SMART CONTRACT SECURITY AUDIT OF

dreamDEX Perps

Smart Contract Security Audit

PROTOCOL Somnia	PREPARED BY Aetherion Labs Limited
DRAFT DATE 13 August 2026	FINAL REPORT DATE 26 October 2026
REVISED 25 November 2026	AUDITED COMMIT 1d290e7d6b509282309a753bf1264e7348ed41cd
FIX REVIEW COMMIT 08afa3b2ada38cc140d6e05bda76c68794f203ad	

Revision History

This page records each change made after the report dated 26 October 2026. The latest change is listed first.

REVISION 1

25 November 2026

The finding published as C-001 is now rated High and appears as H-009. We agreed on 30 September 2026 that this finding is High. The 26 October report still showed it as Critical.

Table of Contents

1. Revision History
2. Project Summary
3. Project Overview
4. Audit Scope
5. Vulnerability Summary
6. Findings & Resolutions
7. Medusa Invariant Testing
8. Appendix

Project Summary

PROTOCOL **dreamDEX Perps**	LANGUAGE **Solidity**
REPOSITORY **https://github.com/somnia-chain/dream-dex-perps-audit**	REVIEW PERIOD **July–August 2026**
AUDIT METHODOLOGY **Manual review, Foundry proof-of-concept validation, Foundry invariant / residual fuzz campaigns, Architecture and threat-model review**	GENERATED **13 August 2026**

Project Overview

dreamDEX Perps is a USDso-collateralized perpetual futures system on Somnia. Each market is a PerpPool wrapping the shared CLOB matching engine and settling through a singleton MarginBank. LiquidationEngine runs a six-stage waterfall (margin call, order cancel, CLOB IOC, bidder takeover, Insurance Fund, ADL). V1 also includes PerpStopOrderRegistry, VoucherDistributor, LinkedWalletRegistry with a linked-wallet-aware ADL ranker, and OperatorPermissionsRegistry.

Audit Scope

The following paths were included in the report scope configuration. Checksums are computed from the current workspace files and should be regenerated whenever the audited commit or source tree changes.

Core perps

FILE	SHA-1 CHECKSUM
src/perps/MarginBank.sol	db6eac3198b081e5a408cbd0a070f01d9c8b9458
src/perps/PerpPool.sol	c5a65f00788b725e0ebb6913d7666a00d57389bd
src/perps/PerpPoolFactory.sol	8339b04384df3d84576d153befbb8bb1251e9c15
src/perps/PerpMath.sol	39d2af9905dff899f109d989dd28873ecd52e7f0
src/perps/PerpTypes.sol	657355c5346d2ccae48a13aee5f0954f449d9248
src/perps/InsuranceFund.sol	606c1dd905b2af9c500c8878c2d936e61ba61ff1

Liquidation and ADL

FILE	SHA-1 CHECKSUM
src/perps/liquidation/LiquidationEngine.sol	2d14656b527876d5e838c3b511572dac50851b45
src/perps/liquidation/LinkedWalletAwareAdlRanker.sol	0f5c8de4862d02fdf326637a58409a5ff8f83f3a
src/perps/liquidation/HyperliquidAdlRanker.sol	3b660611be099e775ea75432bc99f8445f0d76c2
src/perps/liquidation/AdlScoreLib.sol	7ed67fda6aef0e8f9b3544043f94c048659cefcf
src/perps/liquidation/PerpsUnlinkGuard.sol	86eb1d4eb4c63e01106a70f022f43005c6d2be4d

Stops, vouchers, registries, and oracle

FILE	SHA-1 CHECKSUM
src/perps/PerpStopOrderRegistry.sol	5f9effb810bf78966d08ed4bba9069595c49ee84
src/perps/voucher/VoucherDistributor.sol	1f27f623bdcde123167e58261c1e46e313f63ead
src/registry/LinkedWalletRegistry.sol	f344d7a6bdd80586068471fc6ed7c0e9eb8fc3d2
src/registry/OperatorPermissionsRegistry.sol	c333955873cc69ea156d2c1a72e657d49d578bf8
src/oracle/AggregatorOracle.sol	00763824b0e17fd8c6ebbed5842e4366bb9b69c2

Interfaces

FILE	SHA-1 CHECKSUM
src/interfaces/IAdlRanker.sol	4c699fdcb209a2e4916aab4eef9b933be425e9c2
src/interfaces/IAggregatorOracle.sol	3e5ee85c985cb651f7b138295d1fc0d8052af4fb
src/interfaces/IERC20Vault.sol	8fd85a0184be1a577d6c24b96c372b4c5dd54816
src/interfaces/IInsuranceFund.sol	7e9d5285fcb41ce3302f7a6d2eac2dedec7acd76

FILE	SHA-1 CHECKSUM
src/interfaces/ILinkedWalletRegistry.sol	351b2069eb6a4945a0bc9c6712c1e384fb6a8e18
src/interfaces/ILiquidationBidder.sol	54d20b4bec86621baf9b985de673ce399575115b
src/interfaces/ILiquidationEngine.sol	98b18523be62aaa371a494af3e0b39321f85bb98
src/interfaces/IMarginBank.sol	e94f7c0e7710486a075a89682623d77e63d6ed3d
src/interfaces/IOperatorPermissionsRegistry.sol	0f5ee1e8bf18ae281f7723c1b0e8ca5bd7e560c6
src/interfaces/IOracle.sol	5fffc8a1808e9125f52441476ab65d0e2d3deede
src/interfaces/IOrderBook.sol	8079767f1e6978089bf7ab5f0f1d9ab903c08ebd
src/interfaces/IPerpPool.sol	dacd2629976def5ee24e5da12549a81126f344e9
src/interfaces/IPerpPoolFactory.sol	64c83034e003ffc902a4a13fd5e1dce176b4ce86
src/interfaces/IPerpStopOrderRegistry.sol	cc8b439752da4ab4a7be95bf3d4829d58196af27
src/interfaces/ISpotPool.sol	2478ff0c49da77f525a03062787d43b83b7e9a63
src/interfaces/ISpotPoolRegistry.sol	c41e580b6738956437941b0223d3a8d91801d344
src/interfaces/IUnlinkGuard.sol	0eaa3dd745304e095ab641943e985250387965f9
src/interfaces/IVoucherDistributor.sol	b812e227288f023f94d9921ed7343ac4b430ae48

Shared CLOB base (diff + integration review)

FILE	SHA-1 CHECKSUM
src/orderbook/OrderBook.sol	8ff0f202b7603c79df8559ba84b9c60359118d3f
src/orderbook/PriorityIndex.sol	7f4531b471867c17629978cd5290dff2e02fae9e
src/orderbook/LinkedList.sol	921ed73ccb54608800c10d87170eb1afa5ef3809
src/orderbook/OrderIndexManager.sol	41fe89aeb61002ef6a29a359b3e5a1493aaa5422
src/orderbook/PerUserOrderIndex.sol	df484086b7d3c02b81c08b3db0e68eae64eb8cd7
src/common/Common.sol	ef43deb7a11dd5856771f84efac54762f6195438
src/interfaces/IOrderBook.sol	8079767f1e6978089bf7ab5f0f1d9ab903c08ebd
src/vendor/bokkypoobahs/RedBlackTree.sol	ad7f827837b4d7f178b253c2f242b894ae1bdf40

Vulnerability Summary

TOTAL FINDINGS	FIXED
79	37
PARTIALLY FIXED	**ACKNOWLEDGED**
2	38
NOT REACHABLE	**WITHDRAWN**
1	1

Statuses reflect our review of the team's fix pull requests 11 to 16, 19 and 20, from September to October 2026, ending at the fix review commit. A status describes that commit only. Later code, configuration, or deployment changes can change it.

Severity Distribution

SEVERITY	TOTAL	RELATIVE VOLUME
CRITICAL	0	
HIGH	9	
MEDIUM	26	
LOW	15	
INFORMATIONAL	29	

Findings & Resolutions

ID	TITLE	SEVERITY	STATUS	ASSET
H-001	A liquidated trader can pocket Insurance Fund coverage while an offsetting profit stays open in the same account	HIGH	FIXED	LiquidationEngine.sol
H-002	Stale circuit-breaker references can admit fills that open negative-equity positions	HIGH	FIXED	PerpPool.sol
H-003	A trader can pay a close fee to their own second wallet and be credited USDso no deposit backs	HIGH	FIXED	PerpPool.sol, MarginBank.sol
H-004	A registered bidder can keep the takeover profit and leave depositors with the hole by taking over the opposite position	HIGH	FIXED	MarginBank.sol, LiquidationEngine.sol
H-005	A malicious user can drain the stop registry's SOMI and erase live stop-losses by pointing a Schedule at it	HIGH	FIXED	PerpStopOrderRegistry.sol
H-006	A trader can keep an underwater position open by parking a small order on a market whose mark has gone stale	HIGH	FIXED	PerpPool.sol, LiquidationEngine.sol
H-007	Anyone can sandwich a Stage 3 close into a thin bid so depositors back a flat-negative hole	HIGH	FIXED	LiquidationEngine.sol, MarginBank.sol
H-008	A close-out trader can dump into their own wallet at the circuit-breaker edge and pull out tokens no later liquidation collects	HIGH	FIXED	PerpPool.sol, MarginBank.sol
H-009	A malicious user can erase traders' stop orders by resting a wall of dust orders at the front of the book	HIGH	PARTIALLY FIXED	PerpStopOrderRegistry.sol
M-001	A trader can claim the same voucher grant twice by linking a wallet after the first claim	MEDIUM	FIXED	VoucherDistributor.sol
M-002	Broken stop-order Schedule chain lets anyone cancel still-	MEDIUM	FIXED	PerpStopOrderRegistry.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	triggerable user stops			
M-003	A trader can dodge ADL by linking a wallet just before the round runs	MEDIUM	FIXED	LinkedWalletAwareAdlRanker.sol
M-004	A trader can escape accrued funding by closing while the index feed is stale	MEDIUM	ACKNOWLEDGED	MarginBank.sol
M-005	A voucher recipient can unlock house principal by closing the losing leg first	MEDIUM	FIXED	MarginBank.sol
M-006	A trader can lose prepaid stop-loss SOMI while a take-profit still rests	MEDIUM	PARTIALLY FIXED	PerpStopOrderRegistry.sol, PerpPool.sol
M-007	A trader can dodge ADL by parking a temporary deposit before ranking runs	MEDIUM	FIXED	LiquidationEngine.sol, AdlScoreLib.sol
M-008	Anyone can keep Stage 3 off a real bid by resting quotes the pool will cancel	MEDIUM	FIXED	LiquidationEngine.sol
M-009	An attacker can skew funding with unfillable resting quotes that still set the midpoint	MEDIUM	FIXED	PerpPool.sol
M-010	A liquidated trader can keep collateral Stage 6 should burn when multi-page ADL resumes from a stale snapshot	MEDIUM	FIXED	LiquidationEngine.sol
M-011	A bankrupt trader can keep Stage 6 burn budget when it lands on a leg with no profitable opposite holders	MEDIUM	FIXED	MarginBank.sol, LiquidationEngine.sol
M-012	A single resting order at the open-interest cap can revert every opening taker instead of being skipped the way an over-cap maker is	MEDIUM	FIXED	PerpPool.sol
M-013	A Stage 4 bidder can keep close-out margin when the winning market is taken first	MEDIUM	FIXED	MarginBank.sol
M-014	An unrelated counterparty can absorb Stage 3 liquidation fees that never enter the Insurance Fund draw	MEDIUM	ACKNOWLEDGED	PerpPool.sol, LiquidationEngine.sol
M-015	A voucher holder can unlock house principal by naming	MEDIUM	FIXED	MarginBank.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	themselves as their own builder			
M-016	A liquidatable trader can claim a voucher, withdraw their deposit, and leave the open loss on house credit	MEDIUM	FIXED	VoucherDistributor.sol, MarginBank.sol
M-017	A voucher holder can poison a price level with one resting order so that every taker and every liquidation there reverts	MEDIUM	FIXED	MarginBank.sol, PerpPool.sol
M-018	A dropped Schedule tick can leave the stop registry unable to restart automation	MEDIUM	FIXED	SomniaExtensions.sol, PerpStopOrderRegistry.sol
M-019	Alice can withdraw before her funding is booked, so Bob is paid from other depositors	MEDIUM	FIXED	MarginBank.sol
M-020	A healthy short can absorb a bankrupt long's leftover loss when Stage 6 closes that long with no price cap	MEDIUM	ACKNOWLEDGED	MarginBank.sol
M-021	A quiet fallback feed can halt the market so underwater accounts cannot be liquidated	MEDIUM	ACKNOWLEDGED	AggregatorOracle.sol
M-022	Stage 3 closes the whole position when a fifth of it would restore margin	MEDIUM	ACKNOWLEDGED	LiquidationEngine.sol
M-023	An under-margined but solvent backstop cannot be reduced until it is bankrupt	MEDIUM	ACKNOWLEDGED	LiquidationEngine.sol
M-024	A trader can be stuck with a position she cannot close after a partial fill leaves her under the minimum order size	MEDIUM	FIXED	OrderBook.sol, LiquidationEngine.sol
M-025	A covered market's insurance tier can pay for an uncovered market's losses	MEDIUM	FIXED	LiquidationEngine.sol
M-026	An index-feed fault freezes liquidation even when the mark feed is healthy	MEDIUM	FIXED	PerpPool.sol, MarginBank.sol
L-001	A trader who shrinks a flip order to its reducing part can leave collateral locked with nothing left to release it	LOW	FIXED	PerpPool.sol
L-002	A trader's stop-loss cancel refund can become	LOW	FIXED	PerpStopOrderRegistry.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	unclaimable because the chain spends mixed SOMI against the raw balance			
L-003	A liquidatable wallet can unlink by topping up for one transaction, then fall straight back	LOW	ACKNOWLEDGED	PerpsUnlinkGuard.sol, LinkedWalletRegistry.sol
L-004	A liquidatable wallet can unlink when the oracle is stale because the guard fails open on any health-read error	LOW	ACKNOWLEDGED	LinkedWalletRegistry.sol, PerpsUnlinkGuard.sol
L-005	A malicious user can make fill-or-kill takers revert by resting an under-margined order the live fill then cancels	LOW	ACKNOWLEDGED	OrderBook.sol, PerpPool.sol
L-006	Revoking an operator on one market can look successful while a global approval still lets them trade there	LOW	FIXED	OperatorPermissionsRegistry.sol
L-007	A trader cannot arm a stop-loss when another market's feed lapses because create still checks initial margin	LOW	FIXED	PerpStopOrderRegistry.sol
L-008	A caller can skip one side of a stop-loss sweep by picking the block and the gas, and the call still reports success	LOW	ACKNOWLEDGED	PerpStopOrderRegistry.sol
L-009	Anyone can fill a market stop at the worst price in its published band by clearing the book and calling poke	LOW	ACKNOWLEDGED	PerpStopOrderRegistry.sol
L-010	An exhausted Stage 3 budget sends later liquidations to bidders instead of waiting for the next block	LOW	FIXED	LiquidationEngine.sol
L-011	One trader can hold a market's whole open-interest cap with a wash pair	LOW	ACKNOWLEDGED	PerpPool.sol
L-012	A trader in isolated mode can arm a stop on a second market while flat and lose the prepaid SOMI when it fires	LOW	NOT REACHABLE	MarginBank.sol, PerpStopOrderRegistry.sol
L-013	A triggered stop-loss can consume prepaid SOMI and	LOW	FIXED	OrderBook.sol, PerpStopOrderRegistry.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	report success after filling nothing			
L-014	A trader can block the owner from delisting a market by parking one minimum-size order	LOW	ACKNOWLEDGED	MarginBank.sol
L-015	A Stage 4 takeover can push a bidder past max position size so reducing fills that stay over the cap revert	LOW	FIXED	PerpPool.sol
Owner writes and rotations				
I-001	Rotating the insurance fund silently uncovers every market and leaves the funded capital at the old address	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol
I-002	The unbounded deleveraging scan setting is permissionlessly brickable, and the comment defending it assumes a number nothing enforces	INFORMATIONAL	ACKNOWLEDGED	HyperliquidAdlRanker.sol, LiquidationEngine.sol
I-003	An owner who points the fee recipient at a live builder bricks fills against that builder's resting orders	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol
I-004	An owner can settle elapsed funding at a new feed by rotating the aggregator while positions are open	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol, AggregatorOracle.sol
I-005	An owner who rotates the credit granter can leave expired voucher floors stranded with no one able to reclaim them	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol, VoucherDistributor.sol
I-006	An owner who lowers the builder fee cap still lets resting orders charge the old rate	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol
I-007	An owner can reprice live orders and positions in one write, with no bound on the change and no delay	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol
I-008	An owner can revive expired auto-deleveraging sessions so counterparties still close at a stale bankruptcy price	INFORMATIONAL	ACKNOWLEDGED	LiquidationEngine.sol

