

Zxo Protocol Security Audit

REMEDIATION REVIEW COMPLETE

DATE	COMPILER	SCOPE	DEX MODEL
26 September 2026	solc 0.8.36	22 Solidity files · 2,638 lines	Uniswap V3 and forks (Pancake, Liberty)
METHOD	REVIEWER		
Manual line review + numeric tick/price verification	Claude (Anthropic), AI-assisted		

Summary Scores Findings Function flows Launch sequence Farming logic Owner powers Core LP

01

Executive summary

OVERALL SCORE

8.2_{/10}

~~6.6~~ → 8.2 after remediation

Unweighted mean of ten categories. Conditional on successful compilation and the fork tests listed under Before mainnet.

Zxo is a single-contract liquidity farm. Users stake Uniswap V3 positions minted on a fixed tick range around parity, and earn a newly minted reward token (one ZxoToken per pool). Emissions follow a target APR on staked value. Each pool is anchored by a permanently locked, single-sided core LP that sells the reward token on the way up.

The original review found 2 High, 4 Medium, 7 Low and 10 informational issues. Both High issues and every Medium issue that could be fixed in code have been fixed. The two remaining Mediums are spot-price dependence and the core-stake weighting. The team has accepted both as design trade-offs and they are recorded in the accepted-risk register.

No path was found that lets anyone take user principal. Unstake has no price-range gate and no owner gate, so staked positions can always be withdrawn after the 1-day lock-up. The core LP has no withdrawal function.

14 Resolved	10 Accepted / intended	0 Open High / Medium	0 Open findings
-----------------------------------	--	--	---------------------------------------

Limits of this review

The review environment could not install dependencies. The team has since compiled Zxo with solc 0.8.36 and confirmed the explorer verification input reproduces the build bytecode, but no fuzzing, fork tests or live deployment were run as part of this review. This AI-assisted review does not replace an independent human audit. Contracts that will hold user savings should also get a professional re-audit of the diff before mainnet.

Category scores

Original audit	After remediation	Scale 0–10
Reentrancy and CEI Every mutating entry point is nonReentrant. Unstake now makes its external NFT transfer last, after all state is final.	9.5	9.8
Access control Finalize flag and set-once ZXO_CONTRACT close the deployer path. Posman stays arbitrary until finalization, and the treasury uses single-step Ownable.	7.0	8.5
Token supply controls burn(address) removed. Seeding is capped by finalization, and staking can't open until the token is finalized. v3StartAmnt remains uncapped but visible on-chain.	6.0	8.5

Reward accounting	~~6.0~~ 7.5
TVL now counts every stake (M-01). The core stake still takes an unclaimed share of emissions (M-03, accepted).	
V3 integration and LP handling	~~7.0~~ 9.0
Fork-agnostic slot0 read, launch-price check, pair-decimals validation. All tick constants verified.	
Price and oracle dependence	~~4.5~~ 5.0
The health gate and bonus still read spot price (M-02, accepted). Reads now fail safely instead of reverting.	
External token compatibility	~~6.0~~ 8.5
forceApprove supports USDT-style tokens. Fee-on-transfer and rebasing pair tokens are unsupported by policy.	
DoS, gas and liveness	~~8.5~~ 9.0
Unstake accepts a recipient. Fail-safe totals can't block exits. Loops are bounded.	
Deployment and configuration	~~6.5~~ 8.0
ZxoVerify checks the 100 fee tier. Decimals are validated. Chain selection is still a manual import switch.	
Code quality and docs	~~7.0~~ 8.0
NatSpec matches the code. Dead code reduced, with some redundant guards kept by choice.	

