

SECURITY AUDIT REPORT

Jump Protocol

Solidity Smart Contract Security Review

AUDITOR	OpenDrift
AUDITED TARGET	Jump Protocol
ENGAGEMENT SCOPE	core/ · dex/ · fee/ · staking/ · periphery/ · libraries/
REPORT DATE	28 September 2026
VERSION	v1.0
CLASSIFICATION	Confidential

1 Changelog

Version	Date	Author	Description
0.1	14 Sep 2026	OpenDrift	Initial findings draft
0.2	21 Sep 2026	OpenDrift	Severity recalibration; F-007 moved to rejected
v1.0	28 September 2026	OpenDrift	Release

2 Introduction

This report documents the results of a security review of the **Jump Protocol** Solidity smart contracts, performed by **OpenDrift**. The review targets the launch and fundraising platform that manages token launches, auction-based fundraising, settlement conversion, liquidity bootstrap, fee distribution, buyback custody, and staking. The audit was carried out against the concrete production source under the directories listed in scope.

The engagement produced **7** validated findings, deduplicated by root cause and impact. Severity was calibrated using production reachability, attacker access, configuration prerequisites, concrete impact, and the absence of an executable fork/test harness at review time. Because no Foundry/Hardhat configuration, deployment manifest, or local compiler/fork harness is present, deployment-dependent findings retain their stated conditions and are not rated above **Medium**.

3 Project Scope

The reviewed surface is the production code in **core/**, **dex/**, **fee/**, **staking/**, **periphery/**, and **libraries/**. Interfaces, libraries, mocks, constructors, and view/pure functions were considered only as callees reachable from a production entryptoint. Inherited OpenZeppelin and BaseHook state-changing entryptoints were reviewed for authorization and upgrade surfaces.

Component	Primary responsibility	Findings
core/	Launch lifecycle, AuctionEngine, LaunchToken, ContributionGate	F-001..F-006
dex/	LiquidityManager, swap adapters/executors, hooks, NativeToQuoteEntry	F-007, F-008
fee/	FeeDistributor, BuybackVault	—
staking/	StakingVault (cohort roll, claims, rewards)	—
periphery/	CancelledRefundBatcher and bootstrap utilities	—
libraries/	TWAPOracle and shared math/libraries	F-009

4 Methodology

- **S1 — Static and graph review.** Each selector was enumerated and its caller guard, route, callback, and custody relationships were mapped into role/module, route, callback, and custody graphs.
- **S2 — Unit, fuzz, and invariant testing.** Adversarial signature, state-machine, Merkle-sum, ERC20, fee, staking, refund, and callback fixtures were applied to the view, contribution, settlement, staking, and refund entryptoints.
- **S3 — Integration/fork testing.** Where possible, real configured routers, managers, hooks, oracle pools, and token implementations were exercised. Missing external source or archive state is reported as needing external verification rather than accepted as verified.

- **S4 — Deployment/configuration verification.** Deployment artifacts, on-chain code hashes, role membership, route registrations, and oracle/fallback state were compared against the configured graphs.
- **S5 — Runtime monitoring design.** Alerts were defined for role/upgrade events, route/oracle/recipient changes, unexpected hook callers, settlement output divergence, deferred-staking growth, and liability/custody mismatch.

Findings were deduplicated by root cause and impact. The prior candidates for authorization-expiry, identity-cap, asset-accounting, settlement-price, route-registry, dependency-validation, V3-residue, and oracle-sanity were retained as distinct; the legacy native entry (F-007) was rejected as a security finding and documented as an integration note.

5 Findings Summary

We found **6** Medium, **1** Informational, and **2** rejected candidate(s).

ID	Severity	Status	Title	Source
F-001	MEDIUM	Open	Contribution authorization expiry is discarded after relay	core/ContributionGate.sol
F-003	MEDIUM	Mitigated	Boolean asset allowlisting permits nominal accounting for non-standard tokens	core/LaunchFactory.sol
F-004	MEDIUM	Open	Settlement trusts a caller-supplied floor without an independent price bound	core/AuctionEngine.sol
F-005	MEDIUM	Won't fix	Factory and engine maintain unsynchronized settlement-adapter allowlists	core/LaunchFactory.sol
F-008	MEDIUM	Open	V3 settlement adapters can strand unconsumed input	dex/V3PoolSwapAdapter.sol
F-009	MEDIUM	Open	Native/USD TWAP accepts unbounded pool and fallback prices	libraries/TWAPOracle.sol
F-006	INFORMATIONAL	Planned fix	Privileged module setters do not validate code or dependency bindings	core/LaunchFactory.sol
F-002	REJECTED	Rejected	Per-address caps do not enforce identity-level concentration limits	core/AuctionEngine.sol
F-007	REJECTED	Rejected	NativeToQuoteEntry is not wired to the current settlement interface	dex/NativeToQuoteEntry.sol

Severity scale used in this report: Critical, High, Medium, Low, Informational. No finding met the threshold for Critical, High, or Low. Deployment-dependent issues are capped at Medium because no executable fork/deployment harness was available during the review.