ID	TITLE	SEVERITY	STATUS	ASSET
I-009	An owner who lowers the builder fee cap can kill armed stops and burn the trader's prepaid SOMI	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol, PerpStopOrderRegistry.sol
I-010	The owner cannot switch a pool off a broken oracle because the switch first settles funding on that same oracle	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol
I-011	A trader loses their stored limit stop and the SOMI they prepaid for it when the owner changes the market's tick size	INFORMATIONAL	ACKNOWLEDGED	PerpStopOrderRegistry.sol, OrderBook.sol
Market listing and delisting				
I-012	A delisted market still pins the insurance tier cap, so the owner cannot shrink the tier structure until they clear a market nobody can trade	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol, InsuranceFund.sol
I-013	A market can be listed and traded before the engine can deleverage anyone on it, so positions opened in that window cannot be liquidated	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol, LiquidationEngine.sol
I-014	An owner who removes and re-adds a market silently restores voucher trading they had tried to cut off	INFORMATIONAL	FIXED	MarginBank.sol
I-015	A trader whose opening order was already resting keeps opening new exposure after the owner puts the market in close-only mode	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol
Rounding and dust				
I-016	Health marks a position with a division that truncates toward zero while closing it floors toward negative infinity	INFORMATIONAL	ACKNOWLEDGED	PerpMath.sol, MarginBank.sol
I-017	Every fill floors the notional, then the fee, then the insurance fund's share of it, so the protocol under-collects wei dust	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol, Common.sol
I-018	Partial fills release the collateral lock with a rounding-down division, leaving a few wei	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	locked until the order is cancelled			
I-019	The deleveraging claw-back buffer is a price offset that vanishes on small positions, so a multi-counterparty close can leave the bankrupt a few wei negative	INFORMATIONAL	ACKNOWLEDGED	PerpMath.sol
I-020	Averaging a position's entry price rounds down, biasing average entry by up to one price wei per increase	INFORMATIONAL	ACKNOWLEDGED	PerpMath.sol
I-021	The stored collateral scale factor is never read, advertising a decimals conversion the bank does not perform	INFORMATIONAL	FIXED	MarginBank.sol
I-022	The "no sub-wei rounding slack" claim on market health is false by one wei, because the notional it ceils was already floored	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol, PerpMath.sol
I-023	The oracle can keep serving a price the owner meant to reject because the two-feed gap check rounds down	INFORMATIONAL	ACKNOWLEDGED	AggregatorOracle.sol
Stops, operators, and the book				
I-024	One global operator approval covers both spot markets and every perp market the factory has ever minted	INFORMATIONAL	ACKNOWLEDGED	OperatorPermissionsRegistry.sol
I-025	The owner cannot upgrade the stop-loss registry in place, so a stuck callback chain needs a full redeploy	INFORMATIONAL	WITHDRAWN	PerpStopOrderRegistry.sol
Design notes				
I-026	The flash loan exposure here is live equity reads and live book reads, not reentrancy through deposit and withdraw	INFORMATIONAL	ACKNOWLEDGED	MarginBank.sol, AdlScoreLib.sol, PerpPool.sol
I-027	Batch funding catch-up applies one live midpoint across every skipped interval	INFORMATIONAL	FIXED	PerpPool.sol
I-028	A trader decides which market's insurance tier absorbs	INFORMATIONAL	ACKNOWLEDGED	LiquidationEngine.sol, MarginBank.sol

ID	TITLE	SEVERITY	STATUS	ASSET
	their bankruptcy by choosing the order they opened their positions in			
I-029	A maker can fill a resting order after the live initial margin rate has risen, using only the lock they posted at placement	INFORMATIONAL	ACKNOWLEDGED	PerpPool.sol

H-001

A liquidated trader can pocket Insurance Fund coverage while an offsetting profit stays open in the same account

HIGH FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1127-1150`

Description and impact

A liquidated trader can make the Insurance Fund pay a loss while offsetting profit remains open in the same account. They then close the winner and withdraw both. Stage 5 sizes coverage from the realized cash deficit and the covered-market loss booked during liquidation. It does not count profitable positions the account still holds:

```
uint256 deficit = uint256(-balanceBefore);
uint256 coveredNetLoss = coveredNet < 0 ? uint256(-coveredNet) : 0;
uint256 chargeable = deficit < coveredNetLoss ? deficit : coveredNetLoss;
if (chargeable > 0) bank.absorbBadDebt(ctx.account, tiers, losses, chargeable);
```

That works because liquidation can close one leg while another remains open. The profit is real equity, but Stage 5 sees only the cash balance after the losing leg realizes.

1. Alice deposits 90 USDso. She buys 10 at 100 in market A and sells 8 at 100 in market B.
2. The mark drops to 50. The long is down 500 and the short is up 400, so equity is -10 and she is liquidatable.
3. A keeper liquidates her. Stage 3 closes the long. The Insurance Fund pays the realized hole and leaves the short open.
4. Alice closes the short herself and withdraws more than the 90 she deposited.

Alice keeps Insurance Fund cash that her own open profit should have covered. Fund depositors lose that amount. The profitable leg can sit in an uncovered market, so she can choose the split deliberately.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_Stage5OverCoversRealizedLossWithOpenProfit.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_Stage5OverCoversRealizedLossWithOpenProfit.t.sol -vv`

Recommendation

Cap Stage 5 coverage at the shortfall remaining after the account's open positions are marked. Compute `chargeable` as the min of the cash deficit, the covered realized loss, and `max(0, -equity)`, so the Fund does not pay a hole that open profit already covers.

TEAM RESPONSE

Fixed in PR 12 (`13c6e03`).

H-002

Stale circuit-breaker references can admit fills that open negative-equity positions

HIGH FIXED

ASSET

`src/perps/PerpPool.sol:1053-1072`

Description and impact

A trader can fill a quote far from the live mark when the circuit breaker's cached reference lags a fast move. The counterparty locks collateral against the mark that existed when the order was posted, but the fill gate may still compare the trade against the old breaker reference:

```
uint256 refPrice = $.circuitBreakerReferencePrice;
if (refPrice == 0 || block.timestamp >= $.circuitBreakerReferenceUpdatedAt + $.circuitBreakerParameters.priceRefreshIntervalSec)
{
    // refresh refPrice from the live mark
}
require(diff * PerpMath.BPS_DENOMINATOR <= refPrice * maxMoveBps, FillPriceOutsideBand());
```

The margin check is anchored to the placement mark, while the circuit breaker can approve a later fill using a stale reference. A fast move creates a position whose opening loss is much larger than the locked margin.

1. A trade seeds the reference price at a mark of 100. Bob posts an ask at 110 and locks roughly 11 per unit against the 100 mark.
2. The live mark jumps to 200 before the refresh interval elapses, but the breaker reference stays 100.
3. Alice buys at 110. Bob opens short at 110 against a 200 mark and loses about 90 per unit.
4. Alice withdraws the matching profit.

The stale band lets Alice take roughly eight times Bob's posted margin. Bob cannot cover the loss, so the Insurance Fund, deleveraged counterparties, or remaining depositors cover the shortfall.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_CircuitBreakerStaleReferenceBadDebt.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_CircuitBreakerStaleReferenceBadDebt.t.sol -vv`

Recommendation

Compare organic fills to the live mark whenever the oracle is readable. If a resting quote is then outside that band, pull it and keep matching the incoming order against the rest of the book.

TEAM RESPONSE

Fixed in PR 12 (`61ea40f`). The price band moved to resting orders in PR 19 (`8a8adf2`).

AUDITOR NOTE

The team has accepted that fills are no longer checked against the price band, and relies on the maker and taker bad-debt checks to keep this closed. Any change to those checks should be reviewed against this issue.

H-003

A trader can pay a close fee to their own second wallet and be credited USDso no deposit backs

HIGH FIXED

ASSET

`src/perps/PerpPool.sol:774-785` `src/perps/MarginBank.sol:398-412`

Description and impact

A trader can close a position, push their own balance below zero with a fee, and credit a second wallet with unbacked USDso. The pool settles the trade first and charges fees after. The closing order can name its own builder, and the builder transfer is unfloored:

```
bank.settleTrade(takerOrder.owner, takerSizeDelta, fillPrice);
_chargeBuilderFees(takerOrder, makerOrder, quantity, fillPrice, bank, $);
function chargeBuilderFee(address payer, address builder, uint256 amount) external {
    _transferBalance(payer, builder, amount);
}
```

Once the payer crosses below zero, tracked claims rise with no new tokens. Liquidation would repair the hole, but a flat account with no active pools reads healthy. Builder fees are off until the owner raises the cap, but protocol fees already use the same unfloored debit.

1. Alice deposits 51 USDso and approves Carol, her second wallet, as builder. She buys 10 at 100.
2. The mark falls to 95. She sells them back while naming Carol.
3. The 50 USDso loss and taker fee leave about 0.025 USDso. The 9.5 USDso builder fee puts Alice at -9.475 and credits Carol 9.5.
4. Alice holds no position, so she reads healthy and is never liquidated. Carol withdraws 9.5 real USDso.

Carol keeps money the bank never received, and depositors permanently back Alice's abandoned debt.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_BuilderFeeFlatNegativeAttackerCredit.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_BuilderFeeFlatNegativeAttackerCredit.t.sol -vv`

Recommendation

Floor every fee debit at the payer's available balance, as Stage 4 collateral transfer already does with `fromBalance > 0 ? min(...) : 0`. Keep a flat negative account visible to liquidation.

TEAM RESPONSE

Fixed in PR 12 (`7e1f23d` , `5ef20ec`) and PR 19 (`8a8adf2`).

AUDITOR NOTE

For a market whose price is stale, the check counts the loss at the last good mark and never a profit. The team has accepted that a loss growing after that mark is not counted, and is left to the later liquidation of that market.

H-004

A registered bidder can keep the takeover profit and leave depositors with the hole by taking over the opposite position

HIGH FIXED

ASSET

`src/perps/MarginBank.sol:1343-1374` `src/perps/liquidation/LiquidationEngine.sol:1178-1188`

Description and impact

A registered bidder who is already short can take over a bankrupt long, keep the profit as cash, and wipe both positions so Stage 6 has nothing left to match. Stage 4 books realized PnL for both accounts, clears the bankrupt, and clears the bidder too when the transferred size nets them to zero:

```
_applyRealizedPnl(from, realizedPnl);
_clearPosition(fromPos);
_applyRealizedPnl(to, realizedPnl);
if (toOldSize + transferredSize == 0) _clearPosition(toPos);
```

Both positions go to zero while the cash does not. The bidder keeps the gain as collateral and the bankrupt takes the debit. Stage 6 only searches remaining positions, so it cannot give the hole to a profitable counterparty, and whatever the Insurance Fund does not cover stays on the ledger. A bidder can also skip the takeover and still erase the Stage 6 match. Stage 4 asks for a quote by calling `bidOnLiquidation`. That call is not a view, so the bidder can buy back their short on the book during the call and then return false. Stage 4 moves on. Stage 6 later looks for a profitable counterparty, finds the bidder already flat, and skips them.

1. Carol is a registered bidder, short 10 at 100. Alice is long 10 at 100 with 60 USDso.
2. The mark falls to 80 and Alice enters close-out.
3. Stage 4 gives Alice's long to Carol at 2% below the mark, 78.40.
4. Alice realizes -216 and is flat. Carol realizes +216 and her short clears, so Stage 6 finds no remaining Alice position and the bank is 156 USDso short of backing.

If Carol were not a bidder, Alice's long would reach Stage 6 and Carol's short would take the loss. As a bidder, Carol keeps 216 USDso and other depositors back Alice's 156 USDso deficit.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_Stage4OppositeMergeBreaksBacking.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_Stage4OppositeMergeBreaksBacking.t.sol -vv`

Recommendation

Before Stage 4 accepts a takeover that would flatten the bidder, check whether the Insurance Fund can cover the leftover debit. If it cannot, skip that bidder and continue to Stage 6. Snapshot Stage 6 candidates before `bidOnLiquidation`, or make that callback a view, so the bidder cannot close the absorbing position in the same transaction.

TEAM RESPONSE

Fixed in PR 12 (`8f5da83`) and PR 19 (`5cc97e9`).

H-005

A malicious user can drain the stop registry's SOMI and erase live stop-losses by pointing a Schedule at it

HIGH FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:453-458` `src/perps/PerpStopOrderRegistry.sol:533-551`

Description and impact

A malicious user can drain the stop registry's SOMI and erase live stop-losses by pointing a Schedule at it. They deploy a small contract that calls the Somnia reactivity precompile and names the registry as the handler. They pay for that one Schedule themselves. When it fires, Somnia calls the registry. The registry treats the delivery as its own tick, arms a new Schedule, and overwrites the id it tracks:

```
if (emitter != SomniaExtensions.SOMNIA_REACTIVITY_PRECOMPILE_ADDRESS) return;
if (eventTopics.length == 0 || eventTopics[0] != ISomniaReactivityPrecompile.Schedule.selector) return;
_scheduleNextSilently();
$.activeSubscriptionId = newId;
```

The callback lets any Schedule from the precompile through, so a delivery the registry never created still arms a new tick and overwrites the tracked id. The old Schedule stays live and still bills the registry. Only the latest id is stored, so the registry cannot cancel what it forgot. Each repeat leaves another leftover tick that keeps re-arming, and the registry then pays for many Schedules every interval instead of one.

1. The registry has one live Schedule. Alice pays SOMI for a sell stop-loss at 95.
2. The attacker's contract subscribes at the precompile with the registry as handler and waits for Somnia to deliver.
3. The registry arms a new tick and forgets the old id. The attacker repeats until many leftover Schedules fire every interval.
4. The registry's SOMI hits the floor and the tracked id goes to zero. The attacker cancels Alice's stop. The mark later trades through 95 and `poke()` has nothing left to fire.

Alice loses the stop-loss she paid for. Traders who cancel later can also be left with a SOMI refund the registry cannot pay.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_ForeignScheduleDeliveryForksSubscriptionChain.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_ForeignScheduleDeliveryForksSubscriptionChain.t.sol -vv`

Recommendation

Bind each callback to the Schedule the registry created. Store the expected timestamp or id, reject anything else, and cancel the live id before writing a new one. Gate inert cleanup on `intervalSeconds == 0`, not only `activeSubscriptionId == 0`.

TEAM RESPONSE

Fixed in PR 12 (`7a309bc`) and PR 19 (`f19e519`).

H-006

A trader can keep an underwater position open by parking a small order on a market whose mark has gone stale

HIGH FIXED

ASSET

`src/perps/PerpPool.sol:1182-1188` `src/perps/liquidation/LiquidationEngine.sol:2149-2160`

Description and impact

A trader can keep an underwater position open by parking a small order on a market whose mark has gone stale. `liquidate()` reads a fresh mark for every market on the account, including one that only has a resting order. Health runs before those orders are cancelled, so a stale mark reverts the whole call:

```
(price, updatedAt) = IOracle($.oracle).getMarkPrice();
require(
    price != 0 && updatedAt + $.fundingParameters.maxOracleStalenessSec >= block.timestamp, OraclePriceStale()
);
```

The same revert hits every wallet `liquidate()` prices:

- The account being liquidated. A resting order on a second market whose mark is stale aborts the close of a live losing position.
- Funding settlement. After an interval, that same resting order can revert on the index, not only the mark.
- A batch. `batchLiquidate` only skips accounts that are not liquidatable. A stale-mark revert still aborts every later account in the batch.
- Stage 4. Capacity walks every market a registered bidder holds. One stale mark there aborts the call, and a second healthy bidder is never tried. Refreshing that bidder's market unblocks liquidations that never used it.
- Stage 6. Equity walks every market a candidate holds. One stale mark there aborts auto-deleveraging even when the bankrupt's own mark is fresh.

1. Alice deposits 60 USDso, buys 10 at 100, and rests a small bid on a second market.
2. That second market's mark goes stale. The live mark falls to 80.
3. Every `liquidate(alice)` reverts. Her long stays open. Equity is -140 USDso.
4. She walks away. While the mark stays stale the hole can grow. Depositors or the Insurance Fund carry the 140 USDso hole.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_UncaughtStaleMarkReadsFreezeLiquidation.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_UncaughtStaleMarkReadsFreezeLiquidation.t.sol -vv`

Recommendation

Catch a stale mark, index, or equity read inside `liquidate()` and keep going. Cancel or skip markets that only have a resting order before health, including funding settlement on those pools. Treat a bidder who cannot be priced as lacking capacity. Skip a Stage 6 candidate whose equity cannot be priced. Do not let one stale mark abort the whole call.

TEAM RESPONSE

Fixed in PR 12 (`cd304ec`) and PR 16, with tests in PR 19 (`52dd587`).

AUDITOR NOTE

Bidders and ADL counterparties whose markets cannot be priced are now skipped instead of reverting the liquidation. The team has accepted that a stale feed can therefore narrow who takes on a loss.

H-007

Anyone can sandwich a Stage 3 close into a thin bid so depositors back a flat-negative hole

HIGH FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:2046-2062` `src/perps/MarginBank.sol:941-943`

Description and impact

A liquidator can have Stage 3 sell a bankrupt long below what the account still owns, withdraw the difference, and leave a debit no later stage will collect. Clearing the honest bids first, or finding the book already empty, leaves only a bid well below the mark. Stage 3 then sells at mark minus a spread that empty books widen, and that spread is not capped by remaining equity:

```
if (bookDepthNotional == 0) {
    spreadBps = (spreadBps * (PerpMath.BPS_DENOMINATOR + maxSizeMult)) / PerpMath.BPS_DENOMINATOR;
}
uint256 raw = (markPrice * (10_000 - spreadBps)) / 10_000;
```

The fill can close the last position and leave a negative cash balance. An account with no open markets reads healthy even then, so Stage 6 does not run. Stage 5 may credit the Insurance Fund if a covered tier still has room. Whatever is left stays on the ledger, and depositors absorb it. No later `liquidate` comes back, because the account is no longer liquidatable.

1. Alice deposits 100 USDso and buys 10 at 100. The mark falls to 90. Equity is 0 and she is liquidatable.
2. An honest bid sits at 90. Left alone, Stage 3 would sell her at 90 and she would end at 0.
3. A liquidator sells 10 into that bid at 90, rests a bid at 75, and calls `liquidate(alice)` in the same transaction. Thin depth widens the Stage 3 limit so the engine sells Alice into 75.
4. Alice is flat at -150. The liquidator sold at 90 and bought back at 75, so they keep 150 USDso as ordinary withdrawable cash.

Stage 6 cannot socialize a cash hole with no position left. If the Insurance Fund does not cover it, the 150 USDso stays as an unbacked claim against other depositors.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_LiquidatorHarvestsTheLiquidationConcession.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_LiquidatorHarvestsTheLiquidationConcession.t.sol -vv`

Recommendation

Clamp Stage 3 so a book fill cannot push the account below zero. Leave deeper penalties to Stage 4. If nothing rests inside the band, fall through instead of widening on cleared depth:

```
uint256 solvencyCapBps = equity > 0
    ? ((uint256(equity) + mmReq) * PerpMath.BPS_DENOMINATOR) / positionNotional
    : (mmReq * PerpMath.BPS_DENOMINATOR) / positionNotional;
if (spreadBps > solvencyCapBps) spreadBps = solvencyCapBps;
```

TEAM RESPONSE

Fixed in PR 12 (`680472b`) and PR 19 (`492115c`).

AUDITOR NOTE

A Stage 3 sale can no longer take the account below zero, apart from a few wei of rounding when several makers fill one sale. The team has accepted that a buyer who clears a thin book can still take the account's remaining equity as the liquidation concession.

H-008

A close-out trader can dump into their own wallet at the circuit-breaker edge and pull out tokens no later liquidation collects

HIGH FIXED

ASSET

`src/perps/PerpPool.sol:731-740` `src/perps/MarginBank.sol:941-943`

Description and impact

A close-out trader can dump into their own wallet at the circuit-breaker edge and pull out tokens no later liquidation collects. A closing sell skips the initial-margin check. The pool treats that skip as safe, because a reduce should improve health. A reduce does not improve health when the fill is worse than the equity that is left. The circuit breaker still lets the fill print as far as 5% from the mark. That band equals initial margin, the widest setting the pool allows.

```
if (increasingQuantity > 0) {
    require(bank.meetsIMForOrder(order.owner, increasingQuantity, order.price), InsufficientMarginForOrder());
}
if ($.accounts[account].activePerpPools.length == 0) return MarginStatus.Healthy;
```

After the close, the trading wallet holds no position. The bank treats an account with no open markets as healthy, even when the cash balance is negative. Liquidation then refuses it. Auto-deleveraging only searches open positions, so a cash hole with nothing left to close is a no-op.

1. Alice's trading wallet deposits 50 USDso and buys 10 at 100. The mark falls to 95. Equity is 0 and she is already in close-out.
2. Selling at 95 would end that wallet at 0. Instead her second wallet, Carol, rests a bid at 90.25. The trading wallet sells 10 into it.
3. The trading wallet sits at -47.5 and reads healthy. Carol sells at 95 and withdraws 47.5 USDso.
4. A later `liquidate` on the trading wallet reverts.

Carol withdraws 47.5 USDso the bank never received. Other depositors back that withdrawal.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_CloseOutDumpAtCircuitBreakerEdge.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_CloseOutDumpAtCircuitBreakerEdge.t.sol -vv`

Recommendation

Reject a user-placed reducing fill that would leave the closer flat and below zero, so the account stays in the waterfall. Leave liquidation IOCs to the Stage 3 clamp. Keep a flat negative account visible to liquidation and charge leftover cash deficits to the Insurance Fund.

TEAM RESPONSE

Fixed in PR 12 (`7e1f23d` , `5ef20ec`), PR 19 (`8a8adf2`), and PR 20 (`08afa3b`).

AUDITOR NOTE

The team has accepted that a loss growing on a stale market after its last good mark is left to that market's liquidation. The check PR 20 added to stop margin being stripped from an open position also refuses a triggered stop whose partial fill below the mark would leave the trader under maintenance, and the stop is then deleted.

H-009

A malicious user can erase traders' stop orders by resting a wall of dust orders at the front of the book

HIGH PARTIALLY FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:613-633` `src/perps/PerpStopOrderRegistry.sol:689-708`

Description and impact

A malicious user can erase traders’ stop orders by resting a wall of dust orders at the front of the book. Those are bids as small as the book allows, parked where a market sell stop-loss will walk first. When the mark trades through, the scheduled tick places the stop-loss as an immediate-or-cancel order. It spends the prepaid gas matching those bids, then deletes the stop as if placement failed. The trader’s long stays open and the prepaid SOMI is gone. The bids never fill, so they wait for the next stop-loss.

The tick runs with the gas the owner prepaid on the Schedule. Tests use 5,000,000, about 80 minimum-size bids. `poke()` walks the same bids. More gas only means more bids. The tick never keeps gas aside after placement:

```
if (gasleft() < gasFloor) break;
(bool success, DropReason dropReason,,) = _processOrder(orderIndex, price);
OrderIndex nextOrderIndex = stopIndex.eraseOrder(orderIndex);
$.reservedSomi -= somiPaid;
```

1. Alice buys 10 at 100 and pays SOMI for a market sell stop-loss at 95.
2. The attacker is short 10 against her and parks about 80 minimum-size bids at 95. An honest bid that would exit her rests at 94.
3. The mark falls to 94. The tick walks the 95 bids, runs out of prepaid gas, and deletes Alice’s stop. Her long is untouched. The same bids delete a replacement stop-loss the same way.
4. The mark falls to 80. Alice is still long and down 200 USDso. A fill at 94 would have limited her loss to 60 USDso.

The extra 140 USDso is Alice’s loss from the deleted stop. The same bids delete the next trader’s stop-loss the same way. If Alice later goes bankrupt, other depositors or the Insurance Fund pay the extra a 94 stop-loss would have prevented.

Proof of Concept

POC FILE

`findings/pocs/highs/PoC_DustWallDestroysTriggeredStops.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/highs/PoC_DustWallDestroysTriggeredStops.t.sol -vv`

Recommendation

Give placement leftover gas minus a reserve so the registry can finish the tick if the book walk runs dry. Delete the stop-loss only when the pool rejects the order. If placement runs out of gas, leave it pending and keep the SOMI:

```
uint256 reserve = _TEARDOWN_GAS_RESERVE;
if (gasleft() < gasFloor + reserve) break;
(bool success, DropReason dropReason,,) = _processOrder{gas: gasleft() - reserve}(orderIndex, price);
```

Raise the market's minimum order size so a wall of tens of bids locks real margin. More gas still loses to more bids.

TEAM RESPONSE

Partially fixed in PR 11 (`ff8c8e7`) and PR 19 (`acbf47b`).

AUDITOR NOTE

The team has accepted that a wall of minimum-size bids can still run a stop out of gas, and that the stop is then deleted rather than retried, leaving the position with no stop. It limits the payoff with a minimum order size of about $10 and the zero-equity limit on Stage 3 sales, and is exploring a cap on how many orders one sale can match.

M-001

A trader can claim the same voucher grant twice by linking a wallet after the first claim

MEDIUM FIXED

ASSET

`src/perps/voucher/VoucherDistributor.sol:111-127` `src/perps/voucher/VoucherDistributor.sol:276-284`

Description and impact

A trader can claim a voucher grant twice from the same identity. They claim once before linking, then again after linking, because Group mode switches the claim key from the wallet address to the group main. `claim()` marks only the key resolved at call time:

```
bytes32 id = _claimId($, msg.sender);
require(!$.claimed[id], AlreadyClaimed());
$.claimed[id] = true;
IMarginBank($.marginBank).creditAccount(msg.sender, grant);
```

In Group mode `_claimId` returns the wallet while unlinked and the group main after linking, so `claimed[alice]` does not block a later claim under `claimed[M]`.

1. Alice claims as standalone, which sets `claimed[alice]`.
2. Alice links to group main M.
3. Alice claims again. `claimed[M]` is still false, so she receives a second grant.
4. She unlinks, links a fresh throwaway main, and repeats until the campaign budget is gone.

Alice drains house USDso that was meant as one grant per identity, and each repeat also overwrites `voucherExpiry` so recovery on the accumulated floor is postponed.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_VoucherGroupLinkAfterClaimDoubleGrant.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_VoucherGroupLinkAfterClaimDoubleGrant.t.sol -vv`

Recommendation

Make the identity key stable for the full claim lifecycle. Reject linking when either the standalone wallet key or the target group key has already claimed, or record both keys on claim and check both before granting credit.

TEAM RESPONSE

Fixed in PR 15 (`b7f9641`).

M-002

Broken stop-order Schedule chain lets anyone cancel still-triggerable user stops

MEDIUM FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:233-246`

Description and impact

`cancelInertOrders` is permissionless cleanup for stops that can no longer fire. Its only guard is `activeSubscriptionId == 0`. That id is zero after a real shutdown, and it is also zero when the registry falls below the 32 SOMI floor and cannot renew the Schedule. In that second gap `poke()` can still fire the stop, so anyone who calls cleanup deletes a live stop-loss as price approaches the trigger.

```
function poke() external { require($.intervalSeconds != 0, NoActiveSubscription()); }
function cancelInertOrders(OrderId[] calldata orderIds) external {
    require(_getPerpStopOrderRegistryStorage().activeSubscriptionId == 0, SubscriptionStillActive());
}
```

1. Alice arms a stop-loss that should sell when the mark falls to 95 and prepays 0.1 SOMI.
2. The registry balance falls below the 32 SOMI renewal requirement. The next automatic tick cannot renew, so automatic checks pause while `poke()` still works.
3. The mark falls toward 95 and Bob calls `cancelInertOrders()`, deleting Alice's stop.
4. The mark crosses 95 and nothing fires. Alice reclaims the prepaid SOMI, but her exit order is gone.

Any watcher who sees price approaching a trigger during that gap can strip stop protection while automatic checks are down. The refund does not replace the missing hedge if mark keeps moving against the open position.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_StopOrderBrokenChainPermissionlessCancel.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_StopOrderBrokenChainPermissionlessCancel.t.sol -vv`

Recommendation

Only allow `cancelInertOrders()` when the registry is actually dormant. Require `intervalSeconds == 0`, matching `removeSubscription()`. Users can still call `cancelPendingOrder()` for their own orders while keepers use `poke()` during a broken but configured chain.

TEAM RESPONSE

Fixed in PR 12 (`7a309bc`).

M-003

A trader can dodge ADL by linking a wallet just before the round runs

MEDIUM FIXED

ASSET

`src/perps/liquidation/LinkedWalletAwareAdlRanker.sol:101-217`

Description and impact

A trader who is about to be picked for deleveraging can link a second wallet moments before the ADL round runs. That link can shrink their scored size or push a linked wallet to the back of the queue, so an unrelated trader gets deleveraged instead. Linking has no health, age, or liquidation-status gate, and the ranker reads live group membership on every call. Linked wallets get a netting credit that shrinks scored size up to `creditCapBps` (default 30%), and any wallet linked to the bankrupt account sorts strictly last as a backstop:

```
int256 groupNet = _cachedGroupNet(candMain, c.perpPool, cache);
int256 dirNet = c.size > 0 ? groupNet : -groupNet;
effSize = _clampEffSize(dirNet, absInt128(c.size), capBps);
```

1. Alice is short and would normally be picked for ADL. Bob holds the same size short with no link.
2. Alice opens a second wallet with an opposite long and links it to herself.
3. The ranker scores her short as smaller under the netting credit.
4. When an unrelated bankrupt needs a short closed, Alice ranks behind Bob and Bob is ADL'd instead.

Bob absorbs the loss Alice dodged. The same timing gap parks a profitable wallet in the backstop tier so unrelated traders take the hit first, and nothing about the link decays across rounds.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_LinkedWalletAdlLastMinuteLink.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_LinkedWalletAdlLastMinuteLink.t.sol -vv`

Recommendation

Block linking when either wallet is already at partial liquidation or worse. Require link age before ADL ranking, or snapshot linked groups when an ADL session starts and enforce that snapshot on every resume.

TEAM RESPONSE

Fixed in PR 13 (`e4ab964`) and PR 19 (`d5e6f0f`).

AUDITOR NOTE

A zero window is now refused at initialization and in the setter. The team has accepted that a registry upgraded in place reads 0 as off until the owner sets the window.

M-004

A trader can escape accrued funding by closing while the index feed is stale

MEDIUM ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:1482-1507`

Description and impact

A trader can close or flip a position while the index feed is stale and avoid the funding that accrued during that window. The escape is permanent because funding settles only while the position still exists. `MarginBank._settleFunding()` asks the pool to update funding before a size mutation, but it ignores whether the update actually succeeded. When open interest is nonzero and the index is stale, the pool returns `false` without advancing the cumulative index, yet settlement still runs against the old index and clears the position:

```
IPerpPool(pool).updateFunding(); // return value ignored
Position storage pos = $.positions[account][pool];
if (pos.size == 0) return 0;
int256 cumulativeFunding = IPerpPool(pool).getCumulativeFundingPerUnit();
payment = PerpMath.fundingPayment(pos.size, cumulativeFunding, pos.entryFundingIndex, oneBase);
```

1. Two longs and a short are open at cumulative funding index zero.
2. One funding interval elapses, but the index feed goes stale before `updateFunding()` can advance the index.
3. One long closes during the stale window. Funding settlement runs, charges zero, and clears the position.
4. The feed recovers and the index catches up. The remaining long pays its share. The short receives less than the interval produced because the closed long's payment was never collected.

The closed long keeps collateral they owed and the short is underpaid. There is no later billing hook for an exited account, and flipping sides in the same window can make an honest counterparty pay instead of receive.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_StaleIndexLetsTraderAvoidFunding.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_StaleIndexLetsTraderAvoidFunding.t.sol -vv`

Recommendation

Do not let size or collateral mutations proceed when open interest is nonzero and funding could not update. Revert those operations, or snapshot pending funding debt before a position can shrink, close, or flip.

TEAM RESPONSE

Acknowledged in PR 13, with the bound tested in PR 19 (`023fdb5`).

AUDITOR NOTE

The team has accepted that a position closed while the index is stale escapes up to one settlement interval of funding per outage. That is about 0.5% of notional at the current 4% per 8-hour cap, settled hourly.

M-005

A voucher recipient can unlock house principal by closing the losing leg first

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:1564-1589`

Description and impact

A voucher recipient can unlock house principal by realizing the losing leg of an offsetting book before the winning leg. Net trading profit can be zero, but the non-withdrawable floor falls during the loss and never rises when the matching profit later arrives. `_debitBalance()` ratchets `nonWithdrawableCredit` down whenever realized cash plus locked collateral drops below the current floor, and the comparison ignores open profit on the same cross-margin account:

```
int256 economicBalance = newBalance + int256($.accounts[user].lockedCollateral);
if (economicBalance < int256(floor)) {
    $.nonWithdrawableCredit[user] =
        economicBalance > 0 ? uint256(economicBalance) : 0;
}
```

`_creditBalance()` also does not raise the floor back after the offsetting gain becomes realized.

1. Alice receives a 1,000 USDso voucher. `nonWithdrawableCredit` is 1,000.
2. She opens a long in market A and a short in market B at 100.
3. Mark falls to 50. She closes the long first, realizes a 500 loss, and the floor ratchets to 500 while the short still carries open profit.
4. She closes the short for a 500 gain. Balance is back at 1,000 against a floor of 500, so she withdraws 500 of house principal.

Alice keeps the grant's cash with a lower floor. The campaign sponsor loses house credit that was meant to stay locked as margin. The same debit-then-credit shape can arise from funding alone.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_VoucherFloorIgnoresOpenProfit.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_VoucherFloorIgnoresOpenProfit.t.sol -vv`

Recommendation

Include open-position equity in the ratchet comparison. Raise the floor on realized credits that restore house principal, or only ratchet after the voucher account is flat.

TEAM RESPONSE

Fixed in PR 15 (`ae214d7`).

M-006

A trader can lose prepaid stop-loss SOMI while a take-profit still rests

MEDIUM PARTIALLY FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:720-738` `src/perps/PerpPool.sol:1327-1338`

Description and impact

A trader can rest a take-profit and a stop-loss on the same long. The stop should sell when the mark hits. The registry sizes that sell against the raw position. The pool subtracts the resting take-profit first and treats the extra size as a new short. That leftover needs initial margin. A stop fires when free collateral is already thin, so the check rejects the order. The registry burns the prepaid SOMI and the long stays open:

```
qty = orderWithTrigger.order.quantity < absSize
    ? orderWithTrigger.order.quantity : absSize;
uint256 effectiveReducingCapacity =
    absSize > pendingOpposite ? absSize - pendingOpposite : 0;
increasingQuantity = fullQuantity - effectiveReducingCapacity;
```

1. Alice holds a 10-unit long and rests a reducing ask for 4. Free collateral is thin.
2. She arms a stop-loss to sell 8 and prepays SOMI.
3. Mark crosses the stop. The registry fires because 8 fits in her 10-unit long.
4. The pool counts only 6 of reduce left, so 2 units of the sell are a new short. The initial-margin check rejects the order. The registry burns the prepaid SOMI and the long stays at 10.

Alice still has the take-profit on the book and no stop. The dump she prepaid to exit keeps running against the open long.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_RestrictedStopPendingOppositeBurnsSomi.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_RestrictedStopPendingOppositeBurnsSomi.t.sol -vv`

Recommendation

Apply the same pending-opposite math in `_resolveReduceOnly` before a stop fires. Do not burn prepaid SOMI when the pool rejects a stop the registry already treated as reduce-only against raw size. Catching only `MarketRestricted` still burns SOMI on the ordinary thin-margin path.

TEAM RESPONSE

Partially fixed in PR 13 (`c9a32e6`).

AUDITOR NOTE

The team has accepted that a take-profit resting for the whole position leaves the stop nothing to sell, so the stop is dropped when it fires and the position has no stop protection. It will not let the registry cancel the trader's take-profit, and offers `getReducingCapacity` so an app can warn before the stop is armed.

M-007

A trader can dodge ADL by parking a temporary deposit before ranking runs

MEDIUM FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1893-1896`

`src/perps/liquidation/AdlScoreLib.sol:42-62`

Description and impact

A trader who is next in line for deleveraging can deposit a large balance right before an unrelated liquidation runs. Ranking then skips them in favor of the next-ranked counterparty. They withdraw the deposit moments later and keep the position that should have absorbed the loss. ADL ranking reads live account equity on every `liquidate()` call. There is no equity snapshot for the session and no cooldown on deposits, so leverage score can change in the same block as ADL runs:

```
accountEquity: bank.getAccountEquity(holder)
// ...
leverageFactor = saturatingFactor(effSize * markPrice, equity);
```

1. Alice and Bob each hold a short of size 10. Alice is thinly collateralized and ranks first for ADL.
2. Carol's long goes bankrupt and needs opposite-side deleveraging.
3. Alice deposits 5,000,000 USDso immediately before anyone calls `liquidate(Carol)`. ADL runs with her inflated equity and Bob is deleveraged instead.
4. Alice withdraws the deposit and keeps her short.

Alice shifts the loss onto Bob for the cost of a temporary deposit that is returned moments later.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_AdlEquityDepositFrontrun.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_AdlEquityDepositFrontrun.t.sol -vv`

Recommendation

Snapshot each candidate's equity when the ADL session is created and reuse it for ranking and continuation. Alternatively, score leverage from position-locked margin or entry notional so a same-block deposit cannot rewrite priority.

TEAM RESPONSE

Fixed in PR 13 (`e4ab964`), demonstrated in `3e3704f`.

AUDITOR NOTE

The proof of concept still passes because its setup funds every account in the same second as the liquidation. The team has demonstrated the fix with the same scenario using deposits one second older.

M-008

Anyone can keep Stage 3 off a real bid by resting quotes the pool will cancel

MEDIUM FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:2000-2009`

Description and impact

Anyone can rest bids at the mark that the pool will cancel before they fill. Stage 3 then treats the book as thick, skips the size multiplier, and never sells into a real bid far below mark. The depth read sums the top levels with no price band and no check that those makers can take the fill:

```
OrderBookLevel[] memory levels =
    IOrderBook(pool).getBookLevels(isBid, _BOOK_DEPTH_LEVELS);
for (uint256 i; i < levels.length; ++i) {
    depthNotional += (levels[i].quantity * levels[i].price) / oneBase;
}
```

Those quotes still sit on the book, so the engine keeps the Stage 3 limit close to mark. The real bid below that limit never fills.

1. Mark is 95. Alice holds a 10-unit liquidatable long. Bob rests the only real bid: 3 units at 47.50.
2. Carol is already at max position size. She rests a large bid at 95. Two more wallets do the same.
3. Stage 3 counts that size, keeps the limit around 81, and matches nothing against Bob.
4. Those bids cancel before they fill. Alice is still 10. The same `liquidate()` continues into Stage 4.

Bob never gets the fill a thin book would have given him. Later stages of that call close whatever is left. Large bids that sit below the tight limit count as depth the same way, even with no extra wallets.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_Stage3OptimisticBookDepth.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_Stage3OptimisticBookDepth.t.sol -vv`

Recommendation

Count only depth the Stage 3 limit could actually fill. Ignore makers the pool would cancel before a match, and ignore bids outside the widest limit, before applying the size multiplier.

TEAM RESPONSE

Fixed in PR 13 (`ae99430` , `7badd63`).

M-009

An attacker can skew funding with unfillable resting quotes that still set the midpoint

MEDIUM FIXED

ASSET

`src/perps/PerpPool.sol:1140-1159`

Description and impact

An attacker can leave unfillable quotes on two wallets with no collateral and no position behind either one. Those quotes hold the funding midpoint away from index and charge an unrelated counterparty inflated funding until someone clears them. Funding midpoint comes from best bid and ask on the book. Matching cancels a maker who cannot meet initial margin for the fill, but funding still reads those levels when they are not expired:

```
OrderIndex bestBid = _getBestNonExpiredOrder(ob.bidsIndex);
OrderIndex bestAsk = _getBestNonExpiredOrder(ob.asksIndex);
midpoint = (bidPrice + askPrice) / 2;
```

Mid can sit far above index while the quotes are not executable.

1. Index is 100. The attacker shorts on Alice and longs on Bob at 100. Alice rests a post-only bid at 130. Bob rests a post-only ask at 140. Both are reduces.
2. A third wallet they control rests a sell at 135 and a buy at 135.
3. Alice buys the sell and closes the short. Bob sells into the buy and closes the long. Those 135 orders fill and leave the book. The 130 bid and 140 ask stay, but they are now opens, not reduces.
4. They withdraw every last USDso. A taker hitting those quotes would see them cancelled. Funding still reads 130 and 140 as best bid and ask, putting mid at 135 against a 100 index.
5. Carol, long against that market, is charged the inflated premium on the next funding tick.

Carol pays skewed funding to the short side because the midpoint path never applies the fillability rules that matching would. The attacker keeps orders alive that no taker can actually execute against.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_PhantomQuotesControlFundingMid.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_PhantomQuotesControlFundingMid.t.sol -vv`

Recommendation

When computing the funding premium, walk best bid and best ask past makers that matching would cancel, and use the first quotes that would actually fill. If that walk is long, cancel those makers as you skip them so the next update is cheap. A tighter book fix is to cancel leftover zero-lock orders once a separate fill has flattened the position and they are no longer reduces.

TEAM RESPONSE

Fixed in PR 13 (`b8f141a`, `100dabe`).

AUDITOR NOTE

The team has accepted that many minimum-size quotes can still push one side of the funding premium to zero, which pulls the premium toward neutral.

M-010

A liquidated trader can keep collateral Stage 6 should burn when multi-page ADL resumes from a stale snapshot

MEDIUM FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1404-1427`

`src/perps/liquidation/LiquidationEngine.sol:1652-1659`

Description and impact

A liquidated trader can keep collateral that Stage 6 should have burned when a multi-call ADL session resumes from a stale snapshot. The first page freezes bankruptcy prices and records free collateral. A later call clears the session only if that stamp changed. Position size and pending funding can change the real hole without touching that stamp. On resume, settlement reads the live remaining size but still closes at the frozen bankruptcy price:

```
if (balance != existing.snapshotBalance) {
    _clearAdlSession(account);
    return false;
}
_resumeAdlSession(bank, account);
bank.adlSettle(ctx.account, holder, ctx.pool, uint128(settled), ctx.frozenBP);
```

1. A keeper liquidates Alice. Stage 6 freezes bankruptcy prices for her 10-unit long, stamps her free collateral, and closes against the first counterparty before the page budget ends.
2. Between pages, Alice is owed 20 USDso of funding. That credit is still pending, so the stamp does not move.
3. The next call resumes, pays her the 20, then closes the remaining size at the prices frozen in step 1.
4. She ends flat with leftover cash she can withdraw. The same run with no pause burns her to zero.

Alice walks away with collateral a fresh snapshot would have burned. The same stale stamp also survives a partial close between pages, so remaining size can settle at prices that no longer match the leftover position.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_AdlResumeStaleSnapshotLeavesSurplus.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_AdlResumeStaleSnapshotLeavesSurplus.t.sol -vv`

Recommendation

Invalidate the ADL session when position size, entry price, or pending funding diverges from the frozen basis. Alternatively, settle live funding first and re-snapshot prices on each resume before closing more size.

TEAM RESPONSE

Fixed in PR 19 (`9a41ec4`), with the flat-balance sweep in PR 16.