03

Findings and status

SEVERITY	TOTAL	RESOLVED	ACCEPTED / INTENDED	OPEN
HIGH	2	2	0	0
MEDIUM	4	2	2	0
LOW	9	4	5	0
INFO	9	6	3	0

- All
- Resolved
- Accepted
- Open

H-01	Owner could take pre-minted Zxo through an arbitrary position manager	HIGH	RESOLVED
H-02	slot0() decode reverted on PancakeSwap V3	HIGH	RESOLVED
M-01	TVL counted only stakes that qualified for a bonus	MEDIUM	RESOLVED
M-02	Health gate and bonus read the spot price	MEDIUM	ACCEPTED
M-03	Core-stake position dilutes user rewards and its fees can't be collected	MEDIUM	ACCEPTED
M-04	USDT-style approve reverted	MEDIUM	RESOLVED
L-01	Launch price not verified on pool initialization	LOW	RESOLVED
L-02	Pair decimals not validated	LOW	RESOLVED
L-03	Bonus ramp reaches its cap about 20 ticks above parity	LOW	INTENDED
L-04	No user-supplied deadline	LOW	ACCEPTED
L-05	No rescue path for dust or stray NFTs	LOW	ACCEPTED
L-06	ZxoVerify checked the wrong fee tier	LOW	RESOLVED
L-07	Treasury uses single-step ownership	LOW	ACCEPTED
L-08	Contract stakers without an NFT receiver could never unstake	LOW	RESOLVED

L-09	Referrer can be set to the Zxo contract	LOW	ACCEPTED
I-01	Counter overflow on stats	INFO	RESOLVED
I-02	Silent uint16 truncation in updatePoolConfig	INFO	RESOLVED
I-03	Redundant guards kept	INFO	ACCEPTED
I-04	unstake truncated a uint256 token id	INFO	RESOLVED
I-05	Dead owner burn on ZxoToken	INFO	RESOLVED
I-06	HOLDER_TOKEN decimals not verified	INFO	ACCEPTED
I-07	_unpackPoolConfig returned 0 for an index above 5	INFO	RESOLVED
I-08	Single enum-coded event	INFO	ACCEPTED
I-09	Build pipeline hardening	INFO	RESOLVED

04

Function flows

Each flow lists the order of operations in the current code. Tags mark each step as a check (reverts on failure), a state effect, an external interaction, or an event.

CHECK Reverts on failure EFFECT Writes state EXTERNAL Calls another contract EVENT Emits log

stake

```
stake(uint8 poolId, uint256 amount0Desired, uint256 amount1Desired, uint256 slippageBps, address referrer)
```

Caller: `^address`

- | | | |
|----|----------|---|
| 1 | CHECK | Pool exists and is launched (stakes > 0). User has fewer than 21 positions |
| 2 | CHECK | Amounts, slippage ≤ 50% and minimum stake validated |
| 3 | CHECK | Pool is healthy: spot tick strictly inside the stake range |
| 4 | EFFECT | Set referrer if not already set (address(0) → treasury) |
| 5 | EXTERNAL | safeTransferFrom both tokens in, then forceApprove the position manager |
| 6 | EXTERNAL | Position manager mints the NFT to Zxo on the fixed range with slippage minimums |
| 7 | EXTERNAL | Refund unused amounts above the dust threshold |
| 8 | EFFECT | _updatePool, then compute bonus and 18-decimal total (HOLDER balance read) |
| 9 | EFFECT | Record position, add liquidity and bonus to totalStaked, add total to totalTokens |
| 10 | EVENT | Bonus (if any) and Stake events |
| 11 | EFFECT | Auto-adjust rewardRate if the 1-hour cooldown has passed |

The NFT is always held by Zxo. The bonus is virtual liquidity added to reward weight only.

harvest

```
harvest(uint64 tokenId, uint8 poolId, Action action)
```

Caller: Position owner

- | | | |
|---|----------|--|
| 1 | EFFECT | _updatePool accrues rewards to now |
| 2 | CHECK | Valid pool and action. Pool is healthy |
| 3 | CHECK | Caller owns the position and liquidity > 0 |
| 4 | EXTERNAL | Claim: mint the referral share to the referrer and the rest to the user, then reset rewardDebt |
| 5 | EXTERNAL | Collect: position manager collects fees to Zxo, which forwards them to the user |
| 6 | EVENT | Claim and/or CollectFees events |
| 7 | EFFECT | Auto-adjust rewardRate if due |

Blocked while the pool is unhealthy. Rewards keep accruing and can be claimed once the price is back in range.

unstake

`unstake(uint64 tokenId, uint8 poolId, bool acceptForfeit, address to)`

Caller: Position owner

- 1 **EFFECT** `_updatePool` accrues rewards to now
- 2 **CHECK** Valid pool, caller owns the position, liquidity > 0, 1-day lock-up passed
- 3 **CHECK** to defaults to `msg.sender` and can't be `Zxo` itself
- 4 **CHECK** Unless `acceptForfeit`, pending must be $\leq$ `MIN_UNSTAKE` (`1e12`)
- 5 **EFFECT** Decrement counters. Fail-safe subtraction from `totalStaked` and `totalTokens`
- 6 **EFFECT** Delete the position record and remove the user reference
- 7 **EVENT** Unstake event (records the forfeited pending amount)
- 8 **EFFECT** Auto-adjust `rewardRate` if due
- 9 **EXTERNAL** `safeTransferFrom` the NFT to ``to`` (last step)

No health gate and no owner gate. A failed transfer reverts the whole call with no state change.

createPool

`createPool(name, symbol, packedCore, packedParams, max30dRate, v3StartAmnt, pairToken, forceLower, withoutPool)`

Caller: Owner

- 1 **CHECK** Pool count < 255. maxNormalAPR ≤ 320%. pairToken ≠ 0
- 2 **CHECK** Pair decimals match the parameter and are 6, 8 or 18
- 3 **EXTERNAL** Deploy ZxoToken (owner permanently = Zxo)
- 4 **EFFECT** Push PoolInfo (target APR 10%, referral 3%) and pack config with a 365-day bonus window
- 5 **EXTERNAL** Approve the position manager. Mint coreLP + coreStake to Zxo and v3StartAmnt to the deployer

withoutPool = true deploys the token only, with no mint.

initializePool

`initializePool(uint8 poolId, uint16 slippageBps, bool isCore)`

Caller: Owner

- 1 **CHECK** stakes == 0. The core LP must not exist yet (core path) or must already exist (stake path)
- 2 **CHECK** Stake path: `IZxoV3Deployer.isFinalized(rewardToken)`
- 3 **EXTERNAL** Core path: create and initialize the pool, then require slot0 price == launch price
- 4 **EXTERNAL** Mint the single-sided position to Zxo with the slippage minimum
- 5 **EFFECT** Core path: store corePositionId in poolConfig
- 6 **EFFECT** Stake path: record a Zxo-owned position, add to totals, stakes = 1 (opens staking)

Each path runs once per pool. Neither position can ever be withdrawn.

startV3pool / finalizeToken

`startV3pool(poolId, slippageBps, amount, posman) · finalizeToken(token)`

Caller: Deployer owner

- 1 **CHECK** startV3pool: posman ≠ 0, amount > 0, token not finalized, balance ≥ amount

2	EXTERNAL	Read token0, token1 and rewardToken from Zxo.poolInfo. Read pair decimals
3	EXTERNAL	forceApprove posman. Create and initialize the pool. Mint single-sided to 0xdEaD
4	EVENT	PoolStarted(poolId, posman, tokenId, liquidity, amount0, amount1)
5	CHECK	finalizeToken: token $\neq$ 0 and not already finalized
6	EFFECT	Set isFinalized[token] = true
7	EXTERNAL	Burn the deployer's whole balance of the token
8	EVENT	TokenFinalized(token, burned)

Once finalized, seeding is closed for that token permanently.

updatePoolConfig

updatePoolConfig(uint8 poolId, uint8 dataType, uint96 newValue)

Caller: Owner

1	CHECK	Valid pool. Values checked with _safe16
2	EFFECT	Type 0: targetAprBps, clamped to maxNormalAPR (2× during the bonus year), then auto-adjust
3	EFFECT	Type 1: referral rate, 3%–30%
4	EFFECT	Type 2: lower rewardRate only. _updatePool runs first. Lasts until the next auto-adjust

No control over ranges, tokens or user positions.

updatePool / collectFeesFromCoreLP

updatePool(uint8 poolId, bool updateAll) · collectFeesFromCoreLP(uint8 poolId)

Caller: Anyone · Owner

- | | | |
|---|----------|---|
| 1 | CHECK | updatePool: validate poolId (single-pool path) |
| 2 | EFFECT | Accrue acc and auto-adjust the rate for one pool or all pools |
| 3 | CHECK | collectFeesFromCoreLP: owner only, valid pool |
| 4 | EXTERNAL | Collect the core LP's fees to Zxo, then transfer them to ZXO_TREASURY |

Core LP liquidity is never touched. Only accrued fees move.

05

Pool launch sequence

The contracts enforce the order of steps 4 and 5. The core-stake mint reverts until the reward token is finalized on the deployer, and staking can't open until the core stake exists. Step 2 comes before any Zxo is in public hands, so no one can seed the main pool at a price of their choosing.

STEP 1

createPool

Deploys the ZxoToken. Checks pair decimals, mints coreLP + coreStake to Zxo and v3StartAmnt to the deployer.

STEP 2

initializePool · core

Creates and initializes the pool, requires price == launch price, and mints the single-sided core LP.

STEP 3

startV3pool ×n

Seeds external DEXes only. NFTs are sent to 0xdEaD. Never seed the main position manager here.

STEP 4

finalizeToken

Burns the deployer's remaining balance. Permanently blocks further seeding.

STEP 5

initializePool · stake

Requires isFinalized. Mints the core stake, which opens staking.

Seeding-window risk at production size

To push the tick from the launch tick (10) past the core-stake lower bound (31), buyers must absorb about 7% of the core LP. At the planned core sizes that is roughly \$350k for a \$5M core, \$1.05M for \$15M and \$70M for \$1B. Seeded DEX positions are too small to move the main pool through arbitrage. Before step 5, confirm that `queryData(0, poolId, 4, 0)` returns a tick below 31 (or above -31 when Zxo is token1).

06

Farming logic (MasterChef on V3 liquidity)

Accounting model

```
acc += elapsed × rewardRate × 1e18 / totalStaked
pending = liquidity × acc / 1e18 - rewardDebt
rewardDebt = liquidity × acc / 1e18    (on stake, claim)
```

Rewards are minted on claim, not pre-funded. Forfeited or unclaimed rewards are never minted. Every user position shares one fixed tick range, so liquidity is proportional to position value at any price, and liquidity weighting is fair between users.

Emission controller

```
rewardRate = min( TVL × targetApr,
                  TVL × maxApr,           // 2× maxNormalAPR in year 1
                  max30dRate / 30 days )
cooldown   = 1 hour after each change
```

TVL is `totalTokens`, the 18-decimal stake-time amounts summed at 1:1. Since M-01, every stake counts toward TVL. Lowering the rate by hand (`updatePoolConfig` type 2) only lasts until the next auto-

adjust; lowering `targetAprBps` is the lasting control.

Invariants verified

INVARIANT	WHY IT HOLDS
Pending never underflows	<code>rewardDebt</code> is always set from the current <code>acc</code> , and <code>acc</code> only increases.
No retroactive rate changes	<code>_updatePool</code> runs before every change to <code>totalStaked</code> and before every rate change.
Bonus is symmetric	Bonus liquidity is added to <code>pos.liquidity</code> and <code>totalStaked</code> together and removed together.
Referral is not additive	The referral share comes out of the user's pending amount. A self-referral wallet earns nothing extra.
Record $\Rightarrow$ custody	A record with <code>owner $\neq$ 0</code> and <code>liquidity > 0</code> exists only while Zxo holds the NFT. The one exit deletes the record before transferring.
Exit always available	Unstake has no health gate and no owner gate. The fail-safe subtractions stop accounting drift from blocking exits.
Pool cannot un-launch	The core stake keeps <code>stakes $\geq$ 1</code> permanently.

Stake range and health (18/18 pair)

ZX0 PRICE	TICK	STATE
> ~1.0305	> +300	UNHEALTHY
~0.9704 – ~1.0305	(-300, +300)	HEALTHY
< ~0.9704	< -300	UNHEALTHY

When the pool is unhealthy, stake and harvest revert, unstake still works, and rewards keep accruing. Bounds are strict. For 6- and 8-decimal pairs the ranges center on ± 276324 and ± 230270 .

Stake bonus curve

TICKS ABOVE PARITY	0	4	10	16	20+
Bonus	0%	4%	10%	16%	20%
With HOLDER	0%	5%	12.5%	20%	25%

The bonus reaches its cap at a 53.36% paired ratio. The core LP launches at tick 10, so the organic bonus is expected to be around 10–12%.

07

Owner powers

Zxo owner can

- Create up to 255 pools and tokens, setting coreLP, coreStake, maxNormalAPR ($\leq 320\%$), decimals (validated), max30dRate and v3StartAmnt
- Initialize the core LP and core stake once per pool
- Set targetAprBps from 0 to the cap (turns emissions on or off)
- Set the referral rate between 3% and 30%
- Lower rewardRate temporarily
- Send core-LP trading fees to the treasury
- Transfer ownership (two-step) or renounce it

Zxo owner cannot

- Touch user NFTs, principal or unclaimed rewards
- Block or delay unstake
- Withdraw or reduce core LP or core-stake liquidity
- Mint the reward token outside pool creation and emissions
- Burn anyone's tokens (burn(address) removed)
- Change tick ranges, fee tier, position manager, factory or reward token
- Pause, upgrade, or rescue any asset held by Zxo

CONTRACT · ROLE	CAN	CANNOT
ZxoToken owner = Zxo, permanent	Mint	Transfer or renounce ownership. Burn others' tokens. Holders can burn their own tokens.
ZxoV3Deployer Ownable2Step	Seed external DEXes with any position manager until the token is finalized. Finalize (burn the remainder). Set Z XO _ CONTRACT once.	Seed after finalization. Repoint Z XO _ CONTRACT. Withdraw tokens on mainnet (testnet-only validator path).
ZxoTreasury Ownable (single-step)	Sweep any token to the destination. Change the destination after a 7-day delay. Lock the destination forever.	Receive ETH. Bypass the delay.

08

Core LP ownership

The core LP NFT is minted to `address(this)` and its id is stored in `poolConfig`. Zxo has no function that transfers it or reduces its liquidity. **The core liquidity, and all pair tokens it collects as it sells Zxo between ~1.001 and ~1.03, is permanently locked.** Trading fees are the only value that leaves, and they go to the fixed treasury through an owner-only call. If Zxo ownership is renounced, those fees are locked too.

Before minting, the launch price is now checked against the pool's actual price (L-01). All six launch prices were checked with a port of Uniswap's TickMath. Each sits exactly one tick outside its core range, on the correct side for a single-sided Zxo deposit.

PAIR	Z XO IS	LAUNCH TICK	CORE LP RANGE	CORE STAKE RANGE	USER STAKE RANGE
18 / 18	token0	10	[11, 300]	[31, 300]	[-300, 300]
18 / 18	token1	-10	[-300, -11]	[-300, -31]	[-300, 300]
18 / 6	token0	-276314	[-276313, -276024]	[-276293, -276024]	[-276624, -276024]

PAIR	ZXO IS	LAUNCH TICK	CORE LP RANGE	CORE STAKE RANGE	USER STAKE RANGE
18 / 6	token1	276314	[276024, 276313]	[276024, 276293]	[276024, 276624]
18 / 8	token0	-230260	[-230259, -229970]	[-230239, -229970]	[-230570, -229970]
18 / 8	token1	230260	[229970, 230259]	[229970, 230239]	[229970, 230570]

09

Accepted-risk register

These items were reviewed with the development team and deliberately left as they are. Items the team excluded from scope before the audit are listed at the end.

ID	RISK	TEAM POSITION
M-02	Health gate and bonus read the spot price An atomic swap, then stake or harvest, then swap back can bypass the harvest gate or get the maximum bonus for about two 0.01% fees.	Accepted. Emissions are capped, principal is not at risk, and the large core LP makes sustained manipulation expensive. No TWAP will be added.
M-03	Core-stake position dilutes user rewards and its fees can't be collected The contract-owned core stake counts toward TVL and reward weight, with about 2.2× weight per token, but never claims. Users receive less than targetApr by roughly its share of totalStaked. Its trading fees are locked.	Accepted. Auto-adjust does not close this gap. The gap shrinks as user TVL grows, and targetAprBps can compensate.
L-03	Bonus ramp reaches its cap about 20 ticks above parity The constant 59538 hits the 20% cap at a 53.36% paired ratio, so the bonus is close to all-or-nothing.	Intended. At the launch tick the organic bonus is about 10–12%.
L-04	No user-supplied deadline block.timestamp + 20 minutes is evaluated at execution, so a delayed transaction still runs.	Accepted. Price protection comes from the slippage minimums.

ID	RISK	TEAM POSITION
L-05	No rescue path for dust or stray NFTs Refunds at or below the dust threshold, and NFTs sent directly to Zxo, can't be recovered.	Accepted by design. No rescue function exists, so the owner has no path to any asset Zxo holds.
L-07	Treasury uses single-step ownership A mistaken transferOwnership can't be undone.	Accepted. Owner operations are handled carefully.
L-09	Referrer can be set to the Zxo contract The referral share would be minted to Zxo and be unrecoverable. Only that user's own share is affected.	Accepted. Any address may be a referrer, and address(0) defaults to the treasury.
I-03	Redundant guards kept Some checks can't be reached or are no-ops: the queryData type-3 branch, userReferrer_[this], the rewardDebt write when pending == 0, and several library guards.	Kept by choice as defense in depth. Unused enums, ownerOf checks and zero-token checks were removed.
I-06	HOLDER_TOKEN decimals not verified HOLDER_THRESHOLD assumes 18 decimals.	Accepted. HOLDER is a standard 18-decimal token on all target chains.
I-08	Single enum-coded event Only user is indexed, so indexers must filter on the action value.	Accepted. Each separate event adds bytecode, and one shared event keeps Zxo under the 24,576-byte limit while still emitting every action.

Out of scope by design (declared before the audit)

BEHAVIOR	RATIONALE
Bare <code>revert()</code> without reasons	Saves gas and bytecode. Frontends pre-check conditions.
Unstaking forfeits pending rewards	The protocol pays out only while the tick range is healthy. Users harvest first.
HOLDER bonus can be bought, used and sold	The HOLDER token is only available on public DEXes. The team treats this as volume and participation.
Tick-range requirement on stake	Covered by the user-set slippage parameter.

Before mainnet

Build and configuration

- ☐ Switch the ChainLib import from `SepoliaTestLib.sol` to the target chain with `update-chain-lib.py`, and set `ZX0_TREASURY` and `ZX0_V3_DEPLOYER`
- ☐ Compile. Watch `createPool` and `initializePool` for "stack too deep"
- ☐ Confirm Zxo runtime bytecode is under 24,576 bytes (EIP-170)
- ☐ `compile.py` stops on standard-JSON errors (done)
- ☐ Verification input compiles with no local files and matches the build (confirmed for Zxo; repeat for the other contracts)
- ☐ EVM version selectable in `compile.py`. Use Shanghai: PulseChain supports nothing newer (done)
- ☐ Run `ZxoVerify` twice (without, then with, the deployer check)
- ☐ Run Slither or Aderyn and triage the results against this register
- ☐ Move Zxo and deployer ownership to a multisig

Tests

- ☐ Fork: read `slot0` on every target DEX. The old typed call reverts on PancakeSwap
- ☐ Fork: stake a USDT pair on Ethereum
- ☐ Fork: run the full launch order. The core stake reverts until `finalizeToken`
- ☐ Fork: `initializePool(core)` reverts on a pool pre-initialized at another price
- ☐ `createPool` reverts when decimals don't match the pair token
- ☐ Unstake to a contract without the NFT receiver reverts with no state change. A retry with an EOA as `to` succeeds
- ☐ Invariant: `totalStaked ==  $\sum$  pos.liquidity` (core stake included)
- ☐ Invariant: `totalTokens ==  $\sum$  pos.tokens` (core stake included)
- ☐ Invariant: the owner can always unstake after the 1-day lock-up
- ☐ Independent re-audit of the diff since Revision 1

Frontend and ABI changes

- `unstake(uint64 tokenId, uint8 poolId, bool acceptForfeit, address to)` .The selector changed. Pass `address(0)` to send the NFT to the caller.
- `queryData(..., 4)` returns `(false, 0, 0)` when a price can't be read, instead of reverting. Treat `sqrtPriceX96 == 0` as "not ready".
- `totalGlobalStakes()` and `referralsPerUser_()` now return `uint32` .
- New on `ZxoV3Deployer`: `isFinalized(address)` , `finalizeToken(address)` and the event `TokenFinalized(address indexed, uint256)` .
- Trust panel for each pool: `isFinalized` is true, the deployer balance is 0, every `PoolStarted` posman is canonical, and the seeded NFTs are owned by `0xdEaD`.

11

Scope and file fingerprints

Revision 2 covers the contracts as uploaded on 25 September 2026, plus the `ZxoLib.sol` update of 26 September 2026. Hashes are the first 12 hex characters of each file's SHA-256. Any later change to these files is outside this review.

Compiler: solc 0.8.36, legacy pipeline (viaIR off)

Etherscan lists three known bugs for 0.8.36, all fixed in 0.8.37. None applies to Zxo.

`MisorderedNamedParametersInRequireWithCustomErrors` and

`SpillSlotCollisionAcrossMutualRecursion` only affect the via-IR pipeline, and Zxo has no custom errors in `require` and no recursion. `MemoryByteArrayElementDeleteClearsWholeWord` needs `delete` on an element of a memory `bytes` array; the only deletes in the build clear storage mappings. Keep `viaIR` off.

FILE	LINES	SHA-256 (PREFIX)	ROLE
<code>Zxo.sol</code>	1017	86507f3ce7ba	Farm, staking, emissions, core LP
<code>libraries/ZxoLib.sol</code>	504	5c58a8c2c69b	Ticks, bonus, packing, validation

FILE	LINES	SHA-256 (PREFIX)	ROLE
ZxoV3Deployer.sol	331	3c8fb36af7dd	External DEX seeding and finalization
ZxoTreasury.sol	231	aa47be5c2210	Fee and referral receiver
ZxoVerify.sol	152	80ae67949b81	Deployment sanity check
tokens/ZxoToken.sol	103	87ba572691dd	Per-pool reward token
interfaces/Iv3PosManager.sol	56	0a8356439ff0	Position manager interface
interfaces/IZxo.sol	22	5fd91cc8d935	poolInfo reader
interfaces/IZxoToken.sol	12	856487dac631	mint, coreLP, coreStake
interfaces/IUniswapV3.sol	7	e5c8cd93d1d8	Factory interface
interfaces/IZxoV3Deployer.sol	5	d332e8f4212b	isFinalized
libraries/*Lib.sol (11 chains)	18 each	various	Per-chain addresses (all EIP-55 valid)

This report is an AI-assisted security review, based on reading the source and on numeric checks of the tick and price constants. It isn't a guarantee that the code is free of defects, and it isn't financial or legal advice. Behavior depends on the external DEX contracts, chain conditions and deployment configuration, none of which were tested live. An independent professional audit is recommended before contracts hold user funds.