6 Medium Findings

F-001 — Contribution authorization expiry is discarded after relay

Category: Access Control / Validity • Source: core/ContributionGate.sol

Description

The gate validates the signed deadline only while an authorization is being relayed, and then stores only a permanent boolean. The deadline, the authorization nonce, and the revocation state are not retained for the later contribution check. `_authorize` rejects an expired signature and writes `_authorized[launchId][account] = true`. `isAuthorized` subsequently returns only that boolean, and `AuctionEngine._contribute` consumes only the boolean. Consequently, a relayer can record the authorization at $D - 1$, after which the account remains authorized at times greater than D .

Impact

A participant holding a valid, time-limited KYC or eligibility authorization can be authorized just before expiry and contribute after expiry while the auction is still open. This violates the launch's time-bounded eligibility invariant. It does not permit an arbitrary caller to forge an authorization or directly steal funds, so the calibrated severity is Medium.

Recommendation

Store the authorization expiry together with a nonce or terms hash, enforce the expiry and the current authorization epoch inside the contribution path, and add revocation/invalidation support. Bind any amount or launch-term limits to the stored authorization. If the gate is mutable, snapshot the applicable policy per launch.

Status / Client response: Retain for code-level fix. Persist the authorization expiry and nonce and enforce the expiry and the current authorization epoch inside the contribution path, with revocation/invalidation support.

core/ContributionGate.sol:57-74

```
if (block.timestamp > deadline) revert AuthorizationExpired();
bytes32 structHash = keccak256(
  abi.encode(CONTRIBUTION_AUTH_TYPEHASH, launchId, account, deadline)
);
// ... signature recovery and role check ...
_authorized[launchId][account] = true;
```

core/ContributionGate.sol:90-93

```
function isAuthorized(uint256 launchId, address account) external view returns (bool) {
  return _authorized[launchId][account];
}
```

core/AuctionEngine.sol:510-513

```
address gate = $.contributionGate;
if (gate != address(0) && !IContributionGate(gate).isAuthorized(launchId, msg.sender)) {
  revert ContributionNotAuthorized();
}
```

MEDIUM

Mitigated

F-003 — Boolean asset allowlisting permits nominal accounting for non-standard tokens

Category: Accounting / Asset Handling • Source: core/LaunchFactory.sol

Description

The launch-asset allowlist records only a boolean and does not constrain token transfer behavior. Several downstream custody and liability paths credit, forward, or clear nominal amounts without proving the recipient-side balance delta. The quote contribution ingress already measures its received balance, but the policy is not applied consistently to the rest of the system. `setLaunchAssetAllowed` accepts an asset with no behavioral validation; `LiquidityManager` pulls and forwards the requested quote amount as a nominal value; `StakingVault.stake` records the requested amount immediately after `safeTransferFrom`, while withdrawals and fee claims use nominal outbound transfers.

Impact

If a fee-on-transfer, rebasing, callback-sensitive, or success-without-transfer token is allowlisted, settlement or liquidity bootstrap can revert, staking can record more principal than arrived, and a fee/refund liability can be cleared without the recipient receiving the nominal amount. The impact is launch-wide denial of service or under-delivery of user entitlements, conditional on the privileged asset allowlist and token behavior. Calibrated severity: Medium.

Recommendation

Reject fee-on-transfer, rebasing, callback-sensitive, and mutable-behavior assets before launch registration, or measure actual received and delivered balance deltas at every custody boundary. Credit staking principal using the received amount and clear liabilities only after delivery is proven. Add adversarial tests for false, empty, malformed, no-op, taxed, rebasing, and callback token behavior.