AUDITOR NOTE

A paged session now restarts on fresh prices when funding comes due between pages. The team has accepted that a change in position size between pages is not detected, since it needs a close at exactly the entry price within a session window of about ten seconds.

M-011

A bankrupt trader can keep Stage 6 burn budget when it lands on a leg with no profitable opposite holders

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:1927-1944` `src/perps/liquidation/LiquidationEngine.sol:1622-1632`

Description and impact

A bankrupt trader can keep collateral Stage 6 meant to burn when budget is assigned to a leg with no profitable opposite holders. Stage 6 splits free collateral across open legs before it checks whether each leg can deleverage:

```
allocs = PerpMath.adlAllocations(signedBalance, markNotionals, capacities);
if (holder == ctx.account || pos.size == 0
    || PerpMath.unrealizedPnl(pos.size, pos.avgEntryPrice, markPrice, oneBase) <= 0) {
    ++i;
    continue;
}
```

Each leg burns only against opposite holders with positive unrealized profit at the live mark. Unused budget from a skipped leg never moves onto a closable market.

1. Alice is long 20 on A and short 10 on B. Both entered at 100, both marks are 50, and her free collateral is still 80.
2. Stage 5 skips because the free balance is positive. Stage 6 allocates about 53 to the underwater long and 27 to the short.
3. ADL closes the long and burns the 53. No opposite long on B is in profit at 50, so the 27 share never burns.
4. Alice is solvent with the short still open. She buys 10 at 50 to close it, which turns the 500 of open profit into cash. She withdraws that 500 plus the leftover 27.

Stage 6 never spends the share it parked on the skipped leg, so Alice keeps it. If the free balance had already been negative, that unused share would stay as a hole on the ledger.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_Stage6UnderBurnsOffsettingOpenProfit.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_Stage6UnderBurnsOffsettingOpenProfit.t.sol -vv`

Recommendation

Allocate Stage 6 burn only across legs that can deleverage under profit-only rules, and reassign unused share from skipped legs onto closable markets before the session finishes.

TEAM RESPONSE

Fixed in PR 15 (`6a2df8a`).

M-012

A single resting order at the open-interest cap can revert every opening taker instead of being skipped the way an over-cap maker is

MEDIUM FIXED

ASSET

`src/perps/PerpPool.sol:794-805` `src/perps/PerpPool.sol:1229-1271`

Description and impact

A flat trader can park one opening quote at the front of a capped market and make every opening taker on the other side revert. Reducing liquidity behind that quote stays unreachable. The book already cancels resting makers that would breach `maxPositionSize` before fill, but open interest has no matching pre-fill skip. The cap is checked only after settlement, so an OI-growing maker at the best price reverts the whole taker transaction:

```
require(
    $.totalOpenInterest <= $.perpPoolParameters.maxOpenInterest || $.totalOpenInterest <= oldOI,
    OpenInterestCapExceeded()
);
```

With open interest at the cap, a 100 price, and 10-unit clips:

1. Alice has no position and rests a 10-unit bid at 100. A fill would open a new long and push open interest past the cap.
2. Bob is already short 10 and rests a 10-unit bid at the same price behind her. A fill would only reduce his short.
3. Carol sends an immediate-or-cancel ask to open a 10-unit short. Matching reaches Alice first and settlement reverts with `OpenInterestCapExceeded`.

Every opening taker bounces off Alice's single quote while Bob's reducing liquidity sits one slot behind it. Closes and liquidations still work, so Alice blockades new exposure rather than freezing the whole market.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_OiCapToxicRestingOpener.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_OiCapToxicRestingOpener.t.sol -vv`

Recommendation

In `_shouldCancelMakerPreFill`, cancel OI-growing resting openers that would breach `maxOpenInterest`, then continue matching.

TEAM RESPONSE

Fixed in PR 13 (`288f22c`).

M-013

A Stage 4 bidder can keep close-out margin when the winning market is taken first

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:794-801`

Description and impact

A registered bidder can be paid close-out margin in the middle of Stage 4, and Stage 5 never takes it back. Stage 4 walks the liquidated account's markets in list order. After each takeover it pays the bidder from whatever free collateral exists at that moment:

```
int256 fromBalance = $.accounts[from].unlockedCollateralBalance;
uint256 transferAmount = fromBalance > 0 ? minUint256(uint256(fromBalance), cmRequired) : 0;
_transferBalance(from, to, transferAmount);
```

If the winning market is first, that takeover books a profit and the balance is briefly positive, so close-out margin moves. The losing market then books a larger loss. Stage 5 sizes the Insurance Fund draw from the realized trading loss only. It does not see cash that already left.

1. Alice has one account: a 10-unit short on A and a 10-unit long on B, both from 100, mark 50. A is first in her list. Free collateral is 0, so the underwater long puts the account into close-out. Bob takes every remaining leg.
2. Stage 4 takes A first. The short pays her 500. Balance is positive, so 6 of close-out margin on that 500 notional goes to Bob. She has 494 left.
3. Stage 4 takes B. The long costs her 520. She is at -26. Stage 5 pays 20 from the Insurance Fund, which is only the net trading loss.
4. Alice is flat at -6. Bob keeps the 6.

If B had been first, her balance would stay non-positive, no close-out margin would move, and Stage 5 would clear her to zero. Alice can put the winner first by closing and reopening so it sits at the front of her list. The same payment also uses the bidder's gross takeover size, so a merge that flattens Bob can still pay him close-out margin as if the position remained.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_Stage4WinnerFirstCmResidual.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_Stage4WinnerFirstCmResidual.t.sol -vv`

Recommendation

Hold the close-out margin transfer until every Stage 4 takeover finishes, so it is paid from the true post-liquidation balance. Size it from the bidder's position after the merge, and skip it when the merge leaves them flat.

TEAM RESPONSE

Fixed in PR 13 (`22812db`).

M-014

An unrelated counterparty can absorb Stage 3 liquidation fees that never enter the Insurance Fund draw

MEDIUM ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:774-781` `src/perps/liquidation/LiquidationEngine.sol:956-961`

Description and impact

A liquidation's taker fee lands with the fee recipient as ordinary revenue, but the loss it creates never gets attributed to the liquidated account. That fee residual falls through to Stage 6 and gets deleveraged out of an unrelated profitable counterparty. Stage 3 IOC closes run through the same fill path as ordinary trades so after `settleTrade`, `_chargeFees` debits the liquidatee as taker, yet the engine books only trade PnL into the accumulator:

```
bank.settleTrade(takerOrder.owner, takerSizeDelta, fillPrice);
_chargeFees(takerOrder.owner, makerOrder.owner, quantity, fillPrice, bank, $);
// ...
_bookRealized(ctx, positions[i].pool, result.realizedPnl);
```

Stage 5 draws only that booked loss, so the fee hole falls to Stage 6. The design is explicit:

> Stage 3 charges no explicit liquidation penalty

However, ordinary taker fees still debit the liquidatee on these immediate-or-cancel closes.

1. Alice is cross-margin long on pools A and B. Bob is short on B. Mark is 95.
2. Stage 3 IOC-closes Alice on pool A. Trade PnL is small, but the taker fee of about 0.475 still debits Alice and credits the fee recipient.
3. Stage 5 attributes only the booked trade PnL.
4. Stage 6 ADLs about the same fee amount onto Bob's pool B short while the recipient keeps the credit.

Bob pays liquidation fees that never entered Stage 5 attribution, and Alice finishes flat on this path.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_Stage3FeeResidualAdlSocialize.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_Stage3FeeResidualAdlSocialize.t.sol -vv`

Recommendation

Exempt liquidation IOC fills from protocol fees, or book fee debits into the Stage 3 realized accumulator so Stage 5 can absorb them before Stage 6 runs.

TEAM RESPONSE

Acknowledged in PR 13, with the bound restated and tested in PR 19 (`4367da8`).

AUDITOR NOTE

The team has accepted that an ADL counterparty can still pay the liquidated account's taker fee, up to one Stage 3 sale's fee, when that account was already below zero. The team states the taker fee is currently 0 on every live market.

M-015

A voucher holder can unlock house principal by naming themselves as their own builder

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:1583-1588`

Description and impact

A voucher holder can name themselves as their own builder, pay themselves a builder fee from house credit, and withdraw that fee immediately. `approveBuilder` blocks only reserved protocol recipients and does not reject `builder == order.owner`. The fee path debits and credits the same voucher account as two separate steps, and the floor follows the temporary debit:

```
int256 economicBalance = newBalance + int256($.accounts[user].lockedCollateral);
if (economicBalance < int256(floor)) {
    $.nonWithdrawableCredit[user] = economicBalance > 0 ? uint256(economicBalance) : 0;
}
```

The floor never rises again when the self-paid builder fee returns.

1. Alice claims a 1,000 USDso voucher. Her floor is 1,000 and she can withdraw nothing.
2. She approves her own address as builder at 100 basis points.
3. Alice buys 1,000 USDso of notional with an immediate-or-cancel order naming herself as builder.
4. The 10 USDso builder fee drops her floor to 990, credits the same 10 USDso back, and becomes withdrawable. Repeating the fill and withdrawal walks the floor to zero.

Alice converts campaign budget into withdrawable cash with no price movement, and the house funding the campaign loses every USDso she extracts.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_SelfBuilderErasesVoucherFloor.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_SelfBuilderErasesVoucherFloor.t.sol -vv`

Recommendation

Reject `builder == payer` in `approveBuilder` and again in `chargeBuilderFee`, since an order can carry an older approval. Skip the voucher-floor ratchet when `from == to`, because a self-transfer preserves account value.

TEAM RESPONSE

Fixed in PR 15 (`abca025`) and PR 16, with a test in PR 19 (`db0e8c6`).

M-016

A liquidatable trader can claim a voucher, withdraw their deposit, and leave the open loss on house credit

MEDIUM FIXED

ASSET

`src/perps/voucher/VoucherDistributor.sol:116-126` `src/perps/MarginBank.sol:214-218`

Description and impact

A liquidatable trader can claim a voucher, withdraw the deposit that was backing the open loss, and leave that loss on house credit. Claim does not care that she is already down. The credit raises equity above maintenance margin, so a keeper can no longer liquidate her. The withdrawal floor only compares realized balance to the voucher floor, so her own deposit looks free while the position is still losing:

```
bytes32 id = _claimId($, msg.sender);
require(!$.claimed[id], AlreadyClaimed());
IMarginBank($.marginBank).creditAccount(msg.sender, grant);
```

Withdraw then treats everything above that floor as hers:

```
uint256 floor = $.nonWithdrawableCredit[msg.sender];
if (floor > 0) {
    int256 bal = $.accounts[msg.sender].unlockedCollateralBalance;
    uint256 free = bal > int256(floor) ? uint256(bal - int256(floor)) : 0;
    require(amount <= free, WithdrawalBelowCreditFloor());
}
```

1. Alice is long 10 at 100 against 60 USDso. Mark falls to 95.50. She is down 45 and liquidatable.
2. She claims 1,000 USDso. Equity rises to 1,015, so a keeper can no longer liquidate her.
3. Realized balance is 1,060 and the floor is 1,000, so she withdraws all 60 USDso.
4. She closes at 95.50. The 45 loss hits house credit. Expiry returns 955 USDso to the house.

Alice keeps her 60 USDso. The campaign is out 45. If she had claimed while flat, that 45 would have come from her deposit and the house would have recovered the full 1,000.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_VoucherClaimRescuesLiquidatableAccount.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_VoucherClaimRescuesLiquidatableAccount.t.sol -vv`

Recommendation

Require the claimant to be flat before granting credit. A flat account has no open loss to move onto house credit, and no live position on a market or leverage the campaign would have blocked.

TEAM RESPONSE

Fixed in PR 15 (`926dd4e`).

M-017

A voucher holder can poison a price level with one resting order so that every taker and every liquidation there reverts

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:1756-1758` `src/perps/PerpPool.sol:1267-1272`

Description and impact

A voucher holder can turn one resting order into a hard revert for every taker that reaches its price, including Stage 3 liquidations. Before a maker fills, `PerpPool` calls `bank.meetsIMForFill` without catching reverts. For voucher accounts the bank enforces market curation with a hard require:

```
if (additionalSize > 0 && $.nonWithdrawableCredit[account] > 0) {
    require($.voucherMarketAllowed[pool], VoucherMarketNotAllowed());
}
```

That require only runs when the fill would open or increase. Alice's ask starts as a reduce against her long, so a taker would not trip it. While that ask still rests, she closes the long on another order. The same ask would now open a short. After she claims, the uncaught revert fires in the taker's transaction and Stage 3 never reaches later stages.

1. Alice opens a 1-unit long at 120 on a market outside the voucher allowlist, then rests a 1-unit ask at 120.
2. While the ask still rests, she closes the long one tick lower. She is flat. The ask would now open a short.
3. She claims a voucher.
4. Bob is short. Mark rises to 120 and he is liquidatable. Stage 3 buys to close him at 120, hits Alice's leftover ask, and reverts with `VoucherMarketNotAllowed`.

The leftover ask stays until Alice cancels or the owner adds the market to the allowlist. Expiry cannot take the voucher back while that ask still rests, so Bob's liquidation keeps reverting.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_VoucherClaimArmsPrefillPoison.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_VoucherClaimArmsPrefillPoison.t.sol -vv`

Recommendation

Wrap `bank.meetsIMForFill` in `try/catch` and treat any catch as a maker cancel. Let expired voucher recovery cancel resting orders before the flatness check, so a pinned order cannot keep campaign budget stuck.

TEAM RESPONSE

Fixed in PR 13 (`bf32ea8`).

M-018

A dropped Schedule tick can leave the stop registry unable to restart automation

MEDIUM FIXED

ASSET

`src/vendor/somnia/interfaces/SomniaExtensions.sol:227` `src/perps/PerpStopOrderRegistry.sol:273-275` `src/perps/PerpStopOrderRegistry.sol:475-479`

Description and impact

Every Schedule the registry creates is not guaranteed. Somnia can skip the callback when the reactivity queue is full, the block is full, or the registry cannot pay the prepaid gas at fire time. The handler never runs, so it never arms the next tick and never clears the stored id.

```
isGuaranteed: false,
```

Restart and shutdown both treat that stored id as proof the chain is still live. `createSubscription` refuses a second chain. `removeSubscription` has to unsubscribe an id the precompile already deleted, and that call can revert:

```
require($.activeSubscriptionId == 0, SubscriptionAlreadyActive());
SomniaExtensions.unsubscribe(subscriptionId);
$.activeSubscriptionId = 0;
```

1. The registry runs a 60 second chain. Alice has a stop-loss armed against it.
2. The due tick is dropped. The handler never runs. The stored id stays 1.
3. `createSubscription` reverts with `SubscriptionAlreadyActive`. `removeSubscription` reverts with `UnsubscribeFailed`.
4. `createPendingOrder` still takes Alice's prepaid SOMI, because the pointer is not zero.

Paid automation is gone until someone clears the id from outside. Traders can still call `poke()`.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_DroppedScheduleWedgesStopChain.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_DroppedScheduleWedgesStopChain.t.sol -vv`

Recommendation

Do not treat the stored pointer as proof of life. Have `createSubscription` and `removeSubscription` check the precompile first and self-heal a dead id, or give the owner an unconditional clear that does not depend on `unsubscribe`. Gate that path so it cannot unlock mass stop cancellation while a chain should still run.

TEAM RESPONSE

Fixed in PR 16.

AUDITOR NOTE

The team's recovery path is `reviveSchedule`, used while stops are armed. Removing and re-creating the subscription instead leaves a window in which anyone can cancel the armed stops.

M-019

Alice can withdraw before her funding is booked, so Bob is paid from other depositors

MEDIUM FIXED

ASSET

`src/perps/MarginBank.sol:220-230` `src/perps/MarginBank.sol:1871-1879`

Description and impact

Alice can take out her full deposit while the funding she owes is still unbooked. `withdraw` checks that pre-funding balance, pays her, and only then settles funding. Paper profit still covers initial margin after her realized balance goes negative:

```
_debitBalance(msg.sender, amount);
_settleFunding(msg.sender, activePools[i]);
require(equity >= int256(totalIMRequired), InsufficientMarginAfterWithdrawal());
```

Totals count only positive balances, so her debt does not reduce what the bank claims to back. Bob can withdraw the matching credit as real tokens.

1. Alice deposits 60 USDso and buys 10 at 100. Bob is short.
2. The mark is 120. A day of funding is due. Alice owes 0.3 USDso.
3. She withdraws her 60 USDso. The bank pays her, then books the 0.3. Her balance is -0.3. Paper profit of 200 (price went from 100 to 120) still covers initial margin.
4. Bob withdraws the 0.3 USDso. Those tokens come from other depositors.
5. The mark falls to 100. A keeper liquidates her. She finishes flat with -0.3 still on the ledger. No insurance tier is charged.

If she had closed at 120, the 200 profit would have paid the 0.3 and the ledger would heal. After this liquidation the 0.3 is uncollected, and Bob's withdrawal stays with other depositors.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_WithdrawBeforeFundingSettleLeavesHole.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_WithdrawBeforeFundingSettleLeavesHole.t.sol -vv`

Recommendation

Settle funding first in `withdraw`, before the debit and the health check, so every gate reads post-funding state. Route leftover negative balance after liquidation into insurance so the shortfall is recorded.

TEAM RESPONSE

Fixed in PR 13 (`cc95c88`).

M-020

A healthy short can absorb a bankrupt long's leftover loss when Stage 6 closes that long with no price cap

MEDIUM ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:1998-2008` `src/perps/MarginBank.sol:810-862`

Description and impact

Stage 6 can move a bankrupt long's leftover loss onto a healthy short. It picks a close price that zeroes the bankrupt, then settles both sides there. A bankrupt short is capped at entry times size. A bankrupt long is not:

```
return binding
    ? (uint256(avgEntryPrice) * absSize) / oneBase
    : ((uint256(type(uint128).max) - uint256(avgEntryPrice)) * absSize) / oneBase;
```

Stage 5 pays a leftover from a funded insurance tier first. This path needs that tier empty, so Stage 6 still has the loss to place.

1. Bob deposits 600 USDso and opens two longs at 100: 100 units in a busy market, and 1 unit in a second market where Alice is short 1 with 200 USDso.
2. Both marks fall to 50. Stage 3 sells Bob's 100-unit long at 46. That books a 5,400 loss and leaves him at -4,800 with the 1-unit long still open.
3. Stage 6 zeroes that -4,800 on the leftover long. The close price is 4,900, against a mark of 50.
4. Alice's short closes at 4,900 too. She was up 50 at the mark. She books a 4,800 loss.

Alice finishes flat at -4,600. She cannot withdraw the 200 she deposited. She cannot open a new order until she posts that 4,600. No later liquidation can collect it.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_AdlLongClawbackUncappedDestroysCounterparty.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_AdlLongClawbackUncappedDestroysCounterparty.t.sol -vv`

Recommendation

Cap a bankrupt long the same way a bankrupt short is already capped, so leftover loss stays on the bankrupt. Or cap Stage 6 at the counterparty's mark profit.

TEAM RESPONSE

Acknowledged in PR 19. The counterparty floor is in PR 16, and the routes in this report are closed in PR 19 (`5cc97e9`).

AUDITOR NOTE

The team has accepted that the claw-back stays uncapped by design. A counterparty can no longer be taken below zero, but a hole booked another way can still cost a healthy counterparty up to their whole equity.

M-021

A quiet fallback feed can halt the market so underwater accounts cannot be liquidated

MEDIUM ACKNOWLEDGED

ASSET

`src/oracle/AggregatorOracle.sol:131-142` `src/oracle/AggregatorOracle.sol:152-164`

Description and impact

A quiet fallback can halt a market even when the primary feed is fresh. The divergence breaker fetches both feeds' timestamps and never uses them. It compares prices only, so a live primary is checked against a fallback's hours-old quote and a normal move reads as a feed fault:

```
(bool pOk, uint256 pPrice, uint256 pAt) = _tryRead(_primary, isIndex);
(bool sOk, uint256 sPrice, uint256 sAt) = _tryRead(_secondary, isIndex);
if (pOk) {
    if (dev != 0 && sOk && _diverges(pPrice, sPrice, dev)) revert FeedsDiverged(pPrice, sPrice);
```

A feed counts as available whenever it returns a non-zero price without reverting, no matter how old. Per-market staleness runs later, after the router has already reverted. That revert stops orders, fills, funding, and every liquidation stage.

1. A market uses the aggregator with a 1% divergence threshold and a fallback feed for redundancy.
2. Alice is short 10 from 100 with 60 USDso. The mark moves to 106. The primary publishes it on time and she is below maintenance margin.
3. The fallback has not updated for three hours and still quotes 100.
4. The breaker compares 106 to 100, reverts every oracle read, and Alice cannot be liquidated or withdraw.

While the halt lasts, losses keep growing. Clearing the fallback restores service because a lone fresh primary has nothing to compare against. `updateOracle` does not help: it settles funding first and that call reads the broken oracle too. Alice keeps exposure she should have lost, and counterparties absorb any extra deficit.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_LaggingFallbackFeedHaltsMarket.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_LaggingFallbackFeedHaltsMarket.t.sol -vv`

Recommendation

Only compare quotes of similar age. Treat an old fallback as unavailable rather than as evidence against the primary, for example by requiring `|pAt - sAt| <= maxAgeSkewSec` before calling `_diverges`.

TEAM RESPONSE

Acknowledged in PR 13, with the setting tested in PR 19 (`457e882`).

AUDITOR NOTE

The team has accepted that a lagging fallback feed can halt the market and block liquidation whenever `deviationBps` is above 0. It keeps `deviationBps` at 0, which also turns off the divergence check.

M-022

Stage 3 closes the whole position when a fifth of it would restore margin

MEDIUM ACKNOWLEDGED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:948-966`

`src/perps/liquidation/LiquidationEngine.sol:2102-2106`

Description and impact

In the `PartialLiquidation` tier, equity sits below maintenance margin but above close-out. Stage 3 is meant to sell only enough size to bring the account back to health. Instead it always sizes the closing IOC at the full position:

```
uint128 alignedQty = _getAlignedQuantity(pos.pool, uint128(absInt128(pos.size)));
if (alignedQty == 0) {
    emit PositionSkippedBelowMinQuantity(account, pos.pool, pos.size);
    return result;
}
```

`_getAlignedQuantity` only rounds that size down to the pool's lot size. It never asks how many units maintenance margin actually needs. The engine can stop after the fill before moving to the next market, but it cannot shrink the order. An account with one position gets closed out in a single IOC, and the trader pays the liquidation discount on every unit sold.

1. Alice buys 10 units at 100 and deposits about 63 USDso. The mark falls to 96. She sits in `PartialLiquidation`, not full close-out.
2. One bid rests at 94.85, the lowest price Stage 3 can hit at her health level.
3. Anyone calls `liquidate`. Stage 3 sells all 10 units at 94.85. Alice ends flat with about 11 USDso left.
4. Selling only 2 units at the same price would have restored maintenance margin and left her with 8 units and about 21 USDso.

The engine took the full close path when a one-fifth sale was enough. Alice keeps about 9 USDso less than she would have after the smaller sale. The maker at 94.85 keeps that gap as extra liquidation discount.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_Stage3ClosesWholePositionInPartialTier.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_Stage3ClosesWholePositionInPartialTier.t.sol -vv`

Recommendation

Keep the stop between markets. Also size each IOC inside one market to the smallest lot-aligned quantity that restores maintenance margin at the limit price, and use the full position only when no smaller size works or the result is below `minQuantity`.

TEAM RESPONSE

Acknowledged in PR 13.

AUDITOR NOTE

The team has accepted that Stage 3 sends the whole position in every tier. An account that needed only a partial close still pays the spread and penalty on the whole position.

M-023

An under-margined but solvent backstop cannot be reduced until it is bankrupt

MEDIUM ACKNOWLEDGED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:784-791`

`src/perps/liquidation/LiquidationEngine.sol:1328-1333`

Description and impact

Registered liquidation bidders skip the ordinary waterfall and go straight to Stage 6. Design skips Stages 3, 4, and 5 on purpose (D1 / LW-3): a backstop cannot bid on itself and was not meant to trade on the public book. The residual gap is what happens while that backstop is still solvent. Stage 6 only runs when balance or equity is negative:

```
if ($.isBidder[account]) {
    _runStage6AndEmit(bank, account, 0, LiquidationStage.OrderCancellation, statusBefore);
    return;
}
if (signedBalance >= 0 && bank.getAccountEquity(account) >= 0) { /* no bad debt */ }
```

A bidder below maintenance margin but still above zero sits in a dead zone. Stage 3 never runs because of the bidder branch. Stage 6 will not run because there is no bad debt yet. Nothing shrinks the position while it stays registered, and it can keep absorbing new takeovers on an under-margined book.

1. The backstop holds a long of 10 from 100 with 60 USDso after a takeover. The mark falls to 95. Equity is 10 against 23.75 of maintenance margin.
2. A bid rests at 95 for the full size. An ordinary account would close on it.
3. Anyone calls `liquidate`. The position is untouched until the mark falls to 90 and the backstop is 40 USDso underwater.
4. Stage 6 finally runs. Bob, short 10 from 100, is force-closed at the bankruptcy price and gives up 40 of his mark profit. The funded insurance tier is charged nothing.

The small loss could have closed on the book or hit the insurance fund while the backstop was still solvent. Instead it waits until bankruptcy and lands on profitable traders.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_BidderFallThroughSkipsBookAndInsuranceFund.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_BidderFallThroughSkipsBookAndInsuranceFund.t.sol -vv`

Recommendation

Keep the Stage 4 skip, but add a reduction path for under-margined solvent bidders: run Stage 3 (and Stage 5 if a hole remains), or refuse new takeovers and clear the bidder flag once equity falls below maintenance so the ordinary waterfall can finish the job.

TEAM RESPONSE

Acknowledged in PR 13, with the no-bidder case tested in PR 19 (`13df388`).

AUDITOR NOTE

The team has accepted that once the owner registers a bidder, an under-margined but solvent backstop position cannot be reduced until it is bankrupt. It does not occur while no bidder is registered.

M-024

A trader can be stuck with a position she cannot close after a partial fill leaves her under the minimum order size

MEDIUM FIXED

ASSET

`src/orderbook/OrderBook.sol:510-513` `src/perps/liquidation/LiquidationEngine.sol:2102-2106`

Description and impact

Alice can be left holding a long smaller than the market's minimum order size, and then neither she nor Stage 3 can close it. The book only checks the minimum when someone places an order. It does not check it on each fill, so a legal bid can take a smaller slice and leave her under the floor. Selling the leftover size reverts. Selling the full minimum closes the long and opens a short that is also under the minimum. Getting out that way needs spare margin for the flip, which an under-margined account does not have.

The book enforces the floor on every new order:

```
require(quantity >= $.minQuantity, QuantityBelowMinimum(quantity, $.minQuantity));
```

Stage 3 uses the same floor and skips her:

```
if (alignedQty < obParams.minQuantity) return 0;
```

1. The market requires every order to be at least 5 units. Bob rests a bid for 5 at 100. Alice rests a bid for 5 at 99 behind him.
2. A seller hits the book with a legal 7-unit sell. Bob fills. Then 2 of Alice's 5 fill. She cancels the leftover bid and holds a long of 2.
3. Selling 2 reverts. Selling 5 closes the long and opens a short of 3, which is also under the minimum. Escaping through a temporary 5-unit short needs spare margin she does not have.
4. The mark falls to 88 and she drops below maintenance. A bid rests at 88. Stage 3 skips her because 2 is below the minimum.

She keeps exposure she cannot exit. If the mark keeps falling, Stage 6 can still force-close her against profitable traders, but only after Stage 3 ignored a live bid. While she is only under-margined, the owner also cannot delist the market because she still counts as active.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_SubMinimumPositionCannotBeClosed.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_SubMinimumPositionCannotBeClosed.t.sol -vv`

Recommendation

Exempt a reduce that is no larger than the holder's current position from the minimum size check, so the carve-out cannot open new size. Apply the same rule in Stage 3 so a sub-minimum position can be closed on the book.

TEAM RESPONSE

Fixed in PR 13 (`a442b30`).

M-025

A covered market's insurance tier can pay for an uncovered market's losses

MEDIUM FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1141-1142`

`src/perps/liquidation/LiquidationEngine.sol:1806-1812`

Description and impact

Markets without an insurance tier are meant to eat their own losses. The engine comments say uncovered losses stay residual. The charge math does not match. `coveredNetLoss` sums realized losses only on covered markets, while `deficit` is the trader's full negative balance, uncovered losses included:

```
uint256 coveredNetLoss = coveredNet < 0 ? uint256(-coveredNet) : 0;
uint256 chargeable = deficit < coveredNetLoss ? deficit : coveredNetLoss;
```

Taking the smaller of the two spends the trader's collateral on the uncovered market first and charges the covered tier for what is left. When the covered market's own loss is larger, that tier pays for both.

1. Alice deposits 150 USDso. Market A sits on tier 1 with 1,000 USDso funded. Market B has no tier.
2. She goes long 10 at 100 on A and long 5 at 100 on B. Both marks fall to 80. Stage 4 takes both at 78.40.
3. She realizes -216 on A and -108 on B. Her 150 is gone and the hole is 174.
4. `coveredNetLoss` is 216 and the deficit is 174, so tier 1 is charged 174 and Alice ends flat.

Tier 1 paid 58 more than market A's own share. Market B's loss was never left residual. Capital A's traders funded as their backstop paid for a market it was never earmarked for.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_UncoveredMarketLossChargedToACoveredTier.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_UncoveredMarketLossChargedToACoveredTier.t.sol -vv`

Recommendation

Split the deficit before charging so each market's loss carries its own share of the trader's collateral. Attribute the uncovered slice first, then set `chargeable` from the covered share of the remaining deficit rather than `min(deficit, coveredNetLoss)`.

TEAM RESPONSE

Fixed in PR 19 (`22b6631`).

M-026

An index-feed fault freezes liquidation even when the mark feed is healthy

MEDIUM FIXED

ASSET

`src/perps/PerpPool.sol:1093-1097` `src/perps/MarginBank.sol:1699-1714`

Description and impact

Liquidation needs the account balance, the mark price, and the stored margin rates. It does not need the index price for that decision. Whole-account health still reads the index leg through the funding projection. That path handles a stale or zero index quote but not a reverting one:

```
(uint256 indexPrice, uint256 updatedAt) = IOracle($.oracle).getPrice();
if (block.timestamp - updatedAt > fp.maxOracleStalenessSec || indexPrice == 0) {
    return $.cumulativeFundingPerUnit;
}
```

Index and mark route through separate aggregator reads. The index leg can revert with `FeedsDiverged` while the mark leg serves a good price. That revert bubbles out of the projection and blocks liquidation even though every input the waterfall actually uses is available.

1. Alice deposits 60 USDso and buys 10 at 100. Her counterparty is short 10.
2. The mark falls to 80. Alice is 140 USDso underwater. Both mark feeds still agree at 80.
3. The two index quotes drift to 80 and 100 and trip the index divergence breaker.
4. `liquidate(alice)` reverts. Reading her equity reverts too, and so does her counterparty's withdrawal.

Alice keeps a position that should have closed. The profitable counterparty cannot withdraw, and the bankrupt account is not being wound down. Whoever eventually absorbs the hole pays for the outage.

Proof of Concept

POC FILE

`findings/pocs/mediums/PoC_IndexLegFailureFreezesMarkPricedLiquidation.t.sol`

COMMAND

`FOUNDRY_PROFILE=pocs forge test --match-path findings/pocs/mediums/PoC_IndexLegFailureFreezesMarkPricedLiquidation.t.sol -vv`

Recommendation

Treat a reverting index read the same way a stale one is already treated, so the projection falls back instead of aborting. Apply the same fallback to dynamic initial margin by returning the static `initialMarginBps` when the index read fails.

TEAM RESPONSE

Fixed in PR 15 (`dee331b`).

L-001

A trader who shrinks a flip order to its reducing part can leave collateral locked with nothing left to release it

LOW FIXED

ASSET

`src/perps/PerpPool.sol:874-876` `src/perps/PerpPool.sol:902-907`

Description and impact

Alice can shrink a flip order until only the reducing part remains. The order then needs no margin, but the collateral she locked earlier stays locked until she cancels or it fills.

`src/perps/PerpPool.sol:902-907`

```
uint256 quantityBefore = order.quantityRemaining + quantity; // remaining before this fill
uint256 increasingRemaining = info.increasingQuantityRemaining;
uint256 reducingRemaining = quantityBefore > increasingRemaining ? quantityBefore - increasingRemaining : 0;
uint256 reducingConsumed = quantity < reducingRemaining ? quantity : reducingRemaining;
uint256 increasingConsumed = quantity - reducingConsumed;
if (increasingConsumed == 0) return; // fill landed entirely in the reducing band -> release nothing
```

The unlock helper only releases when increasing size is consumed, so a fill that only reduces leaves the old lock stranded.

Recommendation

After a reduction, recompute what increasing exposure the order still carries and release the excess lock when the remainder is purely reducing.

TEAM RESPONSE

Fixed in PR 14 (`19856a8`).

L-002

A trader's stop-loss cancel refund can become unclaimable because the chain spends mixed SOMI against the raw balance

LOW FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:281-287` `src/perps/PerpStopOrderRegistry.sol:367-372` `src/perps/PerpStopOrderRegistry.sol:538-543`

Description and impact

The stop-loss chain starts against the registry's raw SOMI balance, which mixes the owner's float with SOMI reserved for live stops and SOMI owed back to traders.

```
require(address(this).balance >= activationFloor, InsufficientActivationBalance());
```

Alice can cancel a stop, get credited a refund, and still fail to claim it because the chain kept spending that mix. If the float sits near the 32 SOMI floor, Alice claiming her own refund can drop the registry under that floor and stop the chain.

Recommendation

Gate starting the chain, rescheduling, and every payout on the balance covering reserved plus unclaimed SOMI plus the minimum chain balance, rather than on the raw balance against that minimum alone.

TEAM RESPONSE

Fixed in PR 14 (`51c50fe`).

L-003

A liquidatable wallet can unlink by topping up for one transaction, then fall straight back

LOW ACKNOWLEDGED

ASSET

`src/perps/liquidation/PerpsUnlinkGuard.sol:36-39` `src/registry/LinkedWalletRegistry.sol:125-130`

Description and impact

Alice is below maintenance margin. She deposits just enough to pass the unlink check, unlinks in the same transaction, and can be unhealthy again right after. The guard only reads her margin status in that instant.

```
return uint8(marginBank.getMarginStatus(a)) < uint8(MarginStatus.PartialLiquidation);
```

A later withdrawal still has to pass initial margin, so she has to leave real capital in for the length of the top-up. Unlinking also worsens her own auto-deleveraging rank.

Recommendation

If unlinking is meant to show lasting health, require equity above initial margin for a cooldown, or remember liquidatable status from when the wallets linked.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-004

A liquidatable wallet can unlink when the oracle is stale because the guard fails open on any health-read error

LOW ACKNOWLEDGED

ASSET

`src/registry/LinkedWalletRegistry.sol:118-129` `src/perps/liquidation/PerpsUnlinkGuard.sol:36-39`

Description and impact

When a stale mark makes the unlink guard revert, the registry's empty catch treats that as permission to unlink, so the exact wallet the guard was written to stop can walk away from the link during that window.

`src/registry/LinkedWalletRegistry.sol:125-129`

```
try IUnlinkGuard(guard).canUnlink(msg.sender, counterparty) returns (bool allowed) {
    require(allowed, UnlinkBlockedByGuard());
} catch {
    // Treat a guard fault as "unlink permitted".
}
```

Damage is limited. Unlinking only worsens the leaver's own auto-deleveraging rank and shields unrelated traders. The guard is optional to begin with, and the same stale mark that trips the catch also blocks liquidation, so nothing closes during the window.

Recommendation

Fail closed on a guard revert and block the unlink until health is readable again.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-005

A malicious user can make fill-or-kill takers revert by resting an under-margined order the live fill then cancels

LOW ACKNOWLEDGED

ASSET

`src/orderbook/OrderBook.sol:594-604` `src/perps/PerpPool.sol:1254-1257`

Description and impact

A fill-or-kill order must fill completely or the whole trade reverts. Alice rests a bid that later no longer has enough margin. Bob sends a fill-or-kill that needs that size.

The book first dry-runs the fill without pulling under-margined rests, so Alice's bid still counts and Bob's order looks fully fillable. On the live fill the engine cancels Alice's bid for being under-margined. Bob no longer fills completely, so the trade reverts, and that revert puts Alice's bid back.

```
if (in_.orderType == OrderType.FillOrKill && newOrder.quantityRemaining != 0) {
    revert FillOrKillNotFillable();
}
```

Liquidations and stop-losses use immediate-or-cancel, not fill-or-kill, so they walk to the next level instead. Bob spends gas. Any other order type clears Alice's rest.

Recommendation

Skip under-margined rests in the dry run too, so a fill-or-kill that cannot actually complete never starts. After a live cancel, judge fill-or-kill against what filled, not against the earlier dry-run size.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-006

Revoking an operator on one market can look successful while a global approval still lets them trade there

LOW FIXED

ASSET

`src/registry/OperatorPermissionsRegistry.sol:109-128`
`src/registry/OperatorPermissionsRegistry.sol:158-173`

Description and impact

Alice approved an operator globally and now wants them off one market, so she calls `setOperatorApprovalForPool(pool, operator, selectors, false)`. That writes `false` into a per-pool slot that was never `true` and emits a revocation event. The authorization check falls straight past it into the global grant she never touched.

`src/registry/OperatorPermissionsRegistry.sol:160-164`

```
if ($.perPoolDenial[pool][owner][operator][selector]) return false;
if ($.perPoolApproval[pool][owner][operator][selector]) return true;
if ($.globalApproval[owner][operator][selector]) {
```

The operator keeps trading against Alice's cross-margin collateral on the market she thinks she closed. The event tells every front end and indexer watching that the revocation landed. The call that actually works is `setOperatorDenialForPool`, since denial is the one branch that outranks a global grant.

Recommendation

Make a per-pool `approved = false` also set the per-pool denial, mirroring the mutual clear that already runs the other way, or revert when there is no per-pool approval to clear so the caller learns nothing happened.

TEAM RESPONSE

Fixed in PR 14 (`80118ee`).

L-007

A trader cannot arm a stop-loss when another market's feed lapses because create still checks initial margin

LOW FIXED

ASSET

`src/perps/PerpStopOrderRegistry.sol:807-816`

Description and impact

A stop-loss never adds size. Create still runs a whole-account initial-margin check:

```
quoteMeetsIMForOrder(..., additionalSize: 0, ...)
```

That call prices every market Alice holds. A lapsed feed on any one of them reverts it, so she cannot arm a stop-loss on a market whose mark is fine.

Recommendation

Skip the initial-margin check when `additionalSize == 0`. A stop-loss cannot raise that requirement, and the check is what pulls in the other market's feed.

TEAM RESPONSE

Fixed in PR 14 (`149bdb1`).

L-008

A caller can skip one side of a stop-loss sweep by picking the block and the gas, and the call still reports success

LOW ACKNOWLEDGED

ASSET

`src/perps/PerpStopOrderRegistry.sol:586-591` `src/perps/PerpStopOrderRegistry.sol:239-246`

Description and impact

A new mark price walks stop-losses in two passes, one for each trigger direction. Which pass goes first follows whether the block number is even or odd. If the first pass spends the gas budget, the second pass is skipped and the call still returns success.

```
bool gteFirst = block.number % 2 == 0;
_processStopOrdersForOperator(newPrice, gteFirst ? Operator.GTE : Operator.LTE, gasFloor);
if (gasleft() < gasFloor) return;
```

`poke()` is permissionless, so a caller who picks the block and the remaining gas can fire only one side, or neither. The skipped stop-loss stays armed. The next call from anyone fires it.

Recommendation

Alternate sides with a counter that ticks each call, not the block number. If gas runs out before the second side, emit or revert so the skip is visible.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-009

Anyone can fill a market stop at the worst price in its published band by clearing the book and calling poke

LOW ACKNOWLEDGED

ASSET

`src/perps/PerpStopOrderRegistry.sol:239-247` `src/perps/PerpStopOrderRegistry.sol:663-668`

Description and impact

Anyone can empty opposite depth, rest a quote at the worst price in the published slippage band, and fire a triggerable market stop in the same transaction:

```
price = StopOrderPricing.marketLimitPrice(
    storedOrder.orderWithTrigger.order.isBid, price, params.tickSize, $.slippageToleranceBps
);
```

The fill stays inside the published slippage band the market stop already consents to. The attacker chooses the empty book and the trigger instant, but not a price outside Alice's accepted bound.

Recommendation

Require resting opposite depth before placing a triggered market stop, or let traders set a per-stop tolerance. Holding `poke()` to the scheduled cadence also removes the free choice of instant.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-010

An exhausted Stage 3 budget sends later liquidations to bidders instead of waiting for the next block

LOW FIXED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:945-954`
`src/perps/liquidation/LiquidationEngine.sol:1218-1223`

Description and impact

Every Stage 3 book fill in a block draws from one protocol-wide notional cap. When that cap is gone, Stage 3 returns exhausted and the waterfall continues into Stage 4:

```
if (currentBlockVolume >= volumeCap) {
    result.capExhausted = true;
    return result;
}
```

The second liquidatable account in a busy block can skip a live book and take a discounted bidder takeover, even ending below zero when the book path would have left it solvent.

Recommendation

If the intended behavior is deferral rather than escalation, skip Stages 4 through 6 when Stage 3 never ran because the cap was already exhausted. Budgeting per liquidated account also removes the ordering dependency.

TEAM RESPONSE

Fixed in PR 14 (`4c4843b`).

L-011

One trader can hold a market's whole open-interest cap with a wash pair

LOW ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:797-801`

Description and impact

The open-interest cap is market-wide with no per-account share. Both sides of a fill can belong to one person, so a wash long and short can sit on the whole cap and make every other opening taker revert:

```
require(
    $.totalOpenInterest <= $.perpPoolParameters.maxOpenInterest || $.totalOpenInterest <= oldOI,
    OpenInterestCapExceeded()
);
```

Liquidations and closes still go through, and margin comes back on unwind.

Recommendation

Add a per-account (or per linked-wallet group) open-interest share, or stop counting self-matched notional toward the market total.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-012

A trader in isolated mode can arm a stop on a second market while flat and lose the prepaid SOMI when it fires

LOW NOT REACHABLE

ASSET

`src/perps/MarginBank.sol:485-491` `src/perps/MarginBank.sol:990-997`
`src/perps/PerpStopOrderRegistry.sol:810-811`

Description and impact

Isolated mode is supposed to keep Alice in one market, including pending stops, but the check only walks `activePerpPools` and a stop never joins that list. She isolates while flat, prepays SOMI for an ETH stop, then opens BTC. When the ETH mark hits, the trigger burns that SOMI and does not place an exit.

```
if (!$.accountIsolated[account]) return true;
address[] storage pools = $.accounts[account].activePerpPools;
for (uint256 i; i < pools.length; ++i) {
    if (pools[i] != market) return false;
}
return true;
```

`setIsolated(true)` counts the same list, so a stop already sitting on a second market does not block enable either.

Recommendation

Notify the margin bank when a pending stop is created or cancelled so isolation uses the same footprint list as positions and resting orders. Reject `setIsolated(true)` while any armed stop sits outside that single market, and credit the prepaid SOMI back if a trigger drops because the account has already moved to another market.

TEAM RESPONSE

Shown not reachable in PR 14 (`59f5540`).

L-013

A triggered stop-loss can consume prepaid SOMI and report success after filling nothing

LOW FIXED

ASSET

`src/orderbook/OrderBook.sol:609-613` `src/perps/PerpStopOrderRegistry.sol:626-631`

Description and impact

Alice arms a stop-loss and prepays SOMI for the trigger. When the mark hits, the registry places it as an immediate-or-cancel order. If the book has no liquidity, that placement still returns success.

```
if (newOrder.quantityRemaining == 0 || in_.orderType == OrderType.ImmediateOrCancel) {
    _cleanupRemovedOrder(orderIndex, newOrder);
    return (true, newOrder.orderId);
}
```

The registry treats that boolean as a completed trigger and burns her prepaid SOMI. Alice's protection never executed. The event looks the same as a stop that filled.

Recommendation

Return whether anything actually filled. Refund the prepaid SOMI when a trigger places nothing.

TEAM RESPONSE

Reporting fixed in PR 14 (`2b20e4a`), and the cause fixed in PR 19 (`8a8adf2`).

L-014

A trader can block the owner from delisting a market by parking one minimum-size order

LOW ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:529-533`

Description and impact

The owner cannot delist a market while anyone still has a footprint in it. A resting order counts as a footprint.

```
require($.activeAccountCountPerPool[perpPool] == 0, PerpPoolHasActivePositions());
```

One ask at the minimum order size, parked far from the market, is enough. Only that trader can cancel it. Putting the market in close-only stops new orders but leaves the resting one in place.

Recommendation

When delisting, cancel leftover resting orders and refund their locked collateral in the same transaction, so the gate only waits on open positions.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

L-015

A Stage 4 takeover can push a bidder past max position size so reducing fills that stay over the cap revert

LOW FIXED

ASSET

`src/perps/PerpPool.sol:1367-1370`

Description and impact

Stage 4 is allowed to skip max position size so a takeover does not revert. The book still checks the cap on every fill and has no reduce escape, so after Carol is long 13 against a cap of 10, selling 1 reverts even though she is shrinking.

```
uint256 newAbsSize = absInt256(int256(oldSize) + int256(sizeDelta));
require(newAbsSize <= $.perpPoolParameters.maxPositionSize, MaxPositionSizeExceeded());
```

A single fill of 3 or more, including a full close of 13, still works.

Recommendation

Admit a fill when the new size is under the cap or smaller than before. Open interest already uses that pattern.

TEAM RESPONSE

Fixed in PR 13 (`288f22c`).

Owner writes and rotations

Owner setters that apply immediately, rotate an address, or leave live orders on the old rate.

I-001

Rotating the insurance fund silently uncovers every market and leaves the funded capital at the old address

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:1515-1546`

Description and impact

If the owner points the bank at a new insurance fund, every market loses coverage in that transaction and the funded capital stays at the old address. Coverage and capital can be restored with a later `setPoolTier` and a re-fund.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-002

The unbounded deleveraging scan setting is permissionlessly brickable, and the comment defending it assumes a number nothing enforces

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/liquidation/HyperliquidAdlRanker.sol:65-68`
`src/perps/liquidation/LiquidationEngine.sol:1554`

Description and impact

If the owner sets the deleveraging scan bound to zero, anyone can pad a market with dust holders until a liquidation no longer fits in a block. The shipped setting is 50, and at that value the flood does not brick the engine.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-003

An owner who points the fee recipient at a live builder bricks fills against that builder's resting orders

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:398-406` `src/perps/MarginBank.sol:589-603`

Description and impact

If the owner points the fee recipient at a live builder, fills against that builder’s resting orders revert until the makers cancel or the recipient is rotated again. The builder-fee cap is 10 bps on every market at launch, so this is reachable.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-004

An owner can settle elapsed funding at a new feed by rotating the aggregator while positions are open

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:278-286` `src/oracle/AggregatorOracle.sol:61-72`

Description and impact

If the owner rotates the aggregator's primary feed while positions are open, elapsed funding settles at the new feed's index. A pool-level oracle swap settles funding first. This path does not.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-005

An owner who rotates the credit granter can leave expired voucher floors stranded with no one able to reclaim them

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:623-636` `src/perps/voucher/VoucherDistributor.sol:131-149`

Description and impact

If the owner rotates the credit granter, the old distributor still holds expiry records but can no longer reclaim, and the new granter has reclaim rights with no records for prior claims. Those floors stay outstanding until the old granter is put back.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-006

An owner who lowers the builder fee cap still lets resting orders charge the old rate

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:682-704` `src/perps/PerpPool.sol:1014-1021`

Description and impact

If the owner lowers the builder fee cap, new placements must respect it, but already-resting orders keep charging the rate snapshotted at placement. Alice consented to that rate when she placed.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-007

An owner can reprice live orders and positions in one write, with no bound on the change and no delay

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:227-235` `src/perps/PerpPool.sol:1383-1407`

Description and impact

If the owner writes new fees and margin rates, they apply immediately to resting orders and open positions, with no bound on the change and no delay. A trader who rested at one taker fee is filled at the rate live when the match happens.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-008

An owner can revive expired auto-deleveraging sessions so counterparties still close at a stale bankruptcy price

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1381-1387`

Description and impact

If the owner widens the live auto-deleveraging time or block limit, a session that had already expired comes back. Counterparties still close at the bankruptcy price frozen when that session started.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-009

An owner who lowers the builder fee cap can kill armed stops and burn the trader's prepaid SOMI

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:678-682` `src/perps/PerpStopOrderRegistry.sol:626-627`

Description and impact

If the owner lowers the builder fee cap, an armed stop that carried a builder code can only revert at trigger, and the registry still burns the trader’s prepaid SOMI. Nothing tells the trader the stop was already dead.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-010

The owner cannot switch a pool off a broken oracle because the switch first settles funding on that same oracle

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:278-286` `src/perps/PerpPool.sol:592-600`

Description and impact

If the owner tries to switch a pool off a broken oracle, the switch first settles funding on that same oracle, so a reverting or stale feed takes the rotation down with it. Changing funding parameters hits the same settle.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-011

A trader loses their stored limit stop and the SOMI they prepaid for it when the owner changes the market's tick size

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpStopOrderRegistry.sol:663` `src/perps/PerpStopOrderRegistry.sol:777-779` `src/orderbook/OrderBook.sol:69-74`

Description and impact

If the owner widens the market’s tick size, every stored limit stop priced on the old grid reverts when it fires, the trigger still erases the order, and the prepaid SOMI is burned. Market stops survive because their limit is recomputed against the live tick.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

Market listing and delisting

Listing, delisting, and the leftover wiring those operations leave behind.

I-012

A delisted market still pins the insurance tier cap, so the owner cannot shrink the tier structure until they clear a market nobody can trade

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:529-538` `src/perps/InsuranceFund.sol:331-337`

Description and impact

Delisting a market suspends its insurance coverage but does not decrement the per-tier pool count, so the owner cannot shrink the tier cap until they clear a market nobody can trade. The count stays until that pool’s tier is later set to zero.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-013

A market can be listed and traded before the engine can deleverage anyone on it, so positions opened in that window cannot be liquidated

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:508-526` `src/perps/liquidation/LiquidationEngine.sol:1455-1463`

Description and impact

A market can be listed and traded before the engine has a deleveraging ranker, and positions opened in that window cannot be liquidated until the owner wires one. A market also trades at insurance tier 0 until the owner assigns a tier.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-014

An owner who removes and re-adds a market silently restores voucher trading they had tried to cut off

INFORMATIONAL FIXED

ASSET

`src/perps/MarginBank.sol:529-537` `src/perps/MarginBank.sol:649-654`

Description and impact

If the owner removes a market to cut off voucher trading and later re-adds it, the old allowlist entry comes back with the market. The allowlist cannot be cleared while the market is unregistered.

TEAM RESPONSE

Fixed in PR 14 (`fff553a`).

I-015

A trader whose opening order was already resting keeps opening new exposure after the owner puts the market in close-only mode

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:713-720` `src/perps/PerpPool.sol:393-394`

Description and impact

If the owner puts a market in close-only mode, orders already resting never face that check again. A bid that was on the book before the switch can still fill afterwards and open a fresh position.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

Rounding and dust

Truncation and one-wei accounting gaps. One numerical pass covers the family.

I-016

Health marks a position with a division that truncates toward zero while closing it floors toward negative infinity

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpMath.sol:45-46` `src/perps/MarginBank.sol:1603-1621`

Description and impact

Health values a position with division that truncates toward zero, while closing it floors toward negative infinity, so an open position can look one wei healthier than settlement would charge. The gap is at most one collateral wei per position and does not accumulate.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-017

Every fill floors the notional, then the fee, then the insurance fund's share of it, so the protocol under-collects wei dust

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:938-992` `src/common/Common.sol:205-207`

Description and impact

Every fill floors the notional, then the fee, then the insurance fund’s share, so the protocol under-collects less than one wei of fee per fill. Maker rebates floor too, which favours the protocol on the payout side.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-018

Partial fills release the collateral lock with a rounding-down division, leaving a few wei locked until the order is cancelled

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:909-915`

Description and impact

A partial fill releases a proportional slice of the rounded-up lock with a rounding-down division, so a few wei can stay locked until the order is cancelled. The dust sits as extra margin against the user, not as a shortfall against the book.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-019

The deleveraging claw-back buffer is a price offset that vanishes on small positions, so a multi-counterparty close can leave the bankrupt a few wei negative

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpMath.sol:264-271` `src/perps/PerpMath.sol:290-293`

Description and impact

The deleveraging claw-back buffer is one price-wei of headroom, and that headroom vanishes on small positions, so a multi-counterparty close can leave the bankrupt a few wei negative. The code already treats this as accepted dust in the bank's favour.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-020

Averaging a position's entry price rounds down, biasing average entry by up to one price wei per increase

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpMath.sol:156-161`

Description and impact

Growing a position stores an average entry up to one price wei below the exact average because the average is a truncating division. Successive increases do not stack that error, and closing still floors the credit.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-021

The stored collateral scale factor is never read, advertising a decimals conversion the bank does not perform

INFORMATIONAL FIXED

ASSET

`src/perps/MarginBank.sol:42` `src/perps/MarginBank.sol:181` `src/perps/MarginBank.sol:1814-1815`

Description and impact

The bank stores a collateral scale factor at initialization and never reads it. Collateral decimals already cancel out of every formula, so the field converts nothing.

TEAM RESPONSE

Fixed in PR 14 (`41ee092`).

I-022

The "no sub-wei rounding slack" claim on market health is false by one wei, because the notional it ceils was already floored

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:1699-1716` `src/perps/PerpMath.sol:54-56`

Description and impact

The comment on market health claims there is no sub-wei rounding slack, but the notional it ceils was already floored, so the requirement can sit one wei below the true figure. The error does not scale with position size.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-023

The oracle can keep serving a price the owner meant to reject because the two-feed gap check rounds down

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/oracle/AggregatorOracle.sol:173-177`

Description and impact

The two-feed gap check rounds down, so a disagreement a fraction of a basis point above the threshold can still read as agreement and the oracle keeps serving that price. The slack is well under one basis point.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

Stops, operators, and the book

Stop-loss registry, operator approvals, and order-book reporting.

I-024

One global operator approval covers both spot markets and every perp market the factory has ever minted

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/registry/OperatorPermissionsRegistry.sol:152-174`
`src/registry/OperatorPermissionsRegistry.sol:88-94`

Description and impact

One global operator approval covers both spot markets and every perp market the factory has ever minted. Wiring the factory turns the perp half on for existing approvals with no re-prompt, and any loss still needs that already-trusted operator to defect.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-025

The owner cannot upgrade the stop-loss registry in place, so a stuck callback chain needs a full redeploy

INFORMATIONAL WITHDRAWN

ASSET

`src/perps/PerpStopOrderRegistry.sol:39-46`

Description and impact

The stop-loss registry sits behind a proxy but has no upgrade hook, so even the owner cannot patch it in place. Recovering a stuck callback chain means redeploying the registry and migrating every pending stop.

TEAM RESPONSE

Disputed in PR 14 (`603bb9d`).

Design notes

Documented trade-offs and insurance accounting the code already treats as intended.

I-026

The flash loan exposure here is live equity reads and live book reads, not reentrancy through deposit and withdraw

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/MarginBank.sol:198-235` `src/perps/liquidation/AdlScoreLib.sol:37-62` `src/perps/PerpPool.sol:1140-1158`

Description and impact

A same-transaction deposit can still pad live equity and book reads even though every bank entry point is `nonReentrant`. That capital only reaches the deleveraging leverage score and the funding catch-up premium.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-027

Batch funding catch-up applies one live midpoint across every skipped interval

INFORMATIONAL FIXED

ASSET

`src/perps/PerpPool.sol:604-619` `src/perps/PerpPool.sol:1140-1158`

Description and impact

When funding catch-up runs after several intervals, it bills every skipped interval at one live book midpoint. That is the documented batch catch-up trade-off, and the reactivity handler is meant to keep catch-up at one interval.

TEAM RESPONSE

Fixed in PR 14 (`323bdd2`).

I-028

A trader decides which market's insurance tier absorbs their bankruptcy by choosing the order they opened their positions in

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/liquidation/LiquidationEngine.sol:1010-1016` `src/perps/MarginBank.sol:1663-1673`

Description and impact

Stage 4 walks a trader's markets in the order they opened them, so when a bidder can take only one position the insurance tier that pays depends on which market came first. The trader's own balance is unchanged either way.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

I-029

A maker can fill a resting order after the live initial margin rate has risen, using only the lock they posted at placement

INFORMATIONAL ACKNOWLEDGED

ASSET

`src/perps/PerpPool.sol:1258-1272` `src/perps/PerpPool.sol:1342-1352`

Description and impact

A maker locks collateral once at placement, and a later fill of the same increasing size still treats that old lock as enough even after the live initial margin rate has risen. Fills are not meant to re-check maker margin. Liquidation is meant to catch him later.

TEAM RESPONSE

Acknowledged in PR 14 (`603bb9d`).

Medusa Invariant Testing

Residual Foundry invariant campaigns were run as a supplemental check on the perps ledger. The campaigns did not produce additional reportable findings beyond those filed from manual review. The table below records the invariant groups and local validation runs documented for this audit.

CAMPAIGN	TARGET	INVARIANT GROUPS	RUNS	RESULT
Residual ledger invariants	`FizzResidual`	SI-1 realized backing, INV-TIER conservation, BT-9 canReceive, PI-6 organic max position, AD-2 ADL zero-sum, LW-9 no caller credit	2048 × 64 depth (~131k calls)	Passed; 6/6 invariants
Residual smoke (unit)	`FizzResidual`	Tier-0 absorb reverts, AC unprivileged smoke	Unit	Passed

In Scope

The residual campaigns focused on MarginBank realized backing (SI-1), Insurance Fund tier conservation, Stage 4 receive-capacity (BT-9), organic max position size (PI-6), ADL zero-sum settlement (AD-2), and no-caller-credit liquidation safety (LW-9), plus unit smoke for tier-0 absorb and access-control paths.

Out of Scope

These campaigns did not claim coverage for Somnia reactivity precompiles, stop-registry scheduling, voucher campaign operations, linked-wallet registry workflows, weird-token collateral, or report-specific PoC paths. Those areas were reviewed with targeted manual analysis and dedicated Foundry PoCs where applicable.

Considerations

Invariant testing is evidence of checked properties over the harness model, not a proof that the protocol is free of defects. Harness assumptions, skipped integrations, and production parameter choices should be reviewed separately before relying on these results for deployment readiness.

Appendix

Vulnerability Classification

IMPACT / LIKELIHOOD	LOW	MEDIUM	HIGH
HIGH IMPACT	Medium	High	Critical
MEDIUM IMPACT	Low	Medium	High
LOW IMPACT	Informational	Low	Medium

Critical severity is reserved for practical full fund drain, permanent protocol-wide bricking, repeatable standing-allowance drains, or similarly severe outcomes under realistic assumptions.

This matrix is a guide, not an absolute formula. Some issues may be rated above or below the mechanical impact and likelihood intersection when exploitability, blast radius, affected trust boundary, or protocol-specific economic consequences justify an exception.

Methodology

- Manual review
- Foundry proof-of-concept validation
- Foundry invariant / residual fuzz campaigns
- Architecture and threat-model review

Disclaimer

This report reflects the security review performed on the audited commit and should not be interpreted as a guarantee that the codebase is free of all defects. Security properties can change after code, configuration, or dependency updates.

About

This report was prepared as a technical smart-contract security review with emphasis on executable evidence, source-level explanations, and practical remediation guidance.