIEW Remediation Review

## DESCRIPTION

Position increases replace the live entry with `_computeNewOpenPrice`. That helper intends to store the size weighted average cost: total USD notional divided by total coins ( $\sum \text{notional} / \sum (\text{notional} / \text{price})$ ). Every division truncates, and the truncated result is stored as `openPrice` and fed into the next increase as if it were exact.

```

`solidity
uint256 coinExposure =
existingNotional * 1e30 / existing.openPrice +
addedNotional * 1e30 / added.openPrice;
newOpenPrice = (existingNotional + addedNotional) * 1e30 / coinExposure;
`

```

Because `newOpenPrice` is floored, later reconstruction reports more coins than were paid for, and a long's stored entry sits below the true average cost of the fills. Close PnL uses that stored entry in `currentPercentProfit`:  $(\text{currentPrice} - \text{openPrice}) / \text{openPrice} * \text{leverage}$ . A long can therefore close in profit at a price where the fee net fills, valued at their actual execution prices, are flat or losing. `AvantisTranche` pays the difference. Shorts get the opposite bias (a worse entry), so the printer is directional.

The C-03 remediation introduced `_rebaseCoinOI` so a full close no longer consumes another trader's coin OI. The rebase only synchronizes pair coin OI to the same reconstructed, floored entry. It does not preserve the position's actual cost and coin basis, so direct close PnL remains wrong. This is a residual settlement loss distinct from C-03's shared coin OI and funding drain, even though both originate from the rounded entry.

Toy integer example with precision 1000, two equal \$100 legs at prices 4 and 5:

```

`plain text
coins booked = 100*1000/4 + 100*1000/5 = 25000 + 20000 = 45000
true average cost = 200 * 1000 / 45000 = 4.444...
stored openPrice = 200 * 1000 / 45000 = 4 (floor)
`

```

Closing at 4.444 is break even on the fills. The protocol treats entry as 4 and pays long profit. The next increase reuses 4, so the deficit compounds.

BONK pair 24 is a concrete production configuration for this failure. It was previously

**H-01** Floored DCA entry mints long close PnL

CONTINUED

listed and its configuration remains stored: 20x maximum leverage, a \ \$100 minimum notional, a \ \$448,580.78 pair OI cap with 50% available to longs, a 5 bp pair spread, and 30 second TWAP slices. The pair is currently delisted and in close only mode, so BONK cannot be attacked at this moment. However, delisting only flips `isPairListed` ; governance can restore the same feed and disable close only mode without a contract upgrade. The accounting would then become exploitable immediately, as would another similarly priced pair added under the same code.

The POC shows a BONK price of about \ \$0.000003185 used 2,242 minimum \ \$100 long legs at 20x, consuming \ \$224,200 of OI and \ \$11,210 of trader collateral. The exact fills lost 11.75 USDC, while the recursively rounded position reported 7,633.00 USDC of profit. After conservative 0.10% open and close fees totaling 443.91 USDC and a nonzero 5 bp spread, the ordinary market close left the trader 7,189.08 USDC net positive and reduced `AvantisTranche` by the payout. The live fee is lower at 0.045%. One TWAP intent can authorize this number of slices within the configured one day runtime, although every slice still requires normal operator fulfillment.

The result does not require oracle manipulation. Applying the same integer merge to 1,121 observed one minute BONK prices from August 2 to 3 produced 1,537.18 USDC of protocol PnL while the exact fills lost 3.95 USDC. After 221.95 USDC of conservative fees, the trader remained 1,315.22 USDC net positive. The setup requires many fulfilled legs, available side OI, and a suitable price path, but it causes direct, repeatable tranche principal loss under a previously supported production market.

**— RECOMMENDATION**

Maintain a canonical high precision coin exposure for each position and update it from each executed leg. Derive merged entry price and pair coin OI from that accumulator instead of reconstructing prior exposure from the rounded `openPrice` . Close settlement should use the conserved basis or otherwise ensure any remaining one-time rounding cannot produce material PnL.

A smaller contract change is to round `_computeNewOpenPrice` against the trader (ceil for longs, floor for shorts). That stops the vault from paying fake long PnL, but reconstructed coin OI then sits slightly below the fills, so funding share and maker/taker skew can drift and partial closes can leave dust coin OI. Conserved cost and coins remain the complete fix.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/444e908acb24c8b3281783761eca2819245eb020>

---

**H-01** Floored DCA entry mints long close PnL

---

CONTINUED

---

**— RESOLUTION**

Status: Resolved.

MEDIUM **M-01**

ACKNOWLEDGED

# Negative ratios disable withdrawal fee

CATEGORY Math

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/VaultManager.sol#L428-L446>

REVIEW Remediation Review

## DESCRIPTION

`VaultManager` stores `currentBufferRatio` as an `int256`, and the trusted keeper can legitimately report a negative value when marked-to-market trader liabilities exceed protocol backing. However, `getBufferRatio()` casts that signed value directly to `uint256`. For example, `-1` becomes `type(uint256).max`. `getCollateralFee()` compares this enormous value against the configured buffer thresholds. Every comparison fails and the function returns zero, treating the most impaired state like an exceptionally healthy vault. `AvantisTranche` consequently charges no collateral-health fee on withdrawals or redemptions.

## RECOMMENDATION

Keep buffer-ratio handling signed through the fee comparison, or explicitly map every negative ratio to the configured worst-health fee bucket.

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-02**

ACKNOWLEDGED

# Abandoned funding uses settlement-time OI

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L386-L402">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L386-L402</a>
REVIEW	Remediation Review

## DESCRIPTION

`_recordFundingAudit` attempts to exclude unclaimable funding by marking a settled fee as abandoned whenever either side's current coin OI is zero. This asks whether a counterparty exists at settlement time, although the funding was created over earlier blocks.

For example:

1. Only longs exist while 100 USDC of funding accrues. No short can receive it.
2. A short opens later. Its funding baseline starts at the current accumulator, so it cannot claim the historical 100 USDC.
3. The long settles while both sides now have OI.
4. `_recordFundingAudit` sees the new short and treats the historical payment as matched rather than abandoned.

`totalFundingPaid` therefore increases by 100 while `totalFundingReceived` remains zero, and `fundingOwed()` reports a 100 liability even though no trader can claim it. Simply opening the opposite side does not collect the old funding; it only changes how the audit code classifies it.

The result is also close-order dependent. If matched positions accrued funding and the payer closes first, its debit is recorded while both sides exist. After its OI is removed, the receiver closes and its negative fee may be added to `abandonedFundingFees`, leaving an incorrect residual liability after both positions are gone.

`VaultManager` subtracts `fundingOwed` from available liquidity. Phantom liabilities can therefore reserve funds unnecessarily, pull liquidity from the junior tranche, restrict buffer withdrawals, or interfere with trader payouts. Position opening and settlement are permissionless, so these orderings can occur naturally or be repeated deliberately.

## RECOMMENDATION

Classify funding as paired or unpaired when it accrues, not when a trade later settles. Store enough per-pair interval or epoch history for each trade to settle against the historical classification. If one-sided funding is not intended as protocol revenue, do not accrue trader-to-trader funding while receiving-side OI is zero. Do not classify a negative receiver credit as abandoned solely because the paying side has

**M-02** Abandoned funding uses settlement-time OI

CONTINUED

already closed. Add regression tests covering an opposing position opened after one-sided accrual, both close orders for matched positions, partial closes, and the invariant that no `fundingOwed` remains after every eligible receiver has settled.

**— RESOLUTION**

Status: Acknowledged.

MEDIUM **M-03**

ACKNOWLEDGED

# TWAP and RFQ closes use mutable fee mode

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L205-L253">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L205-L253</a> , <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L584-L588">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L584-L588</a>
REVIEW	Remediation Review

## DESCRIPTION

`closeTwapCallback` and the RFQ close path classify an existing position with the pair's current `isPnlOrderTypeAllowed` setting instead of the fee type stored for that position. If governance changes the pair setting after a position opens, a later TWAP or RFQ close can apply the wrong fee model, mischarging either the trader or the vault.

## RECOMMENDATION

After validating that the stored trade and timestamp still match, load `getPosType(state.trader, state.pairIndex, state.index)` and use it consistently for TWAP and RFQ close settlement and events. Keep the mutable pair setting only for opening the first slice, where no position index exists yet. Add a regression test that changes the pair setting between position opening and TWAP/RFQ closure.

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-04**

RESOLVED

## Queued callbacks bypass the pair roll freeze

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L882-L897">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L882-L897</a> , <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L199-L255">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L199-L255</a> , <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L324-L384">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L324-L384</a> , <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/LimitCallbackModule.sol#L414-L508">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/LimitCallbackModule.sol#L414-L508</a>
REVIEW	Remediation Review

### — DESCRIPTION

Standard market closes, position increases, and margin updates can be authorized before a pair roll begins, but their asynchronous callbacks neither recheck `isPositionRollOngoing` nor bind the request to a roll epoch. They can therefore change or delete a position and mutate collateral, fee snapshots, reserves, and open interest while the keeper is migrating that position's instrument basis.

### — RECOMMENDATION

Bind every asynchronous position mutation to the current pair roll epoch and recheck the live roll flag before callback-side accounting; cancel and refund requests whose epoch no longer matches.

### — RESOLUTION

Status: Resolved.

MEDIUM **M-05**

# Cross-pair forex hedges retroactively erase target rollover debt

CATEGORY	Informational
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getBlendedSkew():L624-L640 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getPendingAccRolloverFees():L650-L695 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:_updateOpenInterestUSDC():L723-L762
REVIEW	Remediation Review

## — DESCRIPTION

Forex rollover blends pair-local skew with global directional `usdOI`, but the pending accumulator applies the current blend to every block since the target pair's last checkpoint. Opening an auxiliary forex position checkpoints only that pair before mutating the global `usdOI` used by all forex targets. An attacker can therefore age a costly target, briefly add opposite global-USD exposure on another listing, and checkpoint the target after the endpoint skew crosses into a cheaper fee regime. Closing the auxiliary leg does not restore the erased accumulator. The enhanced PoC uses nonzero open/close fees and identical snapshots: authentic carry makes the target liquidatable through the real LIQ path at the unchanged authenticated price, whereas the cross-pair hedge changes target skew without changing its checkpoint and makes the same liquidation revert.

Impact: The attack defeats a valid solvency liquidation, permanently erases aged carry, and leaves the trader with more aggregate USDC after fully unwinding both positions and paying all auxiliary fees than the control liquidation. The PoC also proves VaultManager retains correspondingly less USDC. One global-USD hedge can affect multiple forex targets, including hard-capped pairs whose `pairMaxOI` does not respond to tranche liquidity.

## — RECOMMENDATION

Maintain time-integrated per-pair and global-USD skew accumulators and charge each elapsed interval using the skew then in effect. At minimum, checkpoint every affected forex pair before any trade mutates global `usdOI`; checkpointing only the auxiliary pair is insufficient.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/commit/f64c45a26e3a6dd39ad3211d2b57d0bab95e0c10>

## — RESOLUTION

Status: Reported during the fix review period.

MEDIUM **M-06**

ACKNOWLEDGED

# Preassigned codes bypass private approval

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L199-L235> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L315-L337> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L411-L465> (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L199-L235 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L315-L337 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L411-L465)

REVIEW Remediation Review

## DESCRIPTION

The user-controlled referral setters reject a code only when `isPrivate[code]` is already true. For an unregistered code, both `isPrivate` and `codeOwners` hold their zero-value defaults, so a trader can store a predictable campaign code before it has an owner.

A separate governance or handler transaction then marks it private. Neither step clears existing trader assignments or records owner approval. At settlement, `getTraderReferralInfo` resolves the code's current owner and `traderReferralDiscount` applies that owner's current tier without rechecking privacy or approved membership.

An attacker can therefore preassign a code before registration, retain it after the code becomes private, and use a tier intended for owner-approved traders.

## RECOMMENDATION

Consider to require a nonzero registered owner before either user-assignment path accepts a code. For private codes, track explicit per-trader approval tied to the current code owner or an approval epoch, and validate it whenever a discount or rebate is applied.

Also consider to provide an atomic register-as-private path, or invalidate unapproved preexisting assignments whenever a code becomes private or changes owner.

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-07**

ACKNOWLEDGED

# Coin intents exceed leverage caps after fees

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L138-L171> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L416-L450> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L118-L138> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L138-L171> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/IntentCallbackModule.sol#L416-L450> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L118-L138>)

REVIEW Remediation Review

## — DESCRIPTION

Coin-sized TWAP and RFQ open callbacks calculate effective leverage from the slice's fixed coin exposure and submitted collateral, then validate that pre-fee value against both the trader's stored

`defaultLeverage` / `maxLeverage` bounds and the pair's leverage bounds.

For a fresh position, the callback then calls `registerTradeWithCoinExposure`. On non-PnL pairs, registration the opening fee from the slice collateral and recalculates leverage against the smaller collateral so the fixed coin exposure is preserved. The returned `finalTrade` is stored without any post-registration leverage validation.

Consequently, a slice whose pre-fee leverage equals the trader or pair maximum is persisted above that maximum whenever a nonzero opening fee is charged.

This permits positions to exceed trader-authorized leverage and, for fresh slices submitted at the pair maximum, the protocol's pair leverage cap, increasing liquidation risk and defeating the configured risk boundary.

## — RECOMMENDATION

Consider to validate the final leverage.