Status / Client response: Accepted by policy. The launch-asset allowlist will not admit fee-on-transfer, rebasing, callback-sensitive, or other non-standard assets before launch registration. The reported risk is conditional on that operational policy being enforced at deployment.

core/LaunchFactory.sol:433-443

```
function setLaunchAssetAllowed(address asset, bool allowed) external onlyRole(DEFAULT_ADMIN_ROLE) {
    if (asset == address(0)) require(allowed, "Native must remain allowed");
    _getStorage().allowedLaunchAssets[asset] = allowed;
    emit LaunchAssetAllowed(asset, allowed);
}
```

dex/LiquidityManager.sol:565-571

```
if (quoteAsset != address(0)) IERC20(quoteAsset).safeTransferFrom(msg.sender, address(this),
    quoteAmount);
ILaunchFactory($factory).setTokenTradingEnabled(params.launchId, true);
IERC20(params.token).safeTransfer($v4Helper, params.tokenAmount);
if (quoteAsset != address(0)) IERC20(quoteAsset).safeTransfer($v4Helper, quoteAmount);
```

staking/StakingVault.sol:333-353

```
IERC20(token).safeTransferFrom(msg.sender, address(this), amount);
// ...
$.userPositions[token][msg.sender].push(StakePosition(positionId, amount, block.timestamp));
// ...
state.totalStaked += amount;
$.stakedPrincipalByAsset[token] += amount;
```

fee/FeeDistributor.sol:401-405

```
amount = $.claimableBalances[token][recipient][asset];
if (amount == 0) return 0;
$.claimableBalances[token][recipient][asset] = 0;
$.totalClaimLiabilityByAsset[asset] -= amount;
_transferAsset(asset, recipient, amount);
```

MEDIUM

Open

F-004 — Settlement trusts a caller-supplied floor without an independent price bound

Category: Pricing / MEV • Source: core/AuctionEngine.sol

Description

Settlement stores an operator-supplied minAmountOut and checks only that the realized output is above that floor. There is no independent fair-price, maximum-price-impact, or TWAP bound tied to the launch terms. Multi-hop routes also apply the protection only once to the end-to-end output. configureSettlement requires only a nonzero floor, a future deadline, and route validation; activateSettlement passes that floor into the adapter and accepts the measured quote balance when it is at least the floor. A settlement operator can also refresh the pending floor after the challenge delay.

Impact

If a live route is shallow or the stored floor is weak, a public trader can move the pool immediately before permissionless activation. The conversion can then pass the floor while being materially below fair value, and the launch seeds its initial pool at the adverse rate with no normal rollback. The exploit requires a configured route and weak market protection, and loss magnitude needs a fork; calibrated severity: Medium.

Recommendation

Bind settlement to an independently calculated TWAP/oracle or signed quote, enforce maximum deviation and price-impact limits, and use per-hop limits where multi-hop routes are supported. Do not lower a pending floor without restarting the challenge or obtaining an independent approval.

Status / Client response: Acknowledged as an economic/MEV concern rather than a fund-safety break. Recommended deployment controls: execute configure/activate settlement in the same block, anchor minAmountOut to a TWAP, and use automated keepers to compress the challenge window.

core/AuctionEngine.sol:736-768

```
if (minAmountOut == 0) revert ZeroMinOut();
if (deadline <= block.timestamp) revert SettlementDeadlinePassed();
ISettlementAdapter(adapter).validateRoute(
    auction.auctionAsset, auction.poolQuoteAsset, data
);
$.settlementConfigs[launchId] = SettlementConfig({
    routeHash: keccak256(abi.encode(adapter, data)),
    minAmountOut: minAmountOut,
    deadline: deadline
});
```

core/AuctionEngine.sol:811-832

```
uint256 balanceBefore = _assetBalance(quoteAsset);
// ... adapter exactInput(..., settlement.minAmountOut, ...)
quoteAmount = _assetBalance(quoteAsset) - balanceBefore;
if (quoteAmount < settlement.minAmountOut) revert SettlementFloorUnreachable();
```

dex/MultiHopSettlementAdapter.sol:252-264

```
// Slippage is enforced once, on the realised end-to-end output.
amountOut = amount;
if (amountOut < minAmountOut) revert SlippageExceeded(amountOut, minAmountOut);
```

MEDIUM

Won't fix

F-005 — Factory and engine maintain unsynchronized settlement-adapter allowlists

Category: Configuration / Availability • Source: core/LaunchFactory.sol

Description

The factory validates and stores a settlement route independently from the engine's allowedSettlementAdapters registry. Launch creation checks the factory route but does not atomically check the engine allowlist, while later configuration and activation require the second registry. createLaunch invokes the factory route's validateRoute, but the engine's configureSettlement separately checks allowedSettlementAdapters[adapter]. The two registries can therefore disagree after funding has begun.

Impact

An operator can register a valid route in the factory, create a launch, and accept contributions while forgetting to allow the same adapter in the engine. Settlement then fails with AdapterNotAllowed and participants wait for the timeout/cancellation path. This is a launch-wide availability and refund-delay issue caused by configuration sequencing. Calibrated severity: Medium.

Recommendation

Use one authoritative route registry, or have launch creation atomically verify and snