

Zxo Protocol Security Audit

DEPLOYED ON ARBITRUM AND PULSECHAIN · SOURCE VERIFIED

DATE	COMPILER	SCOPE	DEX MODEL
5 October 2026	solc 0.8.36	24 Solidity files · 2,831 lines	Uniswap V3 and forks (PancakeSwap-style, LibertySwap)
METHOD	REVIEWER		
Manual line review, numeric tick/price checks, verified-source and on-chain event review	Claude (Anthropic), AI-assisted		

01 Executive summary

OVERALL SCORE

8.8/10

6.6 → 8.5 after remediation
→ 8.8 deployed

Unweighted mean of ten categories.
Raised in Revisions 3.0 and 3.1 on
verified evidence from the Arbitrum
and PulseChain deployments; see the
checklist for what is still open.

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. One Medium remains, the core-stake weighting (M-03), which the team accepted as a design trade-off. The spot-price stake bonus left over from M-02 is also accepted. Both 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 Unstake has no price-range gate and no owner gate, so staked positions can always be withdrawn after the 1-day lock-up.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.

20 Resolved	9 Accepted / intended	0 Open High / Medium	0 Open informational
----------------	-----------------------------	----------------------------	----------------------------

Limits of this review

Revisions 3.0 and 3.1 checked the Sourcify-verified source of the Arbitrum and PulseChain deployments against the reviewed files and read on-chain events and transactions through block explorer APIs. It made no direct contract state reads and ran no fuzzing, static analysis or fork tests. Statements marked as reported by the team were not independently checked. This AI-assisted review does not replace an independent human audit, which is still recommended for contracts that hold user savings.

Revision 3.1 • 5 October 2026 • PulseChain deployment review

The six contracts deployed on PulseChain mainnet (including both ZxoVerify runs) are exact-match verified on Sourcify and are the same audited code: ten files match by keccak256 hash and the rest line by line. PulseChain uses LibertySwap, a PancakeSwap-style V3 DEX, so this is the first live confirmation of the H-02 fix on that pool type. Nine pools launched in the enforced order across 100 transactions; a reward claim was reproduced to the last wei, the HOLDER bonus applied at exactly 1.25 times, and core LP fees were collected to the treasury. Score raised from 8.7 to 8.8. No new findings. I-14 (single-wallet owner) now also applies to PulseChain. Details are in the separate PulseChain Deployment Report.

Correction, same day. The first issue of this revision described the PulseChain core stakes as large and said stakers earn about half of the target APR while a core stake dominates. In value the core stakes are small, about \$330 to \$1,030 per pool, and the half-of-target case applies only to pools holding test-size stakes. M-03, the pool notes and the score rationale now say so. Scores are unchanged.

Revision 3.0 • 2 October 2026 • Arbitrum deployment review

The five contracts deployed on Arbitrum One are exact-match verified on Sourcify and are built from the Revision 2.5 source: seven files match by keccak256 hash and the rest line by line. The only changes are the license header, the ArbitrumLib import and the Arbitrum configuration (the HOLDER token and threshold changed). Six pools were launched in the enforced order, and stake, harvest, unstake and config changes all ran on mainnet; a live reward claim was reproduced to the last wei from the audited formula. Score raised from 8.5 to 8.7. One accepted item added (I-14: the owner is a single wallet until the planned multisig). The pre-mainnet checklist is replaced by a status table. Details are in the separate Arbitrum Deployment Statement.

Revision 2.5 • 27 September 2026

All open items are closed. SepoliaTestLib now has a complete Z XO _ORACLE line, so the tree builds (I-12). ConfigUpdated now logs the value actually stored, including the capped target APR, from one shared log call (I-13). ZxoVerify has full NatSpec for the deploy order and both constructor flags. No change to runtime logic, funds flow or scores.

Revision 2.4 • 27 September 2026

ZxoVerify now checks ZxoOracle on-chain, behind a second constructor flag (verifyOracle): code present, WINDOW() == 900, and a fail-safe answer (no average, never in range) for an address that is not a pool. PoolInitialized now indexes the reward token, matching PoolCreated; I-13 is narrowed to the ConfigUpdated value. No change to runtime logic, funds flow or scores.

Revision 2.3 • 27 September 2026

The Zxo constructor now validates the full ChainLib configuration with readable error messages, at no runtime bytecode cost. New events record deployment, pool creation, pool initialization, core fee collection and config updates, and updatePoolConfig rejects unknown data types. All chain libraries now declare Z XO _ORACLE. I-11 resolved. Two informational items added (I-12 build, I-13 event data).

Revision 2.2 • 27 September 2026

Adds ZxoOracle: the health gate now also requires a 15-minute TWAP inside the stake range, which closes the same-transaction harvest bypass (M-02 resolved; the spot-based bonus remains accepted). Events now go through one shared function, redundant library checks and the write-only tokensIds _ mapping were removed, uint64 casts use a checked downcast, and the Zxo constructor validates the HOLDER settings. Revision 2.1 moved HOLDER_THRESHOLD into each chain library and added HOLDER checks to ZxoVerify. One informational item added (I-11, oracle setup).

02

Category scores

Original audit Current (Revision 3.1) Scale 0–10

Reentrancy and CEI



9.5 9.8

Every mutating entry point is nonReentrant. Unstake now makes its external NFT transfer last, after all state is final. Unchanged in Revision 3.0.

Access control		~~7.0~~ 8.5
Finalize flag and set-once Z XO_CONTRACT close the deployer path; both were used correctly on Arbitrum and PulseChain. Unchanged at 8.5 because the live owner on each chain is a single wallet until the planned multisig (I-14).		
Token supply controls		~~6.0~~ 8.7
burn(address) removed. Seeding is capped by finalization, and staking can't open until the token is finalized. All 15 reward tokens on Arbitrum and PulseChain were finalized before staking opened. v3StartAmnt remains uncapped but visible on-chain.		
Reward accounting		~~6.0~~ 8.2
TVL counts every stake (M-01). Live claims on Arbitrum and PulseChain matched the audited formula to the last wei, the staked value used for emissions matched the core-stake amount on both, and the HOLDER bonus applied at exactly 1.25 times. The core stake still takes an unclaimed share of emissions (M-03, accepted). Core stakes are small and fixed at creation, about \$330 to \$1,030 per pool on PulseChain, so stakers earn about 90% of target once user stakes are ten times the core stake and about 99% at one hundred times.		
V3 integration and LP handling		~~7.0~~ 9.6
Fork-agnostic slot0 read, launch-price check, pair-decimals validation. All tick constants verified. Live on Uniswap V3 (Arbitrum, six pools) and on LibertySwap, a PancakeSwap-style V3 DEX (PulseChain, nine pools), which confirms the H-02 fix on that pool type.		
Price and oracle dependence		~~4.5~~ 7.7
The health gate requires the spot price and a 15-minute TWAP to be in range (M-02). It worked on live Uniswap V3 pools on Arbitrum and LibertySwap pools on PulseChain. The bonus still uses the spot token mix.		
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~~ 9.2
Unstake accepts a recipient. Fail-safe totals can't block exits. Loops are bounded. Nine mainnet unstakes on two chains succeeded after the 1-day lock-up.		
Deployment and configuration		~~6.5~~ 9.7
The Zxo constructor validates every ChainLib address, the DEX wiring, fee tier and tick spacing, the HOLDER token, and the treasury, deployer and oracle contracts. Those checks passed on Arbitrum and PulseChain, the deployed source matches the audited source on both, and every deployed contract is exact-match verified. ZxoVerify ran twice on PulseChain.		
Code quality and docs		~~7.0~~ 8.5
NatSpec matches the code. Dead code removed, out-of-range queries revert, downcasts are checked, and deployment settings are validated in the constructor. Unchanged: no static-analysis report has been provided.		

Findings and status

SEVERITY	TOTAL	RESOLVED	ACCEPTED / INTENDED	OPEN
HIGH	2	2	0	0
MEDIUM	4	3	1	0
LOW	9	4	5	0
INFO	14	11	3	0

H-01	Owner could take pre-minted Zxo through an arbitrary position manager	HIGH	RESOLVED
LOCATION	<code>ZxoV3Deployer.startV3pool, Zxo.createPool</code>		
ISSUE	startV3pool approved any posman address for the seeding amount. A malicious posman could transferFrom the tokens during mint(). setZxoContract could also be repointed to a fake pool source.		
RESOLUTION	Added isFinalized and finalizeToken, which burn the remainder and permanently block seeding for that token. setZxoContract can be set only once. initializePool(core stake) requires isFinalized, so staking can't open before seeding is closed.		
RESIDUAL	Low. Posman is still arbitrary during the seeding window, before any user funds arrive. Users should confirm that every PoolStarted posman is canonical before staking.		

H-02	slot0() decode reverted on PancakeSwap V3	HIGH	RESOLVED
LOCATION	<code>Zxo._poolRatio</code>		
ISSUE	The typed IUniswapV3Pool call decoded feeProtocol as uint8. Pancake stores a packed uint32 there, so decoding always reverted, which would have blocked stake and harvest on BNB.		
RESOLUTION	Replaced with _readSlot0: a low-level staticcall that decodes only sqrtPriceX96 and tick, the fields every V3 fork shares. The Liberty branch and pool imports were removed. A failed read returns unhealthy. Revision 3.1: confirmed live on LibertySwap, a PancakeSwap-style V3 DEX on PulseChain, where nine pool launches, 16 stakes and 6 harvests read the price successfully.		
RESIDUAL	None for PancakeSwap-style pools. PancakeSwap on BNB Chain itself has not been run but uses the same pool design.		

M-01 TVL counted only stakes that qualified for a bonus		MEDIUM	RESOLVED
LOCATION	<code>ZxoLib._calcBonusAndTotal</code>		
ISSUE	Early returns sent back <code>totalAmount = 0</code> for deposits at or below parity. Those stakes added reward weight but no TVL, which lowered emissions and inflated the displayed APR.		
RESOLUTION	<code>totalAmount</code> is now computed before every early return.		
RESIDUAL	Emissions for the same <code>targetApr</code> are higher than before. Recheck the caps per pool.		
M-02 Health gate and bonus read the spot price		MEDIUM	RESOLVED
LOCATION	<code>Zxo._poolRatio</code> , <code>ZxoOracle</code> , <code>ZxoLib._calcBonusAndTotal</code>		
ISSUE	An atomic swap, then stake or harvest, then swap back could bypass the harvest gate or get the maximum bonus for about two 0.01% fees.		
RESOLUTION	Revision 2.2: the health gate now requires both the spot tick and a 15-minute time-weighted average tick (<code>ZxoOracle.inRange</code> , read through <code>observe()</code>) to be inside the stake range. A same-transaction swap cannot move the average, so the harvest-gate bypass is closed. The oracle is only called when the spot check passes.		
RESIDUAL	The stake bonus is still computed from the deposit's token mix at the spot price, so a same-transaction swap can still raise one stake's bonus (capped at 20%, 25% with HOLDER). Accepted: emissions stay capped and principal is not at risk. Optional next step: derive the bonus from the TWAP tick.		
M-03 Core-stake position dilutes user rewards and its fees can't be collected		MEDIUM	ACCEPTED
LOCATION	<code>Zxo.initializePool</code> (core stake)		
ISSUE	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 <code>targetApr</code> by roughly its share of <code>totalStaked</code> . Its trading fees are locked.		
RESOLUTION	Accepted. Auto-adjust does not close this gap. The gap shrinks as user TVL grows, and <code>targetAprBps</code> can compensate. Revision 3.0: the team keeps core stakes very small. On Arbitrum pool 0 the core stake is about 0.19 tokens beside a 361,486-token core LP, so the dilution is negligible once users stake more than a few tokens. Revision 3.1: on PulseChain the core stakes are larger in token terms (pool 0 is 110,385,144 tokens) but small in value: about \$330 to \$1,030 per pool and about \$4,900 across nine pools at DEX prices on 5 October 2026 (the team reports \$1,078.57 for pool 0 and about \$200 to \$600 for the others). Each amount is immutable in the reward token and minted once. Stakers earn about 62% of target when user stakes equal the core stake, about 90% at ten times and about 99% at one hundred times. On 5 October stakers in pools 6 and 8 were at about 90% and 99.9%; the other pools held only test-size stakes, where stakers earn about half of target.		
RESIDUAL	Steady-state APR gap. Minor locked fees.		

M-04	USDT-style approve reverted	MEDIUM	RESOLVED
LOCATION	Zxo.stake, ZxoV3Deployer.startV3pool		
ISSUE	A raw IERC20.approve expects a bool return value. USDT returns none, so staking a USDT pair was impossible.		
RESOLUTION	Switched to SafeERC20.forceApprove after the transfer in, in both contracts.		
RESIDUAL	None.		
L-01	Launch price not verified on pool initialization	LOW	RESOLVED
LOCATION	Zxo.initializePool		
ISSUE	Anyone can pre-create the pool for the predictable token address at another price, which would put the core LP at the wrong price or block it.		
RESOLUTION	After createAndInitializePoolIfNecessary, the pool's slot0 price must equal the launch sqrtPrice, or initialization reverts.		
RESIDUAL	Recovery: a swap with sqrtPriceLimitX96 = launch price.		
L-02	Pair decimals not validated	LOW	RESOLVED
LOCATION	Zxo.createPool		
ISSUE	The decimals parameter drives every tick and normalization calculation but wasn't checked against the pair token.		
RESOLUTION	createPool reads IERC20Metadata.decimals() and requires a match with the parameter and a value of 6, 8 or 18.		
RESIDUAL	None.		
L-03	Bonus ramp reaches its cap about 20 ticks above parity	LOW	INTENDED
LOCATION	ZxoLib._calcBonusAndTotal		
ISSUE	The constant 59538 hits the 20% cap at a 53.36% paired ratio, so the bonus is close to all-or-nothing.		
RESOLUTION	Intended. At the launch tick the organic bonus is about 10-12%.		
RESIDUAL	Documented.		
L-04	No user-supplied deadline	LOW	ACCEPTED
LOCATION	Zxo.stake, initializePool, ZxoV3Deployer.startV3pool		
ISSUE	block.timestamp + 20 minutes is evaluated at execution, so a delayed transaction still runs.		
RESOLUTION	Accepted. Price protection comes from the slippage minimums.		
RESIDUAL	Documented.		