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-08**

RESOLVED

# Borrow caps retroactively reprice positions

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L684-L722">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L684-L722</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L751-L772">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L751-L772</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L856">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L856</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L378">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L378</a> ( <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L684-L722">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L684-L722</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L751-L772">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L751-L772</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L856">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L856</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L378">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L378</a> )
REVIEW	Remediation Review

## DESCRIPTION

`setBorrowRateCaps` overwrites the live minimum and maximum borrowing-rate caps for both directions without preserving the previous values or establishing a new accrual baseline. When fees are later evaluated, `PairInfos.minMaxRollOverFee` reads the current cap and multiplies it by the position's complete `timeSinceOpen`. The cap therefore applies retroactively to time that elapsed before governance changed it.

## RECOMMENDATION

Make sure all affected borrow-fee state and accumulators are checkpointed before updating borrow-rate caps, so the new values apply only prospectively.

## RESOLUTION

Status: Resolved.

MEDIUM **M-09**

ACKNOWLEDGED

# Split opens reduce nonlinear spread

CATEGORY Gaming

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfosExtension.sol#L57-L123>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L748-L853>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L72-L108>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L199-L235>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfosExtension.sol#L57-L123>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L748-L853>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L72-L108>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L199-L235>

REVIEW Remediation Review

## DESCRIPTION

`applySpread` computes the nonlinear impact rate only from the post-fill cumulative volume, `sqrt(abs(finalVol) / bookDepth)`, and applies that endpoint rate to the entire opening notional. Each completed open or position increase then records its contribution in the cumulative flow accumulator. A trader can therefore divide one same-direction position into sequential opens or, more directly, repeated increases: the slices are priced at the lower intermediate endpoints even though they reach essentially the same final collateral, OI, and exposure.

Starting from zero flow, one order of size  $Q$  pays an impact component proportional to  $Q * \sqrt{Q}$ . Four equal fills pay only  $(1 + \sqrt{2} + \sqrt{3} + 2) / 8$ , approximately 76.8%, of that amount; as the number of slices grows, the ratio approaches 66.7%.

## RECOMMENDATION

Consider adjusting the flow-spread formula or settlement method so economically equivalent aggregate openings cannot obtain a lower weighted impact by being partitioned.

PROOF OF CONCEPT <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/guardian-pocs/test/guardian-pocs/SplitOpenSpreadPoC.t.sol>

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-10**

ACKNOWLEDGED

# Legacy loss protection covers added notional

CATEGORY	Gaming
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L218-L228">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L218-L228</a>
REVIEW	Remediation Review

## — DESCRIPTION

Fresh v2 trades receive `lossProtection=0`, but migrated trades can retain nonzero `TradeInfo.lossProtection`. `registerPositionIncrease` merges added collateral and OI into the existing trade and overwrites only `Trade`; it never clears or limits that `TradeInfo` field. Close settlement later reloads the unchanged tier and multiplies all negative PnL of the merged position, so the legacy rebate applies to newly added notional. Partial closes subtract collateral and OI without clearing the tier, allowing the owner to retain a protected anchor and repeat the cycle.

An owner can increase a protected position and open an equal opposite unprotected position, either on Avantis or another venue. The opposite position offsets the protected position's ordinary price profit and loss. If the protected position loses, loss protection reduces that loss while the hedge still realizes the full offsetting gain, leaving the rebate as profit. If the protected position wins, its gain offsets the hedge's loss. This creates a theoretically risk-free trade with respect to price direction before costs because adverse price exposure is hedged while only the losing protected leg receives a subsidy.

At Base block 50,295,491, an exhaustive canonical scan found 133 live positions held by 81 owners with `lossProtection > 0`.

## — RECOMMENDATION

Consider to prevent new notional from inheriting legacy protection. Either reject both position-increase variants for trades with nonzero `lossProtection` (this can be blocked by the operator), or track remaining protected OI separately.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/9e1f54f26194babe891302eda04bfcb874e8e352/test/guardian-pocs/LegacyLossProtectionInheritancePoC.t.sol#L59-L132>

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-11**

RESOLVED

# Increases rebase TP/SL at old leverage

CATEGORY Logical Error

LOCATION \src/TradingCallbacks/TradeRegistry.sol\, \registerPositionIncrease\

REVIEW Remediation Review

## DESCRIPTION

`TradeRegistry` computes the blended open price, leverage, and collateral in memory, then calls `normalizeTpSl` before writing that state. `normalizeTpSl` reloads the live trade from storage, so it still sees the old leverage. It uses that old leverage to measure the existing TP/SL profit percent, to convert the percent back into a price at the new open, and to run `correctTp` / `correctSl`. The merged trade is then stored with the new leverage, so the saved triggers are bounded against the wrong risk parameters.

```
``plain text
```

```
existingTrade.leverage = _computeNewLeverage(existingTrade, newTrade);
existingTrade.initialPosToken += newTrade.initialPosToken;
(existingTrade.tp, existingTrade.sl) = pairsStored.normalizeTpSl(
existingTrade.trader, existingTrade.pairIndex, existingTrade.index, existingTrade.openPrice
);
```

```
`
```

Because the same stale leverage is used in both directions, the helper keeps the old price ratio rather than the profit percent that will apply at the stored leverage. After a leverage up increase, a previously valid maximum loss stop can sit beyond `maxSIP` and beyond the liquidation band. After a leverage down increase, a take profit that was at pair maximum gain stays too close to entry.

Illustration: a 2x long at 50,000 with the configured 80% maximum loss stop at 30,000. Adding equal collateral at 150x raises final leverage above 70x, but the stored SL remains near 30,000. `currentPercentProfit` at the new leverage is below -80%. `correctSl` at the final leverage would place the stop much closer to entry. Liquidation can consume the position before the intended stop is reached. The same stale leverage path also rebuilds TP. Both collateral sized and coin exposure increases are affected.

## RECOMMENDATION

Derive the intended TP/SL profit percents from the old entry, leverage, and triggers, then reconstruct and bound them with the final blended open price and leverage before storage.

**M-11** Increases rebase TP/SL at old leverage

CONTINUED

Pair maximum gain and maximum loss must be enforced against the leverage that will be stored on the trade.

**— RESOLUTION**

Status: Resolved.

MEDIUM **M-12**

ACKNOWLEDGED

# Legacy positions pay no borrow fees

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L855">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L840-L855</a>
REVIEW	Remediation Review

## DESCRIPTION

A live cohort of positions predates the reuse of `Trade.positionSizeUSDC` as the borrowing-fee start timestamp and still stores that field as zero. `PairInfos.minMaxRollOverFee` returns `(0, 0)` for a zero timestamp. `_getTradeRollOverFee` then clamps the position's positive raw accumulator fee to the zero maximum, so the final borrowing fee is always zero.

At Base block 50,390,756, 208 active positions held by 27 owners across 13 pairs had this state, representing 117,436.599583 USDC of open interest and 12,211.498090 USDC of collateral. All 208 direct `getTradeRollOverFee` calls returned zero. About 97.6% of the affected OI is concentrated in USD/JPY. The waived fee is also omitted from liquidation-price calculations. At the validated price snapshot, two large USD/JPY shorts would already have been liquidatable using only their raw accumulated borrowing fees, while the live calculation still treated their fee as zero. Position increases, margin updates, and position rolls initialize the timestamp prospectively, but TP/SL changes and partial closes preserve the zero-fee state on the remaining position.

## RECOMMENDATION

Be aware and consider to apply an explicit one-time migration policy for every active zero-timestamp position so that they do not avoid paying those fees.

**PROOF OF CONCEPT** <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f4ae7f2b96b2f9a1bcd70fa0195a9244957b8e2a/test/guardian-pocs/LegacyZeroTimestampBorrowFeePoC.t.sol>

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-13**

ACKNOWLEDGED

# Margin withdrawals can manufacture retroactive guaranteed stops

CATEGORY	Logical Error
LOCATION	packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:_executeUpdateMargin():L195-L225 packages/avantis-contracts-team2-1783453751962/src/Trading/PositionUpdatesModule.sol:_updateSl():L565-L579 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L203-L233
REVIEW	Remediation Review

## DESCRIPTION

After a withdrawal callback raises leverage, `_executeUpdateMargin` replays the old SL through the correction-only path. `PairStorage` can clamp that stop toward the open price under the new leverage, but the corrected guaranteed stop is never checked against the oracle price already used for the margin update. The generic 80% validation also fails to represent tighter pair-specific maxSIP values.

Impact: A trader can create an already-triggered guaranteed stop on the favorable side of the authentic current price and have settlement use the manufactured stop rather than market price. The passing public-flow PoC demonstrates payout exceeding the position's actual mark-to-market equity, with the difference borne by `VaultManager`/tranche liquidity. The issue scales with eligible OI and can be hedged/repeated during adverse price moves.

## RECOMMENDATION

Compute the post-margin corrected TP/SL before committing the update and validate the corrected guaranteed stop against the same authenticated oracle price. Reject a long SL above current price or a short SL below it, and use the pair-specific maximum SL bound consistently.

## RESOLUTION

Status: Acknowledged.

MEDIUM **M-14**

ACKNOWLEDGED

# Early price-loss clamp lets funding receivers extract LP principal near liquidation

CATEGORY	Logical Error
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:currentPercentProfit():L313-L325 packages/avantis-contracts-team2-1783453751962/src/PairInfos/PairInfos.sol:getTradeValuePure():L920-L940 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/TradeRegistry.sol:unregisterTrade():L315-L402
REVIEW	Remediation Review

## — DESCRIPTION

currentPercentProfit floors leveraged price PnL at -100% before PairInfos combines it with signed funding, rollover, closing fees, and loss protection. For a funding-receiving position whose raw price loss is below -100%, this discards the excess adverse price loss and then adds the funding credit to the floored equity. A receiver can therefore remain just above the funding-adjusted liquidation boundary under the raw-equity formula yet receive many multiples of collateral on voluntary close. The passing full-stack PoC uses ordinary market opens, real funding checkpointing, and normal market-close settlement; it proves raw PnL below -900%, correct net equity below 16% of collateral, a payout above 7x collateral, and a decrease in tranche assets.

Impact: A sufficiently funded receiving-side position can realize the difference between its discarded raw price loss and the -100% floor against VaultManager and tranche liquidity. The discrepancy can be multiple times collateral, directly reducing LP principal and potentially impairing withdrawals and other trader payouts.

## — RECOMMENDATION

Preserve uncapped signed price PnL through loss protection, rollover, closing-fee, and funding calculations, and apply limited liability only to the final net trade value. Make liquidation and close settlement use the same final-net equity equation, with regression tests for raw PnL below -100% and both funding signs.

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-15**

RESOLVED

# Equal-time Pyth cache pinning creates retroactive guaranteed stop-loss insurance

CATEGORY	Logical Error
LOCATION	packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_pythCore Price():L289-L312 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_ifParseUpdateData():L638-L648 packages/onchain-base-type4/src/__onchain/pythupgradable-41c9e39574f40ad3/project:/contracts/pyth/PythSetters.sol:updateLatestPriceIfNecessary():L14-L27 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateSlCallback():L541-L565 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L192-L219
REVIEW	Remediation Review

## — DESCRIPTION

Pyth replaces a cached feed only when `publishTime` strictly increases, while distinct authenticated observations can share a `publishTime`. A searcher can preload a favorable equal-time observation. Because `UPDATE_SL` updates and then reads Pyth's global cache rather than parsing the operator-supplied feed, a faithful fulfillment can validate an already-crossed guaranteed stop against the pinned predecessor. Impact: An unprivileged cache preload can retroactively insure an incurred leveraged loss and shift the gap to VaultManager/LPs. The passing full-flow PoC shows identical honest operator calldata rejecting the stop without preload but storing it after preload; at the authentic price the 100x position has -100% PnL, yet guaranteed-SL settlement pays over 70% of collateral from VaultManager.

## — RECOMMENDATION

Directly parse and bind `UPDATE_SL` to the requested feed observation supplied in the fulfillment transaction. Reject cache/payload mismatches and preserve sufficient Pyth accumulator ordering to distinguish equal-publishTime observations.

## — RESOLUTION

Status: Resolved.

MEDIUM **M-16**

ACKNOWLEDGED

# Insolvent closes reclass fees as profit

CATEGORY	Logical Error
LOCATION	<code>\`src/PairInfos/PairInfos.sol\`, \`getTradeValuePure\`</code>
REVIEW	Remediation Review

## DESCRIPTION

The C-02 patch made partial close borrow and funding proportional to the closed slice, stopped resetting the remaining position's fee snapshot, and rejects a zero collateral close. `getTradeValuePure` was not changed.

When post fee equity is at or below the 15% liquidation cushion, that helper zeros the trader payout and rewrites the returned PnL as `fees - collateral`. If fees exceed the closed collateral, that rewrite is a positive number even though the trader received nothing and the position lost money.

```

plain text
int256 value = int256(collateral) + lossProtectedPnl - fees;
if (value <= (int256(collateral) * int256(100 - liqThreshold)) / 100) {
 value = lossProtectedPnl - pnl;
 lossProtectedPnl = fees - int256(collateral) + value;
}
return (value > 0 ? uint256(value) : 0, lossProtectedPnl, uint256(fees));
`

```

`unregisterTrade` trusts that sign. It calls `decrementPnlRewards` on a fake profit (floored at zero if the bucket is empty) and still `allocateRewards` of the full `borrowFees` into `totalRewards`. No matching USDC was collected. A partial close then only subtracts the slice in `registerPartialTrade`, so leftover fee is not taken from remaining collateral.

Illustration: close 10 USDC of collateral with 0 price PnL and 25 USDC of borrow plus close fees. Payout is 0. Reported PnL is +15. Settlement burns 15 of `pnlRewards` and credits 25 of `totalRewards`. `VaultManager.sendUSDCToTrader` treats `totalRewards` as reserved, so a later real payout can pull the shortfall from `AvantisTranche`.

## RECOMMENDATION

Returned close PnL must be the realized price PnL, never `fees - collateral`. Book only the USDC actually collected from the closed collateral as borrow or close fee rewards. If fees exceed that collateral, cap the collected amount and, on a partial close, charge any remainder against remaining collateral before the position stays open.

**M-16** Insolvent closes reclass fees as profit

CONTINUED

**— RESOLUTION**

Status: Acknowledged.

LOW **L-01**

RESOLVED

# ABIs misdeclare delegatecall-emitted events

CATEGORY Informational

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L379-L381>

REVIEW Remediation Review

## DESCRIPTION

`ITradingStorage` declares `RollAdjustmentDone` with four fields: the trade, new open price, old leverage, and margin fees. The delegatecalled `TradingStorageCoreExtension` instead declares and emits a five-field event that additionally contains signed `fundingFees`.

`TradingStorage.positionRoll` executes the extension through `delegatecall`, so the extension's five-field event is logged at the `TradingStorage` proxy address. Event signatures include all parameter types in `topic0`; consequently, the four-field and five-field declarations produce different topics rather than two decodings of the same event.

The ABI-generation manifest publishes the compiled `TradingStorage` artifact as `tradingStorageAbi`. That artifact inherits the stale four-field interface event and does not include the extension's five-field event. Indexers, migration monitors, and SDK consumers using the generated proxy ABI therefore watch for the wrong topic and can silently miss completed position rolls and the associated funding settlement data. The runtime event is emitted by `TradingStorageCoreExtension.sol:464` with `fundingFees`, while the proxy-facing interface at the linked location omits that field.

## RECOMMENDATION

Define one canonical `RollAdjustmentDone` event signature shared by `ITradingStorage`, `TradingStorage`, and `TradingStorageCoreExtension`, including `fundingFees`, then regenerate and verify the published proxy ABI. Add an ABI/event regression test asserting that the generated `TradingStorage` ABI contains the exact topic emitted through `delegatecall`. If backward compatibility is required, use an explicitly versioned event or intentionally emit both signatures during migration.

## RESOLUTION

Status: Resolved.

LOW **L-02**

RESOLVED

# Forced PnL closes apply the regular spread

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L668-L699">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L668-L699</a>
REVIEW	Remediation Review

## DESCRIPTION

`forceCloseTradeMarket` detects whether the stored position is PnL and creates a `MARKET_CLOSE_PNL` order, but `_forceClose` always calls `pairSpreadP(r.pairIndex, false)`. Passing `false` selects the ordinary `spreadP` instead of the configured `pnlSpreadP`. The resulting parameters set `byPass = true`, which bypasses flow and book-depth impact but still applies this constant spread. A legitimate risk-manager force-close of a PnL position therefore settles at the wrong execution price whenever the two spreads differ. The regular spread replaces, rather than stacks with, the PnL spread. A higher regular spread disadvantages the trader; a lower one disadvantages the vault. This path is restricted to the risk manager and is intended for manual emergency closure, such as when a Pyth feed is deprecated.

## RECOMMENDATION

Select the spread from the stored order type, for example `pairSpreadP(r.pairIndex, r.orderType == OrderType.MARKET_CLOSE_PNL)`. Add forced-close tests where `spreadP` and `pnlSpreadP` differ and assert that ordinary and PnL positions settle with their respective configured spreads.

## RESOLUTION

Status: Resolved.

LOW **L-03**

RESOLVED

## Zero-cap groups pass validation

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L56-L73>

REVIEW Remediation Review

### — DESCRIPTION

`Group.maxOpenInterestP` is a `uint256`, so the `groupOk` check `>= 0` always passes. Governance can therefore add or update a group with a zero cap. The same zero value is used by `groupListed` as the “not listed” sentinel, so after an active group is set to zero, `updateGroup` cannot restore it normally. For non-hard-capped pairs in that group, `pairMaxOI` becomes zero; if the pair has live open interest, rollover utilization divides by zero and closes or liquidations revert until governance applies a positive per-pair hard cap, migrates the pair, or upgrades. This is a governance-configuration footgun only: an unprivileged user cannot set the value.

### — RECOMMENDATION

Require `maxOpenInterestP > 0` in `groupOk`. If zero is intended to disable a group, store listed or enabled status separately and provide a direct reactivation path. Add regression tests for zero-valued `addGroup` and `updateGroup` calls and for recovery with live open interest.

### — RESOLUTION

Status: Resolved.

LOW **L-04**

ACKNOWLEDGED

# LimitExecuted emits price as limitIndex

CATEGORY	Events
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/LimitCallbackModule.sol#L259-L264">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/LimitCallbackModule.sol#L259-L264</a>
REVIEW	Remediation Review

## DESCRIPTION

`LimitExecuted` names its second field `limitIndex`. The limit-open path consistently emits `pendingLimit.index` there, allowing off-chain consumers to associate the event with the relevant order. The liquidation, TP, SL, and partial-close paths instead pass the callback's raw oracle `price` as the second argument. The actual execution price is already emitted separately in the event's `price` field (`values.price`). As a result, otherwise identical `LimitExecuted` logs use `limitIndex` with two incompatible meanings depending on whether the event represents an open or close. Indexers and monitoring systems following the declared event schema receive a price-sized fake order index on closes. They can fail to reconcile limit executions with pending orders or positions, produce broken lifecycle histories, and misreport close activity. The issue affects the corresponding emissions around lines 259, 301, and 391.

## RECOMMENDATION

Emit the real pending-order or trade index in `limitIndex` on every close path, consistent with the open path. If the raw oracle price is operationally useful, add a separately named event field instead of overloading the index. Add event tests for limit opens, liquidations, TP/SL closes, and partial closes that decode the log and assert each field's semantic value.

## RESOLUTION

Status: Acknowledged.

LOW **L-05**

ACKNOWLEDGED

## Final full withdrawal reverts at zero assets

CATEGORY	DoS
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L153-L155">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L153-L155</a>
REVIEW	Remediation Review

### — DESCRIPTION

`Tranche._withdraw` first performs the ERC-4626 withdrawal and transfers any fee, then unconditionally calls `utilizationRatio()` to validate the post-withdrawal state.

`utilizationRatio()` divides `totalReserved * _PRECISION * 100` by `vaultManager.adjustedBalance(totalAssets())`. When the last LP withdraws or redeems all remaining assets while `totalReserved == 0`, the post-withdrawal vault balance is zero. The check consequently evaluates `0 / 0`, triggering Solidity's division-by-zero panic.

The revert rolls back the entire withdrawal, including the transfers and share burn. Thus, although `maxWithdraw` and `maxRedeem` can advertise the owner's full position as available, the last LP cannot actually empty the vault. The vault retains an otherwise withdrawable residual until another party deposits or the implementation is changed.

### — RECOMMENDATION

Define utilization as zero whenever `totalReserved == 0`. Otherwise handle a zero adjusted asset balance explicitly before division and return or revert with intentional semantics. Prefer validating the computed post-withdrawal balance safely before transfers, and add regression tests for final `withdraw` and `redeem` calls with zero reserves, both with fees enabled and disabled.

### — RESOLUTION

Status: Acknowledged.

LOW **L-06**

ACKNOWLEDGED

## USDC blacklist strands roll limit orders

CATEGORY	DoS
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/LimitTradingModule.sol#L216-L225">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/LimitTradingModule.sol#L216-L225</a>
REVIEW	Remediation Review

### — DESCRIPTION

Base mainnet uses Circle USDC. `_cancelLimitOrder` transfers `positionSize + executionFee` to the trader before unregistering the order. If Circle blacklists the trader after order creation, that transfer reverts, so cancellation rolls back, the escrow and `openLimitOrdersCount` remain live, and the position-roll keeper also cannot clear the order because `cancelRollLimitOrders` shares this helper; completing a clean pair roll then requires governance intervention or unblacklisting.

### — RECOMMENDATION

Consider escrowing the residual amount in a claimable balance that the trader can withdraw later to safely navigate this edge case.

### — RESOLUTION

Status: Acknowledged.

LOW **L-07**

ACKNOWLEDGED

# Operator outage strands open-RFQ collateral

CATEGORY	DoS
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L510-L569">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L510-L569</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L198-L205">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L198-L205</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L339-L368">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L339-L368</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L105-L118">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L105-L118</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L158-L176">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L158-L176</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L363-L386">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L363-L386</a> ( <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L510-L569">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L510-L569</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L198-L205">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L198-L205</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L339-L368">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L339-L368</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L105-L118">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L105-L118</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L158-L176">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/interfaces/ITradingStorage.sol#L158-L176</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L363-L386">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L363-L386</a> )
REVIEW	Remediation Review

## DESCRIPTION

Opening an RFQ transfers the trader's collateral into `TradingStorage`, but the only RFQ cancellation functions require an authorized operator and are guarded by `whenNotPaused`. There is no trader/delegate cancellation path. Although `RfqOpenOrder` contains a signature deadline, `RfqState` does not persist an expiry and slice execution does not enforce one after the RFQ has been created. Therefore, if the router is paused or all operators are unavailable or revoked, an existing open RFQ cannot execute or be canceled. Its unused collateral ( `collateral - collateralUsed` ) remains escrowed and its `openLimitOrdersCount` slot remains occupied until governance restores both an operator and the unpaused state. This applies even to RFQs created normally while the protocol was operational.

## RECOMMENDATION

Consider making every RFQ automatically expire after a fixed on-chain duration. After expiry, slice execution should no longer be possible, and the trader or delegate should be able to cancel the RFQ and recover unused collateral while releasing the occupied order slot, including while the protocol is paused.

Permissionless refund-only cleanup after expiry can also be considered.

## RESOLUTION

LOW **L-08**

ACKNOWLEDGED

## Dust full closes retain maker fee tier

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L412-L426">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L412-L426</a>
REVIEW	Remediation Review

### — DESCRIPTION

`closeTradeMarketCallbackWithCoinExposure` converts `pendingOrder.coinExposure` to collateral and promotes `closeCollateral` to the full position when the remainder is below `COLLATERAL_DIFF_THRESHOLD` (0.01 USDC). Settlement then uses the promoted collateral, but `pairCloseFeeP` still receives the original sub-full `pendingOrder.coinExposure`. Because fee classification computes post-close OI from that stale exposure, a nominal dust remainder can select the maker tier even though settlement removes all pair OI. A trader holding all remaining OI on a pair can therefore request a dust-producing close and pay the maker rate on the full finalized notional instead of the sweep taker rate. The setup requires the trader to own all remaining pair OI and maker/taker fees to differ, limiting frequency and impact.

### — RECOMMENDATION

Consider recomputing the effective coin exposure after the collateral dust clamp and using that canonical exposure for `pairCloseFeeP` and emitted close accounting, so fee classification matches the OI actually settled.

### — RESOLUTION

Status: Acknowledged.

LOW **L-09**

ACKNOWLEDGED

# Legacy Trading retains cancellation access

CATEGORY	Configuration
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/754a6070dedd8ec13d9bedfa7ded69935510c7ea/src/Trading.sol#L562-L567">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/754a6070dedd8ec13d9bedfa7ded69935510c7ea/src/Trading.sol#L562-L567</a>
REVIEW	Remediation Review

## DESCRIPTION

After the August migration, the deprecated Trading proxy `0x5FF292d70bA9cD9e7CCb313782811b3D7120535f` remains authorized in the shared TradingStorage ( `isTradingContract(legacy) == true` ) even though `trading()` points to the replacement router `0x44914408af82bC9983bbb330e3578E1105e11d4e` . Pausing the legacy proxy does not remove this authority. Its `cancelPendingMarketOrder(uint256)` entrypoint is restricted only by the legacy operator mapping, has no pause check, and calls `TradingStorage.forceUnregisterPendingMarketOrder` . The legacy router currently retains 45 operators; its operator set has no overlap with the 250 operators on the replacement router. A read-only call from a legacy-only operator succeeds through the legacy router, while the same caller reverts with `OnlyOperator()` through the replacement. Consequently, a compromised or otherwise untrusted legacy operator can still cancel and refund conventional pending market open, increase, or close orders stored in the current shared TradingStorage. This enables order disruption and can leave users exposed while they retry. No direct theft path was demonstrated, and TWAP/Rfq intents use separate cancellation paths.

## RECOMMENDATION

Consider to revoke the deprecated router in TradingStorage with `updateTradingContract(legacy, false)` , remove any remaining authorization in VaultManager, and clear its operator set. Future router migrations should atomically deauthorize retired contracts and verify that paused legacy proxies cannot mutate shared production storage.

## RESOLUTION

Status: Acknowledged.

LOW **L-10**

ACKNOWLEDGED

# Signed increases override pending full exits

CATEGORY	Warning
LOCATION	<code>\src\Trading\PositionUpdatesModule.sol\`, \`_increasePositionSize\`, \`_increasePositionSizeWithCoinExposure\`</code>
REVIEW	Remediation Review

## DESCRIPTION

A signed position increase can remain queued with the operator while the trader later requests a full market close. The close sets `beingMarketClosed`, but both shared increase-request functions ignore that flag. The operator can therefore submit the still-valid signed increase after the newer close request.

`_increasePositionSize` escrows the added collateral and, through `executePositionUpdateBatched`, fulfills the increase synchronously.

```
```plain text
```

```
ITradingStorage.Trade memory existingPos = storageT.openTrades(
    _updateInfo.trader,
    _updateInfo.pairIndex,
    _updateInfo.index
);
```

```
// No beingMarketClosed check before collateral is transferred.
```

```
storageT.transferUSDC(_trader, address(storageT), _updateInfo.initialPosToken);
```

```
``
```

The pending close stores only the collateral that existed when the trader requested the exit. After the delayed signed increase enlarges the position, the close callback removes only that stored amount and treats the full-exit request as a partial close.

For example, a trader has 100 USDC of collateral at 10x and signs an increase adding 900 USDC. Before the operator submits it, the trader requests a full close of the live 100 USDC. The operator then executes the still-valid signed increase: 900 USDC is unexpectedly pulled from the trader's wallet and the position collateral rises to 1,000 USDC. The close callback subsequently removes only its snapshotted 100 USDC, leaving the newly pulled 900 USDC locked in the trade at 10x, or 9,000 USDC of unintended exposure. The trader can continue incurring price losses, funding, rollover fees, or liquidation risk despite having submitted a newer full-exit request.

RECOMMENDATION

Reject both collateral-based and coin-exposure increase requests when the exact trade's

`beingMarketClosed` flag is set, before transferring collateral.

L-10 Signed increases override pending full exitsCONTINUED

— RESOLUTION

Status: Acknowledged.

LOW **L-11**

Wrapped short SL becomes immediately active

CATEGORY	Unexpected Behavior
LOCATION	<code>\src\Trading\MarketTradingModule.sol\`: \_openTrade\`, _openTradeWithCoinExposure\`</code>
REVIEW	Remediation Review

— DESCRIPTION

Market, limit, and coin exposure opens validate a short SL only by checking that it is above the requested open price. They do not reject values that cannot fit in `int256`, so a trader can open a short with `sl = type(uint256).max`. When `registerTrade` calls `correctSl`, `currentPercentProfit` casts that value to `int256(-1)`, treats the malformed stop as highly profitable, and stores it unchanged.

```
```plain text
```

```
if (!(trade.sl == 0 || (trade.buy ? trade.sl < trade.openPrice : trade.sl > trade.openPrice)))
 revert Errors.WrongSl();
```

```
int256 diff = buy
? (int256(currentPrice) - int256(openPrice))
: (int256(openPrice) - int256(currentPrice));
```
```

The malformed stop has concrete downstream effects. For example, a 10x short opened at 50,000 with `sl = type(uint256).max` stores that value. If the trader later increases the position at the same price, `normalizeTpSl` interprets the wrapped stop as maximum profit and rewrites it to 25,000 under the local pair configuration. The market remains at 50,000, which already satisfies the short SL predicate `price >= sl`, so an honest keeper can immediately close the position and charge exit fees. A focused test produced a 1 USDC loss on an otherwise flat price path. Margin deposits and withdrawals also revert with `IncorrectSl` because `updateMargin` revalidates the stored value after fulfillment, forcing the trader to clear or replace the SL before changing margin.

— RECOMMENDATION

Reject TP and SL values greater than or equal to `type(int256).max` at every open and update boundary. Range check unsigned prices before signed conversion, and ensure normalized stops remain on the valid, untriggered side of the current price.

L-11 Wrapped short SL becomes immediately activeCONTINUED

— RESOLUTION

Status: Reported during the fix review period.

LOW **L-12**

RESOLVED

Newer Pyth updates can flip partial TP's into losses

CATEGORY Informational

LOCATION packages/avantis-contracts-team2-1783453751962/src/Trading/LimitTradingModule.sol:executePartialTpSl():L259-L302 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_pythCorePrice():L289-L312 packages/avantis-contracts-team2-1783453751962/src/PriceAggregator.sol:_ifParseUpdateData():L638-L648 package s/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executePartialTpSlCallback():L325-L408

REVIEW Remediation Review

— DESCRIPTION

A faithful operator can broadcast executePartialTpSl with an authentic trigger-crossing Pyth update, but LIMIT_PARTIAL_CLOSE is not directly parsed. PriceAggregator updates Pyth and then reads the mutable globally latest price, so an unprivileged searcher/builder can publish a newer authentic rebound update before the operator transaction. The callback then settles at the rebound price because it never revalidates the signed partial TP/SL threshold.

Impact: A same-block oracle updater can convert a correctly submitted signed take-profit into an adverse full-position close, realizing losses instead of profit and still crediting the executor reward. This does not require oracle compromise or strategic operator timing; it follows from permissionless Pyth cache mutation before inclusion.

— RECOMMENDATION

Directly parse the operator-supplied Pyth update for LIMIT_PARTIAL_CLOSE and independently enforce the signed side-aware TP/SL trigger in executePartialTpSlCallback before settlement and reward distribution.

— RESOLUTION

Status: Resolved.

LOW **L-13**

ACKNOWLEDGED

Zero-amount market closes cannot execute

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L488-L538> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L320> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L488-L538> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/TradeRegistry.sol#L315-L320>)

REVIEW Remediation Review

— DESCRIPTION

`_closeTradeMarket` accepts `_amount == 0`, forwards any execution fee, stores a pending close, and sets `beingMarketClosed`. With a valid nonzero oracle price, the callback reaches `TradeRegistry.unregisterTrade`, which reverts `ZeroAmount` before the pending order can be removed. The trader cannot submit another market close or margin update until an operator force-cancels the order, and the execution fee is not refunded.

— RECOMMENDATION

Reject zero collateral before forwarding the execution fee or creating the oracle request, and add a focused test that zero-amount closes leave no pending ID or `beingMarketClosed` flag.

— RESOLUTION

Status: Acknowledged.

LOW **L-14**

ACKNOWLEDGED

getTradeValuePure wraps negative net fees

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L920-L940 |
| REVIEW | Remediation Review |

— DESCRIPTION

`getTradeValuePure()` combines unsigned rollover and closing fees with the signed funding fee in an `int256`. A legitimate funding credit can exceed the unsigned charges and make the aggregate `fees` negative, but the function unconditionally returns `uint256(fees)`. Solidity therefore represents the negative amount modulo 2^{256} ; for example, a net fee of `-1` is exposed as `$2^{256} - 1$` .

— RECOMMENDATION

Consider returning the net fee as `int256`, or expose separate unsigned charge and credit fields, and document their sign conventions.

— RESOLUTION

Status: Acknowledged.

LOW **L-15**

ACKNOWLEDGED

Stale consent overwrites referral choice

| | |
|----------|---|
| CATEGORY | Gaming |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Referral.sol#L232-L263 |
| REVIEW | Remediation Review |

— DESCRIPTION

A trader can request a private referral code and later select a different public code, but the earlier request remains stored. The old private-code owner can subsequently approve that stale request and overwrite the trader's newer selection without renewed consent, changing the applicable referral discount and rebate recipient for later trades.

— RECOMMENDATION

Be aware of this behavior and consider clearing outstanding private-code requests whenever the trader changes or relinquishes their referral code.

— RESOLUTION

Status: Acknowledged.

LOW **L-16**

RESOLVED

ERC-4626 limits ignore the deposit cap

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L23-L34> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L271-L282> (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L23-L34 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L271-L282)

REVIEW Remediation Review

DESCRIPTION

`Tranche` enforces `depositCap` only in `_deposit`, reverting when `totalAssets() + assets >= depositCap`, but does not override the inherited ERC-4626 `maxDeposit` and `maxMint` getters, which report `type(uint256).max`. Integrations relying on these mandatory limit getters can therefore size deposits or mints that the tranche rejects near its cap.

RECOMMENDATION

Consider overriding `maxDeposit` and `maxMint` to expose the tranche's remaining admissible capacity using the same boundary enforced by `_deposit`.

RESOLUTION

Status: Resolved.

LOW **L-17**

ACKNOWLEDGED

SL correction can install a retroactive guaranteed stop across the current market

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:updateSlCallback():L534-L557 packages/avantis-contracts-team2-1783453751962/src/TradingStorage/TradingStorageCoreExtension.sol:_updateSl():L501-L511 packages/avantis-contracts-team2-1783453751962/src/PairStorage/PairStorageExtension.sol:correctSl():L293-L298 packages/avantis-contracts-team2-1783453751962/src/TradingCallbacks/LimitCallbackModule.sol:executeLimitCloseOrderCallback():L178-L205 |
| REVIEW | Remediation Review |

DESCRIPTION

When guaranteed SL is enabled, updateSlCallback validates the authenticated current price against the uncorrected requested SL. TradingStorage then applies correctSl(), which can clamp an extreme request toward entry and across the already-observed market, without returning or revalidating the corrected value. Guaranteed-SL settlement later values the close at that stored historical stop.

Illustration: 100 USDC, 10x long, open 100, `maxSlP = 20`. At oracle 93 the trader requests 92. The 80% bound and `92 <= 93` both pass. `correctSl` stores 98. Honest SL execution at 93 settles at 98 (~80 USDC pre-fee) instead of mark (~30 USDC).

This pays only if `guaranteedSlEnabled` is on and `maxSlP < 80`. Shipped config has used `maxSlP = 80` on every pair checked, including v1.5 pairs that had guaranteed SL on. At 80 the two formulas agree at the boundary (`correctSl` is strict `<`), so the rewrite does not occur. v2 currently has guaranteed SL off. The mismatch is still live in code and becomes payable if those two knobs are ever set together.

RECOMMENDATION

Correct the requested SL before validation, validate the exact corrected value against the current oracle price, and store and emit only that validated value. Reject corrected long SLs above current price and corrected short SLs below it.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-01

ACKNOWLEDGED

Declared output is never returned

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L601-L607 |
| REVIEW | Remediation Review |

DESCRIPTION

`TradingRouter.delegatedAction()` declares `returns (bytes memory)` and stores the successful module result in `ret`, but after checking `if (!ok) _bubble(ret);` it reaches the end without returning anything. Successful delegated calls therefore change state while the caller receives empty bytes instead of the returned value.

RECOMMENDATION

Return `abi.decode(ret, (bytes))` after the successful delegatecall and add a test asserting that the caller receives the returned bytes.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-02

ACKNOWLEDGED

Rollover multiplier drops at full utilization

CATEGORY Math

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L768-L796>

REVIEW Remediation Review

DESCRIPTION

`_getRolloverMultiplier` uses the near-singular formula $\text{product} / (100\% - \text{product})$ until utilization times skew reaches 100%, then replaces the product with a 90%-utilization fallback. At 100% skew, 99% utilization therefore produces a 99x multiplier, while exactly 100% utilization produces only 9x. The per-trade maximum limits the final fee, but the accrual curve remains discontinuous and non-monotonic. A small increase in utilization caused by OI, TVL, or cap changes can lower future rollover accrual, while traders immediately below the boundary can be charged the configured maximum.

RECOMMENDATION

Clamp the utilization-skew product to a safe maximum before division so the multiplier is bounded, continuous, and never decreases as utilization rises.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-03

ACKNOWLEDGED

Intent closes permit stale settlement

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/RfqTradingModule.sol#L325-L336 , https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TwapTradingModule.sol#L403-L418 , https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L283-L304 , https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L388-L410 , https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingCallbacks/MarketCallbackModule.sol#L389-L447 |
| REVIEW | Remediation Review |

DESCRIPTION

Computing the fee amount from clamped live close collateral fixes the original oversized rebate, but RFQ/TWAP validation relies on beingMarketClosed even though their close-intent storage does not set that flag. A second close can therefore stale the pending request, while pairCloseFeeP still uses its original coin exposure and can select the wrong maker/taker tier.

RECOMMENDATION

Use one close-lock across market, TWAP, and RFQ lifecycles; bind execution to the live remaining trade exposure; and pass the finalized canonical exposure to both fee-tier selection and storage settlement.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-04

ACKNOWLEDGED

Installed Pyth Lazer parser rejects timestamp

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L720-L727>, <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L739-L747>, <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L779-L809>, <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/lib/pyth-crosschain/lazer/contracts/evm/src/PythLazerLib.sol#L103-L113>

REVIEW Remediation Review

DESCRIPTION

PriceAggregator requires FeedUpdateTimestamp property ID 12, while the currently installed PythLazerLib build rejects every property ID above 8 with "Unknown property" before Avantis can parse it. The dependency is ignored rather than pinned at audited HEAD, so this build would make compliant Lazer payloads unavailable even though it fails closed against stale prices; the actual deployed library linkage was not supplied.

RECOMMENDATION

Pin a Pyth Lazer revision whose enum and parser both accept FeedUpdateTimestamp, compile and test the exact linked deployment bytecode, and make that dependency lock part of the reviewed release artifact.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-05

ACKNOWLEDGED

Risk-manager rotation script omits aggregator

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/script/testnet/wire.s.sol#L69-L75> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L74-L75> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L107-L115> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L593-L632> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/script/testnet/wire.s.sol#L69-L75> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L74-L75> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L107-L115> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L593-L632>)

REVIEW Remediation Review

DESCRIPTION

`PriceAggregator` stores a separate `riskManager` and authorizes `forceCloseTradeMarket` against that local role. The documented testnet wiring script rotates only `TradingStorage.riskManager` and does not call `PriceAggregator.setRiskManager`. If operators rely on this script for a risk-manager rotation without an additional aggregator update, the former aggregator manager remains authorized to force-close positions at a supplied price until governance separately updates the aggregator. The aggregator role can be upgraded through `setRiskManager`; the issue is that the wiring path does not update both copies.

RECOMMENDATION

Be aware of this behavior and consider updating the wiring script to rotate both risk-manager slots.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-06

RESOLVED

Multicall rollover quotes diverge from fees

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L687-L725 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L777-L797 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Multicall.sol#L601-L719 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L862-L875 (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L687-L725 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L777-L797 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Multicall.sol#L601-L719 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L862-L875) |
| REVIEW | Remediation Review |

DESCRIPTION

`PairInfos` and `Multicall` independently implement the rollover multiplier but disagree at utilization boundaries. Settlement uses the dynamic branch when utilization equals the configured threshold ($<$), while `Multicall` returns the default multiplier at equality ($\leq$). More importantly, when `utilization * skew  $\geq 1e12$` , settlement recomputes the product using 90% utilization and can return a `9e10` multiplier at 100% skew, whereas `Multicall` abandons the dynamic formula and returns `pairBorrowFeesMultiplier`. Since pair OI may validly equal `pairMaxOI`, this state is reachable. Consequently, public margin and dynamic snapshot endpoints can display a materially lower rollover rate than the accumulator used to settle existing positions. The annual fee cap does not necessarily eliminate the difference, particularly after a lower nonzero direction-specific base rate increases the permitted multiplier.

RECOMMENDATION

Make `Multicall` use the same rollover-multiplier routine, or a shared library, as `PairInfos`, including identical threshold and `product  $\geq 1e12$` handling. Add parity tests for both directions at the utilization threshold and at or above full utilization after direction-specific base-rate updates.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-07

ACKNOWLEDGED

Funding updater strands keeper oracle fees

CATEGORY DoS

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L1121-L1134> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L167-L175> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L289-L320> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/test/units/Funding.t.sol#L677-L680> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L1121-L1134> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L167-L175> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L289-L320> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/test/units/Funding.t.sol#L677-L680>)

REVIEW Remediation Review

DESCRIPTION

`PairInfos.updateFundingAccumulators` is payable, but its batch loop calls `PriceAggregator.validatePrice` without forwarding any value. For `PYTH_CORE` and `PYTH_LAZER`, `PriceAggregator` nevertheless pays the oracle fee from its own ETH balance. Consequently, ETH attached by the funding keeper remains in `PairInfos` with no refund or withdrawal path, while the update succeeds only if `PriceAggregator` was separately prefunded; otherwise the Pyth call and the complete batch revert.

RECOMMENDATION

Consider to forward `msg.value` to `PriceAggregator` once for the complete batch, require it to cover the total oracle fees, and refund any excess instead of relying on a separately prefunded aggregator balance.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-08

ACKNOWLEDGED

Zero-fee orders operator cleanup

| | |
|----------|--|
| CATEGORY | Gas Griefing |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L172-L184
https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L220-L245
https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L407-L414 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L184-L219 (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L172-L184 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/MarketTradingModule.sol#L220-L245 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Trading/TradingRouter.sol#L407-L414 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageIntentExtension.sol#L184-L219) |
| REVIEW | Remediation Review |

— DESCRIPTION

Trader-callable market-order creation accepts any `msg.value`, including zero, and only checks whether forwarding that value to the configured execution-fee receiver succeeds. The order is still created and the trader's USDC collateral is escrowed without any minimum keeper fee. If the order is not executed, only an authorized operator can call `cancelPendingMarketOrder`; that cleanup returns the trader's collateral but provides no compensation for the operator's cancellation gas.

— RECOMMENDATION

Be aware of this behavior and consider to document it, or consider to add a minimum ETH value validation.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-09

ACKNOWLEDGED

Funding updates bypass backup-only mode

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L129-L175 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L236-L260 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L1121-L1134 (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L129-L175 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L236-L260 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L1121-L1134) |
| REVIEW | Remediation Review |

DESCRIPTION

`PairInfos.updateFundingAccumulators` obtains each funding price through `PriceAggregator.validatePrice`. That function invokes `_validatedPrice(..., false)`, while `_validatedPrice` enters backup-only handling only when both `allowBackupOnly` and the governance-controlled `useBackupOnly` flag are true. Funding updates therefore ignore an active backup-only mode and continue accepting Pyth Core, Pyth Lazer, or Avantis input through the normal primary-source path. If governance enabled backup-only mode because the primary source is unreliable, the trusted funding keeper could still advance the pair's funding accumulators with a price from that source.

RECOMMENDATION

Be aware of and document this behavior.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-10

RESOLVED

updatePair can zero a live pair cap

CATEGORY DoS

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L240-L270> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L492-L498> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L650-L695> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L240-L270> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairStorage/PairStorage.sol#L492-L498> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfos.sol#L650-L695>)

REVIEW Remediation Review

DESCRIPTION

`updatePair` does not apply the `groupListed` validation used by `addPair` and does not require `groupOpenInterestPercentageP` to be positive.

Governance can therefore move a non-hard-capped pair to an unlisted or zero-cap group, or set the pair's group allocation to zero. In each case `pairMaxOI` becomes zero.

When the pair has live OI, `PairInfos.getBlendedUtilizationRatio` divides that OI by `pairMaxOI`, causing rollover-fee materialization and settlement paths that depend on it to revert.

RECOMMENDATION

Consider to add a simple `> 0` validation in `updatePair` for the destination group's effective OI capacity, including `groupOpenInterestPercentageP`, before storing the update.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-11

ACKNOWLEDGED

Permissionless validation spends contract ETH

CATEGORY Access Control

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L167-L175> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L315-L320> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L167-L175> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PriceAggregator.sol#L315-L320>)

REVIEW Remediation Review

DESCRIPTION

`PriceAggregator.validatePrice` has no caller restriction, allowing anyone to trigger Pyth validation fees paid from the contract's ETH balance.

RECOMMENDATION

Consider adding access control to `validatePrice` so only authorized protocol callers can invoke it.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-12

RESOLVED

Migration traps live FET/USD positions

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/script/testnet/migration.s.sol#L135-L185 |
| REVIEW | Remediation Review |

DESCRIPTION

The production migration explicitly skipped deprecated pair index 28 (FET/USD) while four live long positions still existed on that pair. At Base block 50,386,268, those positions were held by three owners and represented 994.856800 USDC of leveraged open interest backed by 308.594527 USDC of collateral. The positions remain present in canonical TradingStorage and aggregate pair/global OI, but their migrated wallet OI and coin OI are both zero and their fee migration flags were not initialized. The production migration calldata contains no pair-28 wallet rows; the old aggregate wallet OI exceeds the migrated total by 994.856816 USDC, matching the omitted FET/USD exposure within 16 micro-USDC. The pair is now delisted, so a normal close request reverts with `PairNotListed`. Even if the close reaches storage cleanup, removing the position from the zero migrated wallet/coin OI buckets underflows with panic `0x11`. Liquidations use the same removal accounting. Consequently, the four positions are trapped: owners cannot close them and the protocol cannot liquidate them through the normal paths, while aggregate accounting continues to report their exposure.

RECOMMENDATION

Be aware about that and consider to think about how to get them out of the system.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-13

ACKNOWLEDGED

Margin deposits realize unrelated fee state

| | |
|----------|--|
| CATEGORY | Warning |
| LOCATION | \`src/Trading/PositionUpdatesModule.sol`, \`updateMargin`, \`_executeUpdateMargin` |
| REVIEW | Remediation Review |

DESCRIPTION

`updateMargin(DEPOSIT)` is not limited to adding collateral. It synchronously requests a price, calculates accrued borrow and signed funding fees, incorporates them into collateral, records funding settlement, and resets both fee snapshots. Negative funding therefore increases live collateral during a deposit:

```
``plain text
if (_type == ITradingStorage.updateType.DEPOSIT) {
    signedCollateral = int256(_currentCollateral)
    + int256(_newAmount)
    - int256(_borrowfees)
    - _fundingFees;
}
```

This makes a permissionless, risk reducing action a general oracle and fee settlement checkpoint. Any upstream oracle, coin OI, or funding calculation defect can be capitalized into later withdrawable collateral through this route, as occurred in prior Critical findings. The current requirement that a deposit cover positive fees prevents nominal deposits from reducing collateral, but it does not constrain negative funding credits.

No new malformed funding calculation is asserted here. This change would limit the damage from future accounting bugs. A margin deposit should only add collateral. It should not also settle funding.

RECOMMENDATION

Wherever funding is realized, apply a cumulative funding credit circuit breaker derived from position notional, elapsed blocks, and a configured maximum credit rate. Quarantine excess credit instead of capitalizing it so an anomalous calculation cannot immediately become withdrawable collateral.

Another fix to consider is to make deposits pure collateral operations: require a positive transfer, increase collateral exactly by the USDC received, preserve OI, and leave borrow and funding accumulators and trade snapshots unchanged. However, funding currently

I-13 Margin deposits realize unrelated fee state**CONTINUED**

derives coin size from a reconstructed value. If the funding snapshot stays unchanged, the new rounded coin size would be applied to funding that accrued before the deposit. A pure deposit therefore needs a canonical position size. Funding and borrow fees should use stored OI or stored coin exposure, not reconstruct it from collateral and leverage.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I - 14

ACKNOWLEDGED

Dust margin updates erase fee fractions

CATEGORY Rounding

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/444e908acb24c8b3281783761eca2819245eb020/src/Trading/PositionUpdatesModule.sol#L161-L179>

REVIEW Remediation Review

— DESCRIPTION

`updateMargin` accepts a deposit of one USDC base unit. When accrued borrowing or positive funding fees are below one base unit, integer division returns zero, but the callback still updates the trade timestamp and stores new fee baselines, permanently discarding those fractions. Rollover accumulator checkpoints can similarly discard sub-unit remainders pair-wide. The impact per update is negligible and normally smaller than execution costs; this is an awareness item rather than a material security risk.

— RECOMMENDATION

Be aware of this rounding behavior and consider raising the minimum margin-deposit amount so dust deposits cannot reset fee checkpoints. No urgent remediation is required.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-15

ACKNOWLEDGED

Inactive fulfillments retain native value

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/444e908acb24c8b3281783761eca2819245eb020/src/PriceAggregator.sol#L180-L228 |
| REVIEW | Remediation Review |

DESCRIPTION

`PriceAggregator.fulfill` and `fulfillIntents` are payable, but both return successfully when `orders[orderId].initiated` is false. Any native value forwarded by Trading for an unknown or already-cleared order therefore remains in the aggregator without a refund.

RECOMMENDATION

Be aware and consider reverting in that case.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-16

ACKNOWLEDGED

Protocol rounding in favor of the user

| | |
|----------|--------------------|
| CATEGORY | Rounding |
| LOCATION | GLOBAL |
| REVIEW | Remediation Review |

— DESCRIPTION

Multiple settlement and pricing paths round in the trader's favor:

- Open, close, PnL, and EIP-7702 percentage builder fees use floor division, while maker/taker fee blends are floored before the final charge.
- Utilization and skew are floored before carrying-cost thresholds and blends, allowing slightly above-threshold exposure to remain on the lower base multiplier; hedging-fee thresholds inherit a similar lower-bound bias.
- Signed PnL division truncates toward zero, making losing percentages and realized losses less negative; loss protection and margin-withdrawal checks compound or consume the same favorable rounding.
- Execution pricing floors the volume-to-depth ratio, square-root impact, spread, and final price delta, while signed volume decay truncates toward zero, understating positive price impact on opens and closes.

— RECOMMENDATION

Be aware and consider to always round against the user.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-17

ACKNOWLEDGED

Position rolls distort flow-based price impact

| | |
|----------|---|
| CATEGORY | Logical Error |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L401-L464 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L748-L853 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfosExtension.sol#L57-L123 (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L401-L464 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/TradingStorage/TradingStorageCoreExtension.sol#L748-L853 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/PairInfos/PairInfosExtension.sol#L57-L123) |
| REVIEW | Remediation Review |

DESCRIPTION

`positionRoll` rebases USD open interest while preserving coin exposure, but passes the rebase delta through `_updateOpenInterestUSDC`, which also updates the signed cumulative flow-volume accumulator. The roll is therefore treated as real trading activity. The synthetic flow persists until decay or offset: opens in the same direction can pay excess price impact, while opens against it can pay reduced impact, potentially across multiple trades.

RECOMMENDATION

Be aware about that and document this behavior.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-18

ACKNOWLEDGED

maxWithdraw overstates fee-bearing exits

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L182-L185> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L331-L338> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L182-L185> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/AvantisTranche.sol#L331-L338>)

REVIEW Remediation Review

DESCRIPTION

`maxWithdraw` returns the owner's gross asset balance, but `previewWithdraw` adds the withdrawal fee before converting assets to shares. With fees enabled, withdrawing the advertised maximum can require more shares than the owner holds and revert, so ERC-4626 integrations cannot rely on the limit getter.

RECOMMENDATION

Be ware and document this behavior or consider to return the largest net asset amount whose fee-inclusive `previewWithdraw` share requirement fits the owner's balance and the live utilization limit.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-19

ACKNOWLEDGED

Lowered cap hides high-index positions

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/f729f9e9a4a437498dcbb804675348189bb5520c/src/Multicall.sol#L87-L102>

REVIEW Remediation Review

DESCRIPTION

`setMaxTradesPerPair` can lower the current cap without moving or deleting existing trades. `Multicall.getPositions()` uses that current cap as its scan bound, so positions stored at older indices greater than or equal to the cap remain on-chain and directly accessible, but are omitted from this helper's response. UIs or automation relying on this helper may therefore fail to display those positions.

RECOMMENDATION

Be aware of this behavior and consider adjusting `getPositions()` so it returns all existing positions, including historical indices above the current maximum.

RESOLUTION

Status: Acknowledged.

HIGH **H-01**

RESOLVED

Coin close wanted price inflates fee

| | |
|----------|---|
| CATEGORY | Business logic errors |
| LOCATION | https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L486-L498 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/MarketTradingModule.sol#L630-L654 (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L486-L498 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/MarketTradingModule.sol#L630-L654) |
| REVIEW | Main Review 2 |

DESCRIPTION

The 7702 template prices a coin exposure close as `coinExposure * wantedPrice`. Trading only bounds `coinExposure`. It stores `_wantedPrice` in an event, then settles the close from oracle price and stored entry data. A builder 1CT can therefore submit a real tagged close with a legitimate size and an arbitrary `wantedPrice`. The close is accepted. The builder fee scales with that unused price, up to the trader's BuilderCode USDC allowance.

```

` ` plain text
} else if (_selector == CLOSE_TRADE_COIN_OI_SELECTOR) {
  coinExposure := calldataload(add(_cd.offset, 68))
  wantedPrice := calldataload(add(_cd.offset, 100))
  p.notional = (coinExposure * wantedPrice) / _COIN_NOTIONAL_DIVISOR;
}
`

```

A 0.25 coin close at a 51,000 wanted price and a 0.1 percent cap pays about 12.75 USDC. The same close with a 10x wanted price pays 127.5 USDC. The fill still uses the oracle. This path does not need a direct `chargeFee` call, so blocking that entry point would not remove the overcharge.

RECOMMENDATION

Derive coin close notional from the closed coin size and a price that actually constrains settlement, such as stored entry price or the fill price. Do not use caller supplied `wantedPrice`.

RESOLUTION

Status: Resolved.

HIGH **H-02**

RESOLVED

Delegate can drain fee allowance

CATEGORY Access Control

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L169-L207>

REVIEW Main Review 2

DESCRIPTION

chargeFee trusts any account whose code is an EIP-7702 designator for an approved template. EIP-7702 also lets that account originate an ordinary transaction, so a builder controlled 1CT wallet can call chargeFee directly instead of entering through the template. If it is an active trading delegate, it can supply an invented notional and a permissionlessly registered code that pays its own collector.

```
``plain text
require(!_isDelegatedToApprovedTemplate(msg.sender), "CALLER_NOT_TEMPLATE");
if (msg.sender != _trader) {
  IDelegateable.DelegateState memory delegation = trading.multipleDelegations(_trader,
msg.sender);
  require(delegation.isEnabled && delegation.expiry >= block.timestamp,
"NOT_TRADER_OR_DELEGATE");
}
fee = (_notional * feeRate) / FEE_RATE_PRECISION;
require(usdc.transferFrom(_trader, config.feeCollector, fee), "FEE_TRANSFER_FAILED");
`
```

There is no proof that the template derived `_notional` from a real trade. At the 1 percent cap, a caller can set notional to 100 times the user's USDC balance and consume the entire BuilderCode allowance without opening a position.

RECOMMENDATION

Bind fee collection to a real Trading action and derive the trader, side, pair, and notional inside a trusted protocol path. If chargeFee remains externally callable by delegated accounts, require a one use trader authorization that commits to every charged field and the collector.

RESOLUTION

Status: Resolved.

HIGH **H-03**

RESOLVED

Wrapped margin withdrawal mints collateral

CATEGORY Math

LOCATION - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/LimitCallbackModule.sol#L550-L566> - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L547-L560>

REVIEW Main Review 2

DESCRIPTION

`updateMargin` accepts an arbitrary unsigned withdrawal amount, but the callback casts it to `int256`. A value $2^{256} - x$ therefore becomes `-x`, so subtracting the requested withdrawal adds `x` to the trade's collateral and lowers its leverage.

```

` ` plain text
signedCollateral = int256(_currentCollateral)
- int256(_newAmount)
- int256(_borrowfees)
- _fundingFees;
` `

```

The oversized transfer normally reverts. However, `VaultManager` also casts `_amount` to `int256` when calculating the shortfall. For $2^{256} - x$, it therefore sees `-x`. If the resulting shortfall trips the breaker, the function returns before attempting the oversized transfer:

```

` ` plain text
uint256 shortfall = int256(_amount) > balanceAvailable
? uint256(int256(_amount) - balanceAvailable)
: 0;

if (pb.tripped || pb.counter + shortfall > pb.threshold) {
  _withholdPayout(_trader, _amount);
  return; // skips USDC.transfer
}
require(usdc.transfer(_trader, _amount));
` `

```

`_withholdPayout` pauses trading but does not revert. The margin callback therefore continues to `storageT.updateTrade(existingTrade)` and stores the fabricated collateral even though no USDC was transferred. The same early return can occur if the pull counter exceeds a lowered threshold or trading resumes while the breaker remains latched.

For example, with 1,000 USDC collateral, 10,000 USDC OI, and `balanceAvailable = -5,000 USDC`, withdrawing $2^{256} - 3,000$ USDC produces 4,000 USDC of stored collateral and 2.5x

H-03 Wrapped margin withdrawal mints collateral

CONTINUED

leverage. The signed payout amount is `-3,000 USDC`, so its computed shortfall is only 2,000 USDC. The breaker is designed to pause trading for offchain review, and the enormous `PayoutWithheld` amount should be conspicuous. Loss occurs if recovery misses the corrupted position and resets the breaker before resuming trading: the trader can then withdraw or close against the fabricated 3,000 USDC.

`_tryPauseTrading` also suppresses a failed pause with `try/catch`; the breaker latch still withholds payouts, but once governance clears that latch the router is already live, allowing the trader to race the remaining recovery steps.

If the withholding condition remains, each unsafe reset lets the attacker repeat the call and add more fabricated collateral.

— RECOMMENDATION

Reject a zero withdrawal and require `_amount <= existingTrade.initialPosToken`. Revalidate the amount after applying current borrow and funding fees because the amount safely withdrawable at execution can be lower than the stored collateral.

Also reject every value above `type(int256).max` before converting it, preferably with `SafeCast.toInt256`, or keep payout and collateral amount arithmetic unsigned. If `sendUSDCToTrader` withholds rather than transfers a margin withdrawal, do not persist the reduced collateral or reset the trade's fee snapshots.

More generally, avoid unchecked signedness changes and narrowing casts in accounting, collateral, debt, solvency, and payout paths. Use `SafeCast` or explicit range checks, and test direct and aggregate boundary values such as `2^N`; the very recent [Notional exploit](https://slowmist.medium.com/the-vanishing-debt-an-analysis-of-the-notional-finance-hack-a79b00fa2e47) showed that valid inputs can sum to a value that casts to zero and erases debt.

During recovery, verify that trading actually paused and repair any margin update finalized with `PayoutWithheld` before resetting the breaker or unpausing. The corrupted collateral survives a breaker reset.

— RESOLUTION

Status: Resolved.

MEDIUM **M-01**

ACKNOWLEDGED

Signed max rate follows live cap

CATEGORY Signatures

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L195-L207>
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L96-L114>
(<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L195-L207> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L96-L114>)

REVIEW Main Review 2

DESCRIPTION

The tagged suffix can pass `type(uint256).max ( USE_MAX_FEE_RATE )` to mean "charge the registered cap." The 7702 digest already hashes that suffix inside `Call.data`, so builder code and rate are on the EIP-7821 payload. The signature still binds a sentinel, not a concrete rate or collector. At execution, `chargeFee` substitutes the live `maxOpenFeeRate`, `maxCloseFeeRate`, or `maxPnlCloseFeeRate`, and `transferFrom` pays the live `feeCollector`.

The builder owner can `modifyBuilderCode` after the user or 1CT signs and before the relayer submits. The same signature then pays the new cap and the new collector. Integrator docs tell builders to use this sentinel, so in-flight batches are exposed whenever a frontend uses it. Explicit numeric rates in the suffix stay fixed.

This is not bounded by the old 10 USDC fixed fee cap. Fees are now a percent of notional under a 1 percent global ceiling. A 1,000 USDC, 10x open (10,000 notional) that signs `USE_MAX` at a 0.01 percent cap pays 1 USDC. If the builder raises the cap to 1 percent before relay, the unchanged payload pays 100 USDC. Larger notionals scale the same way. Changing the collector redirects the full fee.

Fixing this does not require a new Execute typehash or extra EIP-712 fields. The 7821 `Call[]` shape already carries the quote in trailing calldata.

RECOMMENDATION

Charge only a concrete signed rate, or snapshot the cap and collector that were in force when the batch was signed. Keep that quote in the existing suffix. A live `USE_MAX_FEE_RATE` lookup against mutable config is not a quote.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-02**

ACKNOWLEDGED

Market cancellations retain builder fees

CATEGORY Unexpected Behavior

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L547-L555> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageIntentExtension.sol#L187-L219> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L547-L555> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageIntentExtension.sol#L187-L219>)

REVIEW Main Review 2

DESCRIPTION

The template transfers the builder fee before calling Trading. A market open only creates an asynchronous pending order at that point. If the operator later force cancels the request, TradingStorage refunds the collateral but has no record of the builder fee and cannot refund it.

The same charge at request applies to limit and momentum opens. Those orders rest until trigger or cancel. `cancelOpenLimitOrder` refunds collateral and the execution fee, and it does not refund the builder fee. The trader can therefore pay a builder fee for an order that never fills, then cancel it themselves.

For example, a 1,000 USDC order at 10x with a 0.1 percent builder rate pays 10 USDC when requested. Canceling it before execution returns the 1,000 USDC collateral, opens no trade, and leaves the 10 USDC with the collector. This charges users for unfilled orders even though the documented Hyperliquid style model charges builder fees on fills.

RECOMMENDATION

Settle builder fees from the actual filled notional in the execution callback. If fees must be collected when the request is created, escrow them and refund the corresponding amount whenever the request is canceled or expires without a fill.

RESOLUTION

Status: Acknowledged.

MEDIUM **M-03**

RESOLVED

Deleted codes can be reclaimed

CATEGORY Access Control

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L154-L157>
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L68-L84>
 (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L154-L157> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L68-L84>)

REVIEW Main Review 2

DESCRIPTION

Governance `deleteBuilderCode` wipes the config, including `builderCodeOwner`, back to zero.

`registerBuilderCode` treats a zero owner as unused, so the first caller after that delete becomes the new owner and sets a new collector and caps.

Docs tell governance to delete or override an abusive code. Delete is the disable path, but it does not retire the identifier. It returns a live, already distributed `bytes32` to the public claim pool. `overrideBuilderCode` already reassigns ownership without that race.

```
```plain text
```

```
require(builderCodeConfig[_builderCode].builderCodeOwner == address(0),
"BUILDER_CODE_ALREADY_REGISTERED");
```

```
function deleteBuilderCode(bytes32 _builderCode) external onlyGov {
delete builderCodeConfig[_builderCode];
}
```

```
``
```

- Alice uses a frontend that tags every trade with builder code `Oxacme`.
- A 1,000 USDC, 10x open pays 10 USDC (0.1%) to that builder.
- The builder misbehaves. Governance deletes `Oxacme` to shut it off.
- Bob immediately registers the same `Oxacme` and points the collector at himself, with a 1% cap.
- Alice does not change anything. Her next tagged trade still says `Oxacme`.
- The fee now goes to Bob. If her suffix was `USE_MAX`, that same open pays him 100 USDC.
- This keeps happening until Alice's app or 1CT stops tagging `Oxacme`.

## RECOMMENDATION

Make deleted identifiers permanently unregistrable, or use `overrideBuilderCode` as the only disable path. Do not return a live code to the unowned state.

MEDIUM **M-04**

ACKNOWLEDGED

# Breaker pause blocks liquidations

CATEGORY	DoS
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L586-L604">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L586-L604</a>
REVIEW	Main Review 2

## — DESCRIPTION

When the pull breaker withholds a payout it calls `TradingRouter.pause()`. That pause is global. Router close, limit, TWAP, and RFQ entrypoints all use `whenNotPaused`, including operator `executeLimitOrder` for liquidations, take profits, and stop losses. `PriceAggregator. fulfill` is `onlyTrading`, so pending oracle orders also cannot settle until governance unpauses.

A large profitable close on one pair therefore stops keeper liquidations on every pair. Positions that go underwater while the router is paused cannot be liquidated on the public path, so the vault can accrue bad debt until governance unpauses. Risk manager `forceCloseTradeMarket` still works and is operational, not automatic.

Traders could also be unfairly liquidated during the pause as margin deposits are not allowed.

## — RECOMMENDATION

Do not pause all markets when a pull trip is caused by one close. Keep liquidation paths executable after a trip, or pause only the affected pair.

## — RESOLUTION

Status: Acknowledged.

MEDIUM **M-05**

ACKNOWLEDGED

# Changing decay factor reprices elapsed volume

CATEGORY Math

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L169-L195>

REVIEW Main Review 2

## — DESCRIPTION

setBreakerParams overwrites halfLifeFactor without first materializing volume decayed under the old factor, while \_decayedNow applies the current factor to the entire interval since lastUpdateTime.

