XORS Auditing Report

FOR

o1-launch

https://o1.exchange/

BY

XORS Software

XORS Software
June 29, 2026

► **Prepared For:**

o1-launch
https://o1.exchange/

► **Prepared By:**

Xiangan He & Ahmed Panju

► **Contact Us:** x@xors.xyz

Contents

1 Executive Summary

XORS conducted a security assessment of the o1-launch launchpad contracts, combining Slopless automated analysis with an independent first-principles review backed by executable Foundry proof-of-concepts.

Summary of issues detected. The assessment uncovered **7 issues**, **0 of which are assessed to be of high severity or above**. The severity distribution is as follows:

- **0 Critical-severity issues**
- **0 High-severity issues**
- **0 Medium-severity issues**
- **7 Low-severity issues**

All seven findings are low severity. Three are fully resolved in code (XORS-L1, L3, L6); two are mitigated with a documented owner-trust residual (L2, L4); one is an intentional opt-in behavior, now documented (L5); and one is accepted by design (L7). See the status assigned to each finding. Code remediations are delivered via XORS pull requests — L1–L5 and L3 in the remediation PR, and L6 in the independent proof-of-concept and hardening PR — which are open and pending merge to main at the time of writing.

Disclaimer. This report is generated by automated tooling and provides no warranty of any kind, explicit or implied. The contents should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall XORS or any of its employees be liable for any claim, damages or other liability arising from, out of or in connection with the results reported here.

About Slopless. Slopless is XORS's automated security analysis engine for smart contract codebases. It runs a multi-perspective adversarial cognitive loop, surfacing candidate vulnerabilities across reentrancy, access control, arithmetic, business logic, oracle, flash-loan, and upgradeability dimensions. Each candidate is then cross-validated by a prosecutor and defense pair before it is confirmed in the report. Manual review and Foundry proof-of-concept exploits supplement the automated output where deeper protocol-specific reasoning is required.

2 Project Dashboard

Table 2.1: Application Summary.

Name	Version	Type	Platform
o1-launch	`HEAD - HEAD`	Solidity 0.8.26	EVM

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Jun. 29, 2026	Slopless + Foundry PoC	2	2 person-day

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	0	0
Low-Severity Issues	7	3
Warning-Severity Issues	0	0
Informational-Severity Issues	0	0
TOTAL	7	3

Table 2.4: Category Breakdown.

Name	Number
Security	7

3 Audit Goals and Scope

3.1 Audit Goals

The scan was scoped to provide a comprehensive automated security assessment of $O token contract. The analysis sought to identify security vulnerabilities, code quality issues, and reliability concerns using Slopless static analysis.

3.2 Classification of Vulnerabilities

Findings are classified by severity: Critical, High, Medium, and Low, based on the potential impact and likelihood of exploitation.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

4 Vulnerability Report

In this section, we describe the vulnerabilities found during the Slopless scan. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status. Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
XORS-L1	Unprotected initializer — LaunchHook.initialize guarded by a custom flag, not the initializer modifier	Low	Resolved
XORS-L2	Factory owner can redirect platform fees to an arbitrary treasury via setFeeDefaults	Low	Mitigated
XORS-L3	Referrer self-referral recaptures the platform fee slice in _distribute	Low	Resolved
XORS-L4	Factory owner can grief/censor launches by repeatedly bumping configVersion	Low	Mitigated
XORS-L5	METADATA_ROLE retained by creator can rename the token, breaking the EIP-712 domain	Low	Documented
XORS-L6	currency0 token ordering bricks at extreme owner-set tick frames	Low	Resolved
XORS-L7	Exact-output swaps revert during the anti-snipe window	Low	Acknowledged

4.1 Detailed Description of Issues

4.1.1 XORS-L1: Unprotected initializer — LaunchHook.initialize guarded by a custom flag, not the initializer modifier

Severity	Low	**Type**	Security
Status	Resolved	**Location**	121
File(s)	contracts/src/LaunchHook.sol:121		
CWE	CWE-665		

Description. LaunchHook.initialize() (LaunchHook.sol:121) sets the critical factory and feeEscrow addresses. It was guarded only by a custom wired boolean (LaunchHook.sol:69) and a msg.sender == deployer check, rather than OpenZeppelin's initializer modifier — the standard, audited protection for this pattern. The deployer is the deploy EOA/script address set in the constructor. The residual exposure is the multi-transaction deploy window between hook deployment and the initialize() call: if the deployer key is compromised in that window, an attacker controlling the deployer address could wire a malicious factory and feeEscrow before the legitimate wiring transaction is mined.

Impact. Severity is assessed as Low. Wiring the hook to an attacker-controlled factory and fee escrow would redirect all platform fees and could brick future launches. Exploitation requires control of the deployer address or a front-run of the one-time wiring transaction, so the window is narrow and operator-dependent rather than openly exploitable.

Recommendation. Use OpenZeppelin's Initializable with the initializer modifier, and/or deploy and wire the hook atomically in a single transaction to remove the front-run window.

Resolution. Resolved. LaunchHook now inherits Initializable and initialize() carries the initializer modifier, so re-initialization reverts with InvalidInitialization; the deploy script wires atomically and asserts the wiring. Covered by test_initializeGuards.

4.1.2 XORS-L2: Factory owner can redirect platform fees to an arbitrary treasury via setFeeDefaults

Severity	Low	**Type**	Security
Status	Mitigated	**Location**	588
File(s)	contracts/src/LaunchpadFactory.sol:588		
CWE	CWE-269		

Description. LaunchpadFactory.setFeeDefaults (LaunchpadFactory.sol:588, onlyOwner) sets the FeeDefaults, including platformTreasury, which is read at launch and frozen into each pool's PoolConfig. The owner can point platformTreasury at an address they control and skew the splits — for example creatorBps = 0, referrerBps = 0, and platformBps at the MAX_BASE_FEE_BPS cap — so that the platform's fee share on all subsequently launched pools flows to that address. Combined with aggressive anti-snipe parameters, nearly the entire swap fee on newly launched pools can be routed to the owner-chosen treasury.

Impact. Severity is assessed as Low. This is a privileged owner action and affects only pools launched after the change — existing pools' configs are already frozen. No unprivileged actor can trigger it, so it is a centralization / owner-trust risk rather than an externally exploitable bug.

Recommendation. Make treasury changes observable and place factory ownership behind a multisig + timelock; optionally bound the platform split in code as a product decision.

Resolution. Mitigated. setFeeDefaults now emits FeeDefaultsUpdated carrying the indexed treasury and headline fee fields, so a hostile reconfiguration is observable on-chain before any launch. The residual is an owner-trust assumption, addressed operationally by placing factory ownership behind a multisig + timelock (a stated mainnet condition). An in-code platformBps cap was considered and deferred as a product decision — it does not constrain the redirect vector, since the owner controls the treasury address regardless of the split.

4.1.3 XORS-L3: Referrer self-referral recaptures the platform fee slice in _distribute

Severity	Low	**Type**	Security
Status	Resolved	**Location**	256
File(s)	contracts/src/LaunchHook.sol:256		
CWE	CWE-840		

Description. In LaunchHook._distribute (LaunchHook.sol:256), the referrer address is decoded from caller-controlled swap hookData via _parseHookData with no validation against the creator or the platform treasury. referrerShare is computed as referrer == address(0) ? 0 : baseFee * cfg.referrerBps / BPS (LaunchHook.sol:270) and credited to the referrer; when referrer is zero it instead rolls into the platform share. A creator (or any swapper) who passes a chosen address as referrer therefore captures fee that would otherwise accrue to the platform. This finding consolidates two cases: the targeted creator / platform-treasury self-referral, and the broader case where any swapper passes its own address to recapture the referrer slice on every trade.

Impact. Severity is assessed as Low and is platform-revenue-only. The creator share, other recipients, and pool/escrow solvency are untouched, and value conservation holds exactly (the credits still sum to totalFee, equal to the ERC-6909 claim minted to the escrow). No theft, insolvency, or privilege escalation; the dilution scales with referrerBps (20% of the base fee at defaults).

Recommendation. Reject self-referral against known protocol addresses, and disclose the general open-referral behavior in the fee-model documentation.

Resolution. Resolved for the actionable case: _distribute now rejects referrer == cfg.creator and referrer == cfg.platformTreasury, rolling the slice into the platform (covered by tests). The general open-referral case cannot be cleanly blocked on-chain — afterSwap's sender is the swap router, not the trader's EOA, and tx.origin checks break smart-contract-wallet and aggregator traders — so it is accepted as a documented fee-model property: the platform's take is understood as the worst case where every swap self-refers. A proof-of-concept confirms the behavior is exactly value-conserving.

4.1.4 XORS-L4: Factory owner can grief/censor launches by repeatedly bumping configVersion

Severity	Low	**Type**	Security
Status	Mitigated	**Location**	672
File(s)	contracts/src/LaunchpadFactory.sol:672		
CWE	CWE-269		

Description. createLaunch requires p.expectedConfigVersion == configVersion (LaunchpadFactory.sol:189–190, reverting StaleConfig otherwise). Governance setters bump configVersion via configVersion++ (LaunchpadFactory.sol:672). A malicious or compromised owner could repeatedly call a no-op governance change to bump configVersion faster than a targeted creator can land a valid createLaunch transaction, indefinitely blocking that creator; the deadline parameter offers no protection since the owner can keep bumping until it expires.

Impact. Severity is assessed as Low. This is privileged griefing / censorship of specific creators; it causes no fund loss and does not affect already-launched pools. It requires a hostile or compromised owner.

Recommendation. Stop bumping configVersion on no-op changes and route governance through a timelock so any real config churn is rate-limited and observable.

Resolution. Mitigated. No-op governance setters no longer bump configVersion, closing the spam vector (verified by test_xorsL4_noOpSettersDoNotBumpConfigVersion). The residual is an owner-trust assumption, addressed operationally by timelock-routing factory governance.

4.1.5 XORS-L5: METADATA_ROLE retained by creator can rename the token, breaking the EIP-712 domain

Severity	Low	**Type**	Security
Status	Documented	**Location**	291
File(s)	contracts/src/LaunchpadFactory.sol:291		
CWE	CWE-269		

Description. When a creator launches with roleMode bit0 set (keepMetadata = true), the factory grants METADATA_ROLE to the creator (LaunchpadFactory.sol:291; the roleMode semantics are documented at LaunchpadFactory.sol:126). That role lets the creator change the token name and symbol after launch. Because the EIP-712 signing domain includes the token name, a post-launch rename invalidates outstanding signed permits and approvals and can be used to deceive users — for example, launch under a reputable name, build trust and volume, then rename to confuse holders or invalidate pending signed transactions.

Impact. Severity is assessed as Low. The exposure is limited to launches that opt into keepMetadata, and it is exercised by the creator over their own token; the effect is signature invalidation and user confusion rather than direct theft of funds.

Recommendation. Disclose the keepMetadata trust assumption prominently, and surface the rename / EIP-712 implication in the launch UI for keepMetadata launches; optionally freeze the EIP-712 name.

Resolution. Documented. The keepMetadata / METADATA_ROLE trust assumption is now disclosed prominently in code comments and the project docs. It is an intentional, opt-in behavior; removing the capability would require out-of-scope B20-side changes. Surfacing the assumption in the launch UI remains a recommended product follow-up.

4.1.6 XORS-L6: currency0 token ordering bricks at extreme owner-set tick frames

Severity	Low	**Type**	Security
Status	Resolved	**Location**	_validateStartTickFrame
File(s)	contracts/src/LaunchpadFactory.sol		
CWE	CWE-20		

Description. _validateStartTickFrame bounded the quote start-tick frame only to [MIN_TICK, MAX_TICK]. For a launch whose token sorts as currency0, the pool initializes one tickSpacing below the start tick (initTick = startTick - tickSpacing in _openAndSeed). An owner-set frame within one tick-spacing of MIN_TICK therefore produced initTick < MIN_TICK and reverted with an opaque Uniswap v4 TickMath.InvalidTick at launch. Because which ordering a token takes is determined by its CREATE2 address (salt-dependent), the same configuration launched successfully for some creators and bricked for others.

Impact. Severity is assessed as Low. Liveness only, and owner-misconfiguration-gated: createLaunch reverts atomically — no token is partially created, no quote is pulled, and no funds are at risk — and existing launches are unaffected. The defect is the opaque, ordering-asymmetric failure mode rather than any value loss.

Recommendation. Reserve one tickSpacing of headroom at both tick extremes at registration time, and range-check initTick at seed time.

Resolution. Resolved. _validateStartTickFrame now reserves one tickSpacing of headroom at both extremes, rejecting the dangerous frame at registration with a clean OutOfBounds (immediate owner feedback); _openAndSeed adds a belt-and-suspenders initTick range check for the case where setTickSpacing widens the spacing after a frame was registered. Confirmed pre-fix (createLaunch reverted InvalidTick(-887400) for a currency0 token at frame -887200); regression test test_L01_extremeFrameRejectedReservesTickHeadroom.

4.1.7 XORS-L7: Exact-output swaps revert during the anti-snipe window

Severity	Low	**Type**	Security
Status	Acknowledged	**Location**	_afterSwap
File(s)	contracts/src/LaunchHook.sol		
CWE	CWE-703		

Description. While the anti-snipe surcharge is active (the first antiSnipeWindowBlocks, default 8 ≈ 16s on Base), exact-output swaps revert with ExactOutputDisabledDuringAntiSnipe. This is deliberate: the Uniswap v4 afterSwap return delta is denominated in the unspecified currency, which on an exact-output swap is the input — so a near-99% surcharge cannot be metered correctly against gross spend, and the hook fails closed. Exact-output auto-resumes at the base fee once the window elapses.

Impact. Severity is assessed as Low. Short-lived unavailability of exact-output router/aggregator paths on a freshly launched pool. Exact-input is fully available throughout; the reverted swap accrues no fee and no funds are at risk. The behavior keys off the caller's own amountSpecified sign plus the global block counter, so it is not attacker-controllable against a third party.

Recommendation. Accept as designed; document the window for integrators so routers detect the revert and fall back to exact-input or wait the window out.

Resolution. Acknowledged — by design. The fail-closed behavior is intentional and the window is documented for integrators. Covered in-tree by test_exactOutputRejectedDuringAntiSnipeAndAllowedAfter, which asserts the revert during the window (zero fee accrued) and a successful base-fee exact-output swap afterward.