L - 05 No rescue path for dust or stray NFTs

LOW

ACCEPTED

LOCATION `Zxo`

ISSUE Refunds at or below the dust threshold, and NFTs sent directly to Zxo, can't be recovered.

RESOLUTION Accepted by design. No rescue function exists, so the owner has no path to any asset Zxo holds.

RESIDUAL Documented.

L - 06 ZxoVerify checked the wrong fee tier

LOW

RESOLVED

LOCATION `ZxoVerify`ISSUE It checked `feeAmountTickSpacing(500)` while the protocol uses the 100 tier.RESOLUTION It now checks the 100 tier. The deployer-address check is conditional on `verifyV3Deployer`.

RESIDUAL None.

L - 07 Treasury uses single-step ownership

LOW

ACCEPTED

LOCATION `ZxoTreasury`ISSUE A mistaken `transferOwnership` can't be undone.

RESOLUTION Accepted. Owner operations are handled carefully.

RESIDUAL Documented.

L - 08 Contract stakers without an NFT receiver could never unstake

LOW

RESOLVED

LOCATION `Zxo.unstake`ISSUE `safeTransferFrom` to `msg.sender` reverts for contracts, including some EIP-7702 wallets, that don't implement `onERC721Received`. That would lock their position permanently.RESOLUTION Added a to recipient (`address(0)` means `msg.sender`), a `to == address(this)` guard, and moved the transfer to the end. A failed transfer rolls back the whole unstake.

RESIDUAL None.

L - 09 Referrer can be set to the Zxo contract

LOW

ACCEPTED

LOCATION `Zxo._setRef`

ISSUE The referral share would be minted to Zxo and be unrecoverable. Only that user's own share is affected.

RESOLUTION Accepted. Any address may be a referrer, and `address(0)` defaults to the treasury.

RESIDUAL Documented.

I-01 Counter overflow on stats		INFO	RESOLVED
LOCATION	Zxo		
ISSUE	totalGlobalStakes and referralsPerUser_ were uint16, and the referral counter wrapped silently.		
RESOLUTION	Both widened to uint32.		
RESIDUAL	None.		
I-02 Silent uint16 truncation in updatePoolConfig		INFO	RESOLVED
LOCATION	Zxo.updatePoolConfig		
ISSUE	uint16(newValue) truncated oversized input, for example 70000 became 4464.		
RESOLUTION	Added ZxoLib._safe16, which reverts on overflow.		
RESIDUAL	None.		
I-03 Redundant guards kept		INFO	ACCEPTED
LOCATION	Zxo, ZxoLib		
ISSUE	Some checks can't be reached or are no-ops: userReferrer_[this], the rewardDebt write when pending == 0, and several library guards. The unreachable queryData type-3 branch was removed in Revision 2.1 (verified: that branch can only see type 0 or 3).		
RESOLUTION	Kept by choice as defense in depth. Unused enums, ownerOf checks and zero-token checks were removed.		
RESIDUAL	None.		
I-04 unstake truncated a uint256 token id		INFO	RESOLVED
LOCATION	Zxo.unstake		
ISSUE	The id was cast to uint64 to look up the position.		
RESOLUTION	The parameter is now uint64, matching harvest.		
RESIDUAL	None.		
I-05 Dead owner burn on ZxoToken		INFO	RESOLVED
LOCATION	ZxoToken, IZxoToken		
ISSUE	burn(address, uint256) could never be called but looked like a rug vector to scanners.		
RESOLUTION	Removed. The NatSpec was updated.		
RESIDUAL	None.		

I - 06	HOLDER_TOKEN decimals not verified	INFO	RESOLVED
LOCATION	ZxoVerify		
ISSUE	HOLDER_THRESHOLD assumes 18 decimals.		
RESOLUTION	Revision 2.1: ZxoVerify now requires HOLDER_TOKEN.decimals() == 18 and HOLDER_THRESHOLD ≥ 1 token. The threshold moved into each chain library, so every chain sets its own.		
RESIDUAL	None, provided ZxoVerify is run with the same ChainLib as Zxo.		
I - 07	_unpackPoolConfig returned 0 for an index above 5	INFO	RESOLVED
LOCATION	ZxoLib._unpackPoolConfig		
ISSUE	An out-of-range index fell through and returned 0 from queryData(type 2), which could be misread as no cap or not launched.		
RESOLUTION	Added revert() after index 5.		
RESIDUAL	None.		
I - 08	Single enum-coded event	INFO	ACCEPTED
LOCATION	Zxo.ZxoEvent		
ISSUE	Only user is indexed, so indexers must filter on the action value.		
RESOLUTION	Accepted. Each separate event adds bytecode, and one shared event keeps Zxo under the 24,576-byte limit while still emitting every action.		
RESIDUAL	Indexer ergonomics only.		
I - 09	Build pipeline hardening	INFO	RESOLVED
LOCATION	compile.py, deploy.py, run.sh		
ISSUE	The contracts imported SepoliaTestLib, compile.py ignored standard-JSON errors, the verification input depended on local files, and run.sh continued after a failed step.		
RESOLUTION	compile.py now builds deploy bytecode and explorer input from one standard-JSON compile with import-name source keys, stops on errors, checks EIP-170/3860 sizes, offers an EVM menu (Shanghai for PulseChain), and warns about test libraries or zero addresses. deploy.py checks the RPC chain id against the compiled ChainLib, probes PUSH0/TLOAD, estimates gas, checks receipt status and compares on-chain code with the build. run.sh stops at the first failure. The team confirmed with solc 0.8.36 that the input compiles with no local files and reproduces the build bytecode.		
RESIDUAL	The ChainLib must still be switched to the target chain for each production build. The tooling enforces this.		

I - 10 Holder settings are not checked by Zxo itself		INFO	RESOLVED
LOCATION	Zxo.stake, ChainLib (per chain)		
ISSUE	Revision 2.1 moved HOLDER_THRESHOLD into each ChainLib. Zxo trusts both values without checking them. With HOLDER_THRESHOLD = 0 every staker passes balance ≥ 0 and gets the +25% bonus. With HOLDER_TOKEN = 0x0 the balanceOf call reverts and staking is impossible. Eight mainnet libraries currently have both set to zero; only Arbitrum and the two test libraries are configured.		
RESOLUTION	The Zxo constructor now reverts when HOLDER_TOKEN is the zero address or HOLDER_THRESHOLD is below 1 token, so a misconfigured build cannot be deployed. Constructor code is not part of the runtime bytecode, so the check does not count toward the 24,576-byte limit. compile.py also warns at build time and ZxoVerify checks the same values on-chain.		
RESIDUAL	None.		
I - 11 TWAP oracle needs setup and is not checked on-chain		INFO	RESOLVED
LOCATION	ZxoOracle, Zxo constructor, set-observations.py		
ISSUE	Each new V3 pool stores one price observation, so observe() over 15 minutes fails while the pool is active, which blocks stake and harvest. ZXO_ORACLE was also not checked on-chain.		
RESOLUTION	Revision 2.3: the Zxo constructor now requires every ChainLib address (position manager, factory, HOLDER token, treasury, deployer, oracle) to be a deployed contract, checks the position manager's factory, requires fee tier 100 with tick spacing 1, HOLDER decimals of 18, the treasury and deployer markers, an unlinked deployer, and ZxoOracle.WINDOW() > 0. The per-pool history step is handled by set-observations.py, run once after initializePool(core).		
RESIDUAL	Operational: run set-observations.py for every new pool. If missed, stake and harvest pause for that pool until it is run; unstake and funds are unaffected.		
I - 12 SepoliaTestLib.sol did not compile		INFO	RESOLVED
LOCATION	libraries/SepoliaTestLib.sol		
ISSUE	The ZXO_ORACLE line has no address ("ZXO_ORACLE = " with nothing after it). Zxo, ZxoV3Deployer and ZxoVerify currently import this library, so the uploaded tree does not build until it is fixed or the import is switched.		
RESOLUTION	Revision 2.5: ZXO_ORACLE is now set in SepoliaTestLib, and update-chain-lib.py detects and repairs a line like this. The tree builds with the current import.		
RESIDUAL	None. Switch the import to the target chain for each production build (enforced by compile.py warnings).		

I - 13	ConfigUpdated could log a value that differs from the stored one	INFO	RESOLVED
LOCATION	Zxo.updatePoolConfig		
ISSUE	ConfigUpdated logs the requested value, but a target APR above the pool's cap is stored clamped, so the logged value can differ from the applied one (the AutoAdjust event that follows shows the applied target). PoolInitialized uses token0 as its indexed address, which is the pair token when Zxo is token1, while PoolCreated indexes the reward token.		
RESOLUTION	Revision 2.4: PoolInitialized now indexes tokens[2], the reward token, so it matches PoolCreated. Revision 2.5: updatePoolConfig now logs the value it stored (the capped targetApr, the referral rate or the new rewardRate) from a single _log call after the branches.		
RESIDUAL	None.		

I - 14	Owner is a single wallet on each chain	INFO	ACCEPTED
LOCATION	Zxo, ZxoV3Deployer (Arbitrum One, PulseChain)		
ISSUE	Zxo and ZxoV3Deployer on Arbitrum are owned by the deployer account 0xFf9623ABe5D081c3c4E586330ab714FcB718fa53, an externally owned account. No ownership transfer appears in the on-chain logs up to 1 October 2026. On PulseChain the owner is the deployer account 0x98C3eEc2b28D82b05edfA3D40F8A468a58BD1e24, with no transfer up to 5 October 2026.		
RESOLUTION	Accepted for now. The team will build a multisig after all mainnet deployments and then transfer ownership to it. Both contracts use two-step transfers.		
RESIDUAL	Until the transfer, one key can create pools, set target APR and referral rates within the contract limits, and collect core LP fees. It cannot touch staked positions, block unstaking, or withdraw core liquidity.		

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: Any user

- 1 **CHECK** Pool exists and is launched (stakes > 0).
User has fewer than 21 positions
- 2 **CHECK** Amounts, slippage $\leq$ 50% and minimum
stake validated
- 3 **CHECK** Pool is healthy: spot tick and 15-minute
TWAP 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. $\text{maxNormalAPR} \leq 320\%$. `pairToken` $\neq 0$
- 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 ≠ 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 Z XO_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	STEP 2	STEP 3	STEP 4	STEP 5
createPool	initializePool · core	startV3pool *n	finalizeToken	initializePool · stake
Deploys the ZxoToken. Checks pair decimals, mints coreLP + coreStake to Zxo and v3StartAmnt to the deployer.	Creates and initializes the pool, requires price == launch price, and mints the single-sided core LP.	Seeds external DEXes only. NFTs are sent to 0xdEaD. Never seed the main position manager here.	Burns the deployer's remaining balance. Permanently blocks further seeding.	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).

Oracle setup (Revision 2.2): after step 2, run `set-observations.py` for the new pool (it calls `increaseObservationCardinalityNext` with a size suited to the chain's block time). Staking and harvest stay closed until the pool has 15 minutes of price history.

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.

INVARIANT	WHY IT HOLDS
Record $\Rightarrow$ custody	A record with owner $\neq 0$ and liquidity > 0 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 ≥ 1</code> permanently.

Stake range and health (18/18 pair)

ZX0 PRICE	TICK	STATE
$> \sim 1.0305$	$> +300$	UNHEALTHY
$\sim 0.9704 - \sim 1.0305$	$(-300, +300)$	HEALTHY
$< \sim 0.9704$	< -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%.

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 ZXO_CONTRACT once.	Seed after finalization. Repoint ZXO_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.

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	ZX0 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]
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-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. Revision 3.0: the team keeps core stakes very small. On Arbitrum pool 0 the core stake is about 0.19 tokens beside a 361,486-token core LP, so the dilution is negligible once users stake more than a few tokens. Revision 3.1: on PulseChain the core stakes are larger in token terms (pool 0 is 110,385,144 tokens) but small in value: about \$330 to \$1,030 per pool and about \$4,900 across nine pools at DEX prices on 5 October 2026 (the team reports \$1,078.57 for pool 0 and about \$200 to \$600 for the others). Each amount is immutable in the reward token and minted once. Stakers earn about 62% of target when user stakes equal the core stake, about 90% at ten times and about 99% at one hundred times. On 5 October stakers in pools 6 and 8 were at about 90% and 99.9%; the other pools held only test-size stakes, where stakers earn about half of target.
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.
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.

ID	RISK	TEAM POSITION
I - 03	Redundant guards kept Some checks can't be reached or are no-ops: <code>userReferrer_[this]</code> , the <code>rewardDebt</code> write when <code>pending == 0</code> , and several library guards. The unreachable <code>queryData</code> type-3 branch was removed in Revision 2.1 (verified: that branch can only see type 0 or 3).	Kept by choice as defense in depth. Unused enums, <code>ownerOf</code> checks and zero-token checks were removed.
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.
I - 14	Owner is a single wallet on each chain Zxo and ZxoV3Deployer on Arbitrum are owned by the deployer account <code>0xFf9623ABe5D081c3c4E586330ab714FcB718fa53</code> , an externally owned account. No ownership transfer appears in the on-chain logs up to 1 October 2026. On PulseChain the owner is the deployer account <code>0x98C3eEc2b28D82b05edfA3D40F8A468a58BD1e24</code> , with no transfer up to 5 October 2026.	Accepted for now. The team will build a multisig after all mainnet deployments and then transfer ownership to it. Both contracts use two-step transfers.

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.

10

Arbitrum One deployment

The contracts below were deployed on 28 September 2026 and verified on Sourcify with an exact match of creation and runtime bytecode. The full evidence, including file hashes, configuration and the on-chain activity reviewed, is in the Zxo Arbitrum Deployment Statement.

CONTRACT	ADDRESS	BLOCK	SOURCE AGAINST REVISION 2.5
ZxoTreasury	0x0341A7a6CC031662526492e10796E551125AE2bc	509546642	Identical (hash match)
ZxoOracle	0x98c9695bb012A42FDcB734324A78E63eE6c23D44	509547888	Identical (hash match)
ZxoV3Deployer	0xb6CB455CB67d59DD269C865428acece75eA08594	509561805	Identical apart from the ArbitrumLib import (hash match)
Zxo	0x5356B3004D0394dEA51161d175265EC8F92c96b7	509563331	No code difference in Zxo.sol or ZxoLib.sol (line match); ABI matches
ZxoToken (pool 0)	0x304441b6BDADA52d41f4AA0A0b13dC4185B3979B	509698291	Identical (hash match)

Confirmed on-chain

- Deploy order: Treasury, Oracle, V3 Deployer, Zxo, then `setZxoContract` once
- All constructor checks passed, so the Arbitrum configuration is valid
- Six pools launched; every reward token finalized before its core stake
- 11 stakes, 8 harvests, 7 unstakes and 10 config changes, all successful
- A reward claim reproduced exactly from the audited formula
- The deployer's testnet-only validator is the zero address

Read this with care

- All activity so far came from the deployer wallet. It is a team test on mainnet, not public usage
- The owner is a single wallet until the multisig (I-14)
- Every pool uses the 320% APR cap; pool 0 targets the 640% first-year ceiling, limited by its 30-day cap
- Referral rate is 30% on pools 0 and 4
- No contract state was read directly; owner and settings are inferred from events up to 1 October 2026

11

PulseChain mainnet deployment

The contracts below were deployed on 3 October 2026 and verified on Sourcify with an exact match of creation and runtime bytecode. The main DEX is LibertySwap, a PancakeSwap-style V3 DEX. The full evidence is in the Zxo PulseChain Deployment Report.

CONTRACT	ADDRESS	BLOCK	SOURCE AGAINST THE AUDITED CODE
ZxoTreasury	0x923cbe1C05e9c0014D6514a506A8E74B9254FF86	27704288	Identical (hash match)
ZxoOracle	0xC8CaA5F5961d4679C79A18731C53b6023Ac8862B	27704319	Identical (hash match)
ZxoVerify (first run)	0xa8D7A6A9291E60400F0CA64c5dfed64997Cd3817	27704333	Identical apart from the PulseChainLib import (hash match)
ZxoV3Deployer	0x3D62C13A12F246A2CAC2d7fCe1A43D63ED253FBE	27704349	Identical apart from the PulseChainLib import (hash match)
ZxoVerify (second run)	0x923028273B38b24fb98F846f0e4c809fAB58523f	27704375	Identical apart from the PulseChainLib import (hash match)
Zxo	0x813984963d4e0CC7fF5d04E830767fe63D815B0A	27704394	No difference in Zxo.sol, ZxoLib.sol or ZxoToken.sol (line match); ABI matches

Confirmed on-chain

- Deploy order: Treasury, Oracle, ZxoVerify, V3 Deployer, ZxoVerify, Zxo, then `setZxoContract` once
- Nine pools launched; every reward token finalized before its core stake
- Price reads and the 15-minute average work on LibertySwap pools (H-02 fix confirmed live)
- 16 stakes, 6 harvests, 2 unstakes, 17 config changes and 1 core LP fee collection
- A reward claim reproduced to the last wei; HOLDER bonus exactly 1.25 times
- Four addresses other than the deployer staked or harvested

Read this with care

- 94 of 100 transactions came from the deployer wallet
- The owner is a single wallet until the multisig (I-14)
- Target APRs range from 10% to 510%; referral rates from 3% to 30%
- Core stakes are small, about \$330 to \$1,030 per pool. Stakers earn about 90% of target once user stakes are ten times the core stake, and about half only while a pool holds test-size stakes (M-03)
- Two early `initializePool` calls reverted on a gas limit that was too low; nothing changed on-chain
- No contract state was read directly; owner and settings are inferred from events up to 5 October 2026

12

Checklist status

This replaces the pre-mainnet checklist. Verified means this review saw the evidence in verified source or on-chain records. Team-reported means the development team states it is done and this review did not check it. Open means not done or not evidenced.

Build and configuration

ITEM	STATUS	EVIDENCE
Switch the ChainLib import to the target chain and set the three Zxo addresses	VERIFIED	The Arbitrum and PulseChain deployments import their own chain library with all three addresses set (hash match on PulseChain; hash and line match on Arbitrum).
No "stack too deep" in createPool and initializePool	VERIFIED	Zxo compiled and deployed on Arbitrum and PulseChain mainnet with via-IR off.
Zxo runtime bytecode under 24,576 bytes	VERIFIED	The Arbitrum and PulseChain deployments succeeded, which the size limit would otherwise prevent.
compile.py stops on standard-JSON errors	VERIFIED	Done in Revision 2.
Verification input compiles with no local files and matches the build	VERIFIED	All eleven contracts on Arbitrum and PulseChain are exact-match verified on Sourcify from that input.
EVM version Shanghai	VERIFIED	Sourcify reports evmVersion shanghai, optimizer 200 runs, via-IR off for every deployed contract.
Set HOLDER_TOKEN and HOLDER_THRESHOLD in each chain library	OPEN	Set for Arbitrum (token 0x6F29...E66A) and PulseChain (token 0xDB3f...40d6), each with a 3.69 billion threshold; the HOLDER bonus applied correctly on PulseChain. Still zero on the other 7 mainnets; the team will set each before deployment and the constructor rejects zero values.
Run ZxoVerify before and after ZxoV3Deployer	VERIFIED	On PulseChain both ZxoVerify deployments are on-chain, exact-match verified, and succeeded (ZxoVerify reverts unless its checks pass). On Arbitrum the team reports the same steps; no ZxoVerify address was provided there, and the Zxo constructor repeats the checks.
Run Slither or Aderyn and triage the results	OPEN	The team note reports successful testing but does not say a static analyzer was run. No report was provided.
Move Zxo and deployer ownership to a multisig	OPEN	Deferred by the team until after all mainnet deployments (I-14). The owner on Arbitrum and on PulseChain is a single wallet.

Tests

ITEM	STATUS	EVIDENCE
Read slot0 on every target DEX	VERIFIED	Works live on Uniswap V3 (Arbitrum) and on LibertySwap, a PancakeSwap-style V3 DEX (PulseChain, 16 stakes and 6 harvests). PancakeSwap on BNB Chain uses the same pool design and has not been run yet.
Stake a USDT pair on Ethereum	OPEN	Not evidenced. Fork-test before the Ethereum deployment.
Full launch order; core stake reverts until finalizeToken	VERIFIED	Six pools on Arbitrum and nine on PulseChain followed the order: every TokenFinalized event precedes the core-stake mint. The team confirms the revert when finalization is skipped.
initializePool(core) reverts on a pool pre-initialized at another price	TEAM-REPORTED	Confirmed by the team.
createPool reverts when decimals don't match	TEAM-REPORTED	Confirmed by the team.
Unstake to a contract without an NFT receiver reverts; retry with another recipient works	TEAM-REPORTED	Confirmed by the team. Note from the team: a staking contract must itself be able to pass the <code>to</code> argument, or it cannot recover.
Invariant: totalStaked equals the sum of position liquidity	OPEN	No invariant test run was reported. On-chain, claims on both chains matched the formula to the last wei, which is consistent with the invariant. The core stake is included by design.
Invariant: totalTokens equals the sum of position tokens	OPEN	No invariant test run was reported. On-chain, the staked value used for emissions matched the core-stake amount.
Invariant: a staker can always unstake after the 1-day lock-up	VERIFIED	Nine mainnet unstakes on two chains succeeded after 24.4 to 37.4 hours. The code has no price or owner gate on unstake.
Independent re-audit of the diff since Revision 1	TEAM-REPORTED	The team states that all audits passed. No third-party report was provided to this review.

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)`.
- New event action codes: 8 Deployed, 9 PoolCreated, 10 CoreFeesCollected, 11 ConfigUpdated, 12 PoolInitialized. Codes 0–7 are unchanged.
- `tokensIds_(address)` was removed. Map a reward token to its pool by scanning `poolInfo(i).rewardToken`.
- `queryData(..., 4)` "healthy" now also requires the 15-minute TWAP in range. Use `ZxoOracle.twapTick(pool)` to show the average price and explain a paused pool.
- 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 `OxdEaD`.

13

Scope and file fingerprints

Revision 3.0 adds the Arbitrum One deployment review of 2 October 2026 and Revision 3.1 the PulseChain mainnet review of 5 October 2026. The file fingerprints below are the Revision 2.5 working files (27 September 2026), without the license header. The deployed files carry the BUSL-1.1 header and the ArbitrumLib import, so their hashes differ; the keccak256 values of the deployed files are listed in the Arbitrum Deployment Statement and the PulseChain Deployment Report. 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 (vialR 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 `vialR` off.

FILE	LINES	SHA-256 (PREFIX)	ROLE
Zxo.sol	1067	8c8c2947cfe6	Farm, staking, emissions, core LP

FILE	LINES	SHA-256 (PREFIX)	ROLE
libraries/ZxoLib.sol	502	2af09f5c47af	Ticks, bonus, packing, validation
ZxoV3Deployer.sol	331	3c8fb36af7dd	External DEX seeding and finalization
ZxoTreasury.sol	231	aa47be5c2210	Fee and referral receiver
ZxoVerify.sol	211	02ff34b11012	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
ZxoOracle.sol	26	6ac1f047bc61	15-minute TWAP tick reader
interfaces/IZxoOracle.sol	17	bb73eefaf56f	twapTick, inRange, WINDOW
libraries/*Lib.sol (11 chains)	21–22 each	various	Per-chain addresses and HOLDER threshold (all addresses 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. Live behavior was reviewed on Arbitrum One and PulseChain mainnet only, from verified source and event records. An independent professional audit is recommended for contracts that hold user funds.