## — RECOMMENDATION

Be aware and consider checkpointing decayedVolume and lastUpdateTime under the old factor before changing halfLifeFactor or its associated threshold.

## — RESOLUTION

Status: Acknowledged.

LOW **L-01**

ACKNOWLEDGED

# Guaranteed SL accepts prerequest prices

CATEGORY Oracle

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PriceAggregator.sol#L316-L345> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PriceAggregator.sol#L718-L730> (https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PriceAggregator.sol#L316-L345 https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PriceAggregator.sol#L718-L730)

REVIEW Main Review 2

## — DESCRIPTION

The M-20 remediation adds `UPDATE_SL` to `_ifParseUpdateData`. This binds fulfillment to the supplied observation, but it also moves guaranteed stop updates into the 60 second freshness class for Pyth Core, Pyth Lazer, and Avantis prices. The pending price order stores no request timestamp, so the signed update may predate the stop request.

```
``plain text
```

```
return _orderType == OrderType.MARKET_OPEN ||
```

```
// ...
```

```
_orderType == OrderType.TWAP_CLOSE ||
```

```
_orderType == OrderType.UPDATE_SL;
```

```
`
```

An approved operator or a stale operator pipeline can therefore fulfill a new guaranteed stop with an authentic observation from up to 59 seconds earlier. For example, a long can request a stop at 49,000 after the price falls below 49,000, then be fulfilled with a T-45 observation at 50,000. `updateSlCallback` accepts the stop because it only compares the stop with the supplied price. A later guaranteed stop settlement can shift the intervening leveraged loss to the vault. This is distinct from M-20's unprivileged cache preload, but the remediation doubles the previous 30 second freshness allowance and leaves it unbound to request time.

## — RECOMMENDATION

For guaranteed stop updates, require the accepted observation to be at least as recent as the stop request and keep a narrowly defined maximum age. Do not derive this rule from the shared 60 second parse classification.

## — RESOLUTION

Status: Acknowledged.

LOW **L-02**

ACKNOWLEDGED

## Lowered cap leaves old rates active

CATEGORY Configuration

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L52-L54> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L189-L207> ( <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L52-L54> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L189-L207> )

REVIEW Main Review 2

### DESCRIPTION

`setMaxFeeRateCap` updates the global cap, but the cap is checked only when a builder registers or modifies its configuration. `chargeFee` continues to use the stored per code maximum without constraining it to the current global cap. A builder registered at 0.1 percent therefore keeps charging 0.1 percent after governance lowers the global cap to 0.05 percent. This makes the advertised adjustable global ceiling ineffective for every existing builder until governance overrides or deletes each configuration separately.

### RECOMMENDATION

Constrain the applicable rate by both the stored per code maximum and the current global cap during every charge, or update all affected configurations as part of lowering the cap.

### RESOLUTION

Status: Acknowledged.

LOW **L-03**

RESOLVED

# TWAP and RFQ fees are unreachable

CATEGORY Unexpected Behavior

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L460-L473> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/TradingRouter.sol#L428-L529> ( <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L460-L473> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/TradingRouter.sol#L428-L529> )

REVIEW Main Review 2

## DESCRIPTION

The template recognizes the direct `openTwap` and `openRfq` selectors as fee eligible. Both direct Trading entry points always revert with `OnlyBatchedFlowAllowed`, while successful TWAP and RFQ orders use the operator only `executeTwapBatched` and `executeRfqBatched` paths. Those calls are not executed by the user's EIP-7702 account and are not recognized by the template.

As a result, the documented TWAP and RFQ builder fee feature has no successful execution path. Existing tests only prove that the fee rolls back when the direct calls revert.

## RECOMMENDATION

Carry signed builder fee parameters in the supported TWAP and RFQ intent format, then collect the fee from the actual successful intent or fill path.

## RESOLUTION

Status: Resolved.

LOW **L-04**

RESOLVED

## Delegate expiry mismatch on fees

CATEGORY Access Control

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L184-L187>  
<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/DelegateModule.sol#L107-L114> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/BuilderCode.sol#L184-L187> https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/DelegateModule.sol#L107-L114)

REVIEW Main Review 2

### DESCRIPTION

`chargeFee` treats a 1CT as valid when `expiry >= block.timestamp`. `delegatedAction` requires `expiry > block.timestamp`. At the exact expiry second the delegate can still pull a builder fee, including a direct `chargeFee` with an invented notional, but it cannot execute the trade.

The window is one second. Combined with caller supplied notional, that second is enough to take a fee with no matching order.

### RECOMMENDATION

Use the same expiry comparison as `delegatedAction` (`expiry > block.timestamp`) in `chargeFee`.

### RESOLUTION

Status: Resolved.

LOW **L-05**

ACKNOWLEDGED

# Withheld payouts remain redeemable by LPs

CATEGORY	Logical Error
LOCATION	<a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L547-L592">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L547-L592</a>
REVIEW	Main Review 2

## DESCRIPTION

The pull breaker completes a close even when it does not pay the trader. When a payout would exceed the breaker threshold, `VaultManager._sendUSDCToTrader` emits `PayoutWithheld` and returns without pulling assets from the tranche or transferring USDC. `_withholdPayout` records the exact debt only in an event.

``plain text

```
if (pb.tripped || pb.counter + shortfall > pb.threshold) {
 _withholdPayout(_trader, _amount);
 return;
}
```

```
function _withholdPayout(address _trader, uint256 _amount) internal {
```

```
...
```

```
_tryPauseTrading();
emit PayoutWithheld(_trader, _amount);
}
```

```
``
```

`sendUSDCToTrader` does not return whether payment happened. `TradeRegistry` therefore continues the normal settlement path: it decrements reward accounting, releases the position's reserved balance, and deletes or shrinks the trade. The trader has no on-chain claim after this point.

`plain text

```
storageT.vaultManager().sendUSDCToTrader(_trade.trader, usdcSentToTrader);
```

```
...
```

```
storageT.vaultManager().releaseBalance(_collateral.scaleByLeverage(_trade.leverage));
storageT.unregisterTrade(_trade.trader, _trade.pairIndex, _trade.index);
```

```
`
```

Only trading is paused. ERC-4626 withdrawals remain enabled, and `maxWithdraw` / `maxRedeem` account for `totalAssets` and `totalReserved` but not event-only payout debts. Because the failed payout leaves tranche assets unchanged while settlement releases `totalReserved`, LPs can redeem at a NAV that still includes the assets owed to the unpaid trader.

**L-05** Withheld payouts remain redeemable by LPs**CONTINUED****— RECOMMENDATION**

When pullBreaker.tripped is true, make maxDeposit, maxMint, maxWithdraw, and maxRedeem return zero, and make the tranche's deposit and withdrawal paths revert. Governance must settle every emitted PayoutWithheld amount before calling resetBreaker.

This prevents LPs from withdrawing the assets backing unpaid traders and prevents new LPs from depositing against an incorrect share price. It requires no new liability ledger or trader claim system.

**— RESOLUTION**

Status: Acknowledged.

LOW **L-06**

RESOLVED

# Calls decoded from unsigned signature bytes

CATEGORY Signatures

LOCATION - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L219-L228> - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/EIP7702Template/Eip7702Template.sol#L566-L589>

REVIEW Main Review 2

## DESCRIPTION

`validateUserOp` and `executeUserOp` pass `userOp.callData[4:]` to `_decodeCalls`. The decoder trusts the first word as an offset without proving that the resulting `Call[]`, its element heads, or its nested `bytes` remain inside that slice. `EntryPoint v0.8` hashes `callData` but excludes `signature`, and canonically encodes the operation again when calling both account functions. A deliberately out of bounds offset can therefore resolve calls from later, unsigned signature bytes during validation and execution. This does not let a relayer alter an ordinary, canonically encoded operation after it is signed because changing its `callData` offset would change `userOpHash`. The attacker must influence `UserOperation` construction before signing, for example through a malicious or compromised dapp, wallet backend, or sponsored transaction service. The user or remote signer must then approve short, noncanonical `callData`, usually while relying on that service's description or simulation instead of reconstructing and validating the encoding locally.

The account must also have an enabled modular validator that authenticates `userOpHash`, accepts trailing signature data, and does not independently restrict the decoded calls. The direct 65 byte owner signature path is unaffected. Under these conditions, the service can initially present benign calls in the signature suffix, obtain a valid signature, and replace only that unsigned suffix before submission. For a 36 byte `callData`, a raw array body placed after the 20 byte validator address, 65 byte validator signature, and 27 bytes of padding is reached by offset `236` (`124 + 20 + 65 + 27`). The substituted calls can transfer account assets or call `addValidator` through the zero target self call convention, resulting in loss of the affected account and persistent control.

## RECOMMENDATION

Decode `Call[]` only from the bounded `callData` slice, or validate the top level offset, array and tuple heads, and every nested `bytes` range before using calldata pointers. Reject noncanonical or out of bounds encodings.

---

**L-06** Calls decoded from unsigned signature bytes

---

CONTINUED

---

**— RESOLUTION**

Status: Resolved.

LOW **L-07**

ACKNOWLEDGED

# Pull breaker excludes live margin payouts

CATEGORY Trust Assumptions

LOCATION - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L547-L560> - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L75-L79>

REVIEW Main Review 2

## DESCRIPTION

`TradeRegistry` moves live trader collateral into `VaultManager`. The pull breaker counts only USDC pulled from the junior tranche, while the payout calculation treats `VaultManager`'s balance minus `totalRewards` and `fundingOwed` as available cash. Live collateral is not excluded. `reserveBalance` tracks leveraged exposure on the tranche and does not reserve this cash within `VaultManager`.

```
```plain text
```

```
int256 balanceAvailable = int256(storageT.usdc().balanceOf(address(this))) - int256(totalRewards) -
fundingOwed;
```

```
uint256 shortfall = int256(_amount) > balanceAvailable ? uint256(int256(_amount) - balanceAvailable) : 0;
```

```
if (pb.tripped || pb.counter + shortfall > pb.threshold) {
```

```
  _withholdPayout(_trader, _amount);
```

```
  return;
```

```
}
```

```
```
```

While the breaker is healthy and its counter is within the threshold, a payout covered by that available balance leaves the counter unchanged, even if it spends cash backing other open positions. For example, a 500 USDC payout from a 1,000 USDC balance with no reserved rewards or funding requires no tranche pull. That payout includes any returned collateral and is not necessarily 500 USDC of profit.

Abnormal payouts resulting from incorrect operator execution can therefore consume this cash before a tranche shortfall activates the breaker. Later trader settlements may need LP liquidity. This limits the breaker's usefulness as a measure of total settlement outflows. Once the breaker is tripped, buffer-funded payouts are also withheld.

## RECOMMENDATION

Document that the breaker limits tranche pulls and does not cap total payout volume or isolate open trader collateral. If the operational goal is to detect abnormal settlement outflows, monitor `VaultManager` payouts alongside tranche pulls.

**L-07** Pull breaker excludes live margin payouts

CONTINUED

**— RESOLUTION**

Status: Acknowledged.

LOW **L-08**

ACKNOWLEDGED

# Rebates remain payable after breaker trip

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Logical Error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| LOCATION | - <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L365-L375">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L365-L375</a> - <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L390-L424">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L390-L424</a> |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## DESCRIPTION

The pull breaker withholds `sendUSDCToTrader` payouts once it is tripped. Close settlement sends the referrer share through `sendReferrerRebateToStorage`, which transfers raw VaultManager USDC without checking the latch or updating the pull counter.

```
``plain text
if (feeValues.referrerRebate > 0) {
storageT.vaultManager().sendReferrerRebateToStorage(feeValues.referrerRebate);
}
`
```

This remains reachable after a successful pause because the risk manager can use `forceCloseTradeMarket`. It is also reachable through an ordinary close if the breaker's best-effort pause fails. `Referral` and `claimRebate` remain active, so a trader can attach a self-owned public code to an existing position after the trip and claim the rebate when that position is closed. The trader cannot initiate a force close, and the rebate cannot exceed the close fee charged to the position. An unfunded raw transfer reverts the close instead of pulling tranche assets. The issue therefore allows a bounded outflow during breaker recovery and can consume cash needed for withheld settlements, but does not create an independent TVL drain.

## RECOMMENDATION

While the breaker is tripped, defer close rebates and make them claimable only after the rebate amount is funded during recovery. Do not fund them through a helper that can pull tranche assets.

## RESOLUTION

Status: Acknowledged.

LOW **L-09**

ACKNOWLEDGED

# Long coin orders can inflate builder fees

CATEGORY Gaming

LOCATION - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/EIP7702Template/BuilderCode.sol#L290-L300> - <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/TradingCallbacks/MarketCallbackModule.sol#L170-L190>

REVIEW Main Review 2

## DESCRIPTION

The BuilderCode remediation (commit afda24cfc9e1) binds each fee to a real pending order, but coin exposure opens and increases still derive fee notional from the caller supplied `wantedPrice`. The value is capped at collateral multiplied by the pair maximum leverage, rather than the notional created by the later oracle fill.

```
``plain text
```

```
uint256 priced = (o.coinExposure * o.wantedPrice) / _COIN_NOTIONAL_DIVISOR;
uint256 bound = (collateral * pairStorage.pairMaxLeverage(
f.pairIndex, pairStorage.isPnlOrderTypeAllowed(f.pairIndex)
)) / _LEVERAGE_PRECISION;
f.notional = priced < bound ? priced : bound;
```

The fee is transferred immediately after the market request creates the pending order. For a long, the callback only rejects an execution price above `wantedPrice` plus slippage, so an inflated quote still permits a much lower real fill. A builder controlled 1CT with trading delegation and BuilderCode allowance can therefore overcharge the trader while executing a valid position.

For example, 2,500 USDC collateral and 0.5 BTC filled at 50,000 creates 25,000 USDC of actual notional, or 10x leverage. On a pair capped at 100x, an inflated quote makes the fee use the 250,000 USDC bound. At a 0.1 percent rate, the collector receives 250 USDC instead of 25 USDC. The pair leverage cap limits the overcharge but does not bind it to the executed trade. The previous coin close variant is fixed because closes now use stored entry price. The open and increase variants remain reachable through real orders, so preventing fabricated direct `chargeFee` calls does not resolve them.

## RECOMMENDATION

Calculate coin open and increase fees from the canonical coin size and actual callback fill. If the fee must be collected when the request is created, escrow a maximum amount and refund the difference after execution.

**L-09** Long coin orders can inflate builder fees

CONTINUED

A smaller request-time alternative is to cap notional at `collateral * pending.minLeverage` instead of pair max leverage. The callback already rejects a fill below that minimum, so a successful 10x trade cannot be charged as 100x. The drawback is that a wide leverage band undercharges honest fills ( `min` 1x, fill 10x pays a 1x fee), and a builder who sets `minLeverage` to pair max still takes that larger fee if the order later fails or is cancelled.

**— RESOLUTION**

Status: Acknowledged.

LOW **L-10**

ACKNOWLEDGED

## Builder fees can claim untagged orders

CATEGORY Access Control

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/EIP7702Template/BuilderCode.sol#L181-L276>

REVIEW Main Review 2

### — DESCRIPTION

The H-01 remediation derives fee facts from a real pending order, but it does not bind that order to the transaction that requested the fee. `chargeFee` accepts any approved template account that is the trader or a live Trading delegate, while order freshness only requires that the pending order was created in the current block.

```
```plain text
```

```
require(!_isDelegatedToApprovedTemplate(msg.sender), "CALLER_NOT_TEMPLATE");
```

```
require(!chargedOrders[_orderId], "ORDER_ALREADY_CHARGED");
```

```
OrderFacts memory f = _deriveOrderFacts(_orderId);
```

```
// ...
```

```
require(o.trade.trader != address(0) && o.block == block.number, "ORDER_NOT_FRESH");
```

```
```
```

A malicious builder controlled 1CT that remains a live delegate can therefore observe the trader's untagged market order and call `chargeFee` independently later in the same block. The order carries no builder code or fee rate, but the delegate selects both and transfers the resulting fee to its collector. At the 1% cap, claiming a genuine 1,000 USDC, 100x order transfers 1,000 USDC from the trader despite that order never authorizing a builder fee.

### — RECOMMENDATION

Charge builder fees from a protocol controlled Trading hook that binds the builder code, rate, collector, and order ID to the specific trade request. Current block freshness alone must not make an untagged order chargeable.

### — RESOLUTION

Status: Acknowledged.

LOW **L-11**

PENDING

## Coin RFQs can finish below signed target

|          |                                                                                                                                                                                                                                                                                                                                                                     |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Business logic errors                                                                                                                                                                                                                                                                                                                                               |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team2-1783453751962/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/TradingCallbacks/IntentCallbackModule.sol#L446-L532">https://github.com/GuardianOrg/avantis-contracts-team2-1783453751962/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/TradingCallbacks/IntentCallbackModule.sol#L446-L532</a> |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                       |

### — DESCRIPTION

The remediation completes a fixed coin RFQ when its coin target is reached and refunds unused collateral. However, it also treats the RFQ as complete whenever its collateral budget is exhausted, even if cumulative coin exposure remains below the signed target.

The operator chooses each slice's collateral and coin exposure within the signed leverage range. For example, a trader can sign a 0.1 BTC target with 1,000 USDC collateral and an allowed 2x to 10x range. At a 50,000 USDC price, the operator can consume all collateral to fill only 0.05 BTC at 2.5x. The callback registers that smaller position and deletes the RFQ, although the full target was feasible at 5x and can no longer be filled through the signed order.

### — RECOMMENDATION

For fixed coin RFQs, complete only when cumulative coin exposure reaches the signed target. Reject a final slice that exhausts collateral below that target so the operator must submit a conforming slice or cancel and refund the RFQ.

### — RESOLUTION

Status: Pending.

LOW L-12

ACKNOWLEDGED

## Coin TWAPs do not complete at their target

|          |                                                                                                                                                                                                                                                                                                                                                                     |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | DoS                                                                                                                                                                                                                                                                                                                                                                 |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/TradingCallbacks/IntentCallbackModule.sol#L119-L192">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/afda24cfc9e1e20251ef49a64f91246e0bd209c6/src/TradingCallbacks/IntentCallbackModule.sol#L119-L192</a> |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                       |

### — DESCRIPTION

`openTwapCallback` marks an open TWAP complete only when its collateral budget is exhausted. For coin-sized TWAPs it separately enforces and accumulates the signed coin target, but reaching that target does not complete the order.

For example, a trader can escrow 2,000 USDC for 0.1 BTC. At a 50,000 USDC price, one valid 5x slice spends 1,000 USDC and fills the full 0.1 BTC target. The callback nevertheless leaves the TWAP state and order-count slot active with 1,000 USDC still escrowed. Every further positive slice reverts with

`TwapCoinSizeExceeded`; a zero-coin slice cannot pass effective-leverage validation. Recovery requires a separate trader/delegate or operator cancellation.

### — RECOMMENDATION

Be aware and consider to mirror the RFQ completion logic for coin TWAPs.

### — RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-01

PENDING

# Delegate return is double encoded

CATEGORY Compatibility

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/TradingRouter.sol#L612-L618>

REVIEW Main Review 2

## DESCRIPTION

The I-01 remediation returns the raw result of a delegatecall to a module whose function already returns `bytes`. The delegatecall result is therefore the ABI encoding of that `bytes` value. Returning it again as `bytes` wraps the intended returndata in a second dynamic bytes encoding.

```
``plain text
```

```
(bool ok, bytes memory ret) = delegateModule.delegatecall(msg.data);
if (!ok) _bubble(ret);
return ret;
`|
```

For an inner action that returns one 32 byte word, `delegatedAction`` gives a Solidity caller a 96 byte value containing an offset, length, and the word. Integrators expecting the documented raw return data cannot decode the action result directly, so the original interface remains unusable without an undocumented extra decode.

## RECOMMENDATION

Decode the module's `bytes` return before returning it from the router, for example by returning `abi.decode(ret, (bytes))`.

## RESOLUTION

Status: Pending.

## INFORMATIONAL I-02

PENDING

# Sub-1% PnL distributions revert

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Rounding                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L630-L640">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L630-L640</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L663-L680">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L663-L680</a> ( <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L630-L640">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L630-L640</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L663-L680">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L663-L680</a> ) |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## DESCRIPTION

`distributeRewardsWithThreshold` calculates a positive PnL reward amount from the keeper-provided total and ratio, then converts that exact amount into a whole-number percentage of the current `pnlRewards` balance before calling `distributePnlRewardsFraction`. If the target is less than 1% of `pnlRewards`, the conversion truncates to zero. The callee calculates a zero distribution and reverts with `NoRewardsAllocated`, reverting the combined reward distribution. This is an operational edge case controlled by the authorized keeper; reward balances remain in the contract and the keeper can retry with a zero PnL share or a larger request.

## RECOMMENDATION

Be aware of and document this behavior.

## RESOLUTION

Status: Pending.

## INFORMATIONAL I-03

ACKNOWLEDGED

# Pull breakers default to zero

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Configuration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L95-L108">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L95-L108</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L538-L568">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L538-L568</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L79-L86">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L79-L86</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L150-L160">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L150-L160</a> ( <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L95-L108">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L95-L108</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L538-L568">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/VaultManager.sol#L538-L568</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L79-L86">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L79-L86</a> <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L150-L160">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/AvantisTranche.sol#L150-L160</a> ) |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## DESCRIPTION

The VaultManager and tranche pull-breaker thresholds are zero by default because their initializers do not configure them. As a result, the first payout requiring any tranche liquidity is withheld, the breaker trips and attempts to pause trading, while the trader's close still completes without immediate payment.

## RECOMMENDATION

Be aware of these zero defaults and consider setting the intended breaker parameters during initialization or an upgrade initializer.

## RESOLUTION

Status: Acknowledged.

## INFORMATIONAL I-04

PENDING

# Position rolls bypass leverage limits

|          |                                                                                                                                                                                                                                                                                                                                                                               |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATEGORY | Informational                                                                                                                                                                                                                                                                                                                                                                 |
| LOCATION | <a href="https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L418-L434">https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L418-L434</a> |
| REVIEW   | Main Review 2                                                                                                                                                                                                                                                                                                                                                                 |

## — DESCRIPTION

positionRoll settles accrued fees and calculates a replacement leverage, then stores that value directly without checking the pair's configured minimum or maximum leverage. A roll can therefore leave an existing live position below the minimum or above the maximum that ordinary entries must satisfy. This is a trusted keeper and governance migration path, but the resulting position remains active while outside the configured leverage range.

## — RECOMMENDATION

Be aware of this behavior and document it.

## — RESOLUTION

Status: Pending.

## INFORMATIONAL I-05

PENDING

# Partial closes leave phantom pair coin OI

CATEGORY Rounding

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L217-L246> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L756-L779> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PairInfos/PairInfos.sol#L1018-L1065> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L217-L246> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L756-L779> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/PairInfos/PairInfos.sol#L1018-L1065>)

REVIEW Main Review 2

## DESCRIPTION

Pair coin OI is credited by floor-converting a trade's full USD OI at its entry price. Partial closes later call `registerPartialTrade`, which floor-converts every closed USD slice independently and subtracts that amount from aggregate coin OI. Because floor conversion is not additive, the sum of the partial debits and final debit can be smaller than the original credit. The trade and its USD OI are then deleted while a small ownerless coin-OI remainder persists.

Ordinary partial market/limit closes, signed partial TP/SL closes, and fixed-coin market closes all reach this shared accounting path. Repetition can accumulate dust that marginally biases funding, hedging-fee, and maker/taker-fee calculations.

## RECOMMENDATION

Be aware and document this behavior.

## RESOLUTION

Status: Pending.

## INFORMATIONAL I-06

PENDING

# Terminal closes strand group OI reserves

CATEGORY Rounding

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L315-L425> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L217-L260> (<https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L315-L425> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingStorage/TradingStorageCoreExtension.sol#L217-L260>)

REVIEW Main Review 2

## DESCRIPTION

Group OI, `TradeInfo.openInterestUSDC`, and tranche reservations are initially credited with the same leveraged USD amounts. Partial closes debit independently rounded slices, while position increases can retain exact combined OI in the accounting ledgers but store a floor-rounded merged leverage. On terminal close, `TradeRegistry` releases group OI and the tranche reservation by reconstructing OI from the remaining collateral and stored leverage, while `TradingStorage` clears the exact remaining `TradeInfo` OI. Integer flooring is not additive, so the reconstructed terminal amount can be smaller after partitioned closes or a rounded position merge. The position is deleted, but the unmatched amount remains as ownerless group OI and tranche reservation dust, reducing exposure headroom and LP withdrawal capacity.

## RECOMMENDATION

Be aware and document this behavior, or use one authoritative per-position USD-OI value across the trade record, group OI, and tranche reservation accounting. Partial reductions should debit every ledger using the same exact delta, and terminal closes should release the complete stored remainder instead of reconstructing it from rounded collateral and leverage.

## RESOLUTION

Status: Pending.

## INFORMATIONAL I-07

PENDING

# Fees can strand fixed-coin market opens

CATEGORY Validation

LOCATION <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/MarketTradingModule.sol#L317-L356> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L114-L135> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/MarketCallbackModule.sol#L140-L204> ( <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/Trading/MarketTradingModule.sol#L317-L356> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/TradeRegistry.sol#L114-L135> <https://github.com/GuardianOrg/avantis-contracts-team1-1783118546829/blob/59240c5d6ab13f15ae43c973b357119165f3d60f/src/TradingCallbacks/MarketCallbackModule.sol#L140-L204> )

REVIEW Main Review 2

## DESCRIPTION

Standard non-PnL fixed-coin market opens validate coin notional and that `minLeverage <= maxLeverage`, but do not validate the implied leverage or whether collateral covers the opening fee before storing TradingStorage and PriceAggregator request state.

During later asynchronous fulfillment, the protocol derives leverage from coin exposure and collateral, calculates the opening fee from leveraged notional, and subtracts that fee before enforcing the submitted and pair leverage limits. If the fee after rebates exceeds the nonzero collateral, the subtraction panics before registration and pending-order cleanup complete. The already-committed request therefore cannot be fulfilled normally. An operator can refund and remove the TradingStorage record, but that cancellation does not remove the PriceAggregator request.

## RECOMMENDATION

Be aware of and document this behavior, or validate implied leverage and projected opening-fee sufficiency before transferring collateral or creating either pending record. Reject requests whose collateral cannot cover the fee or whose implied leverage falls outside the submitted or pair bounds.

## RESOLUTION

Status: Pending.

## § 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](mailto:audits@guardianaudits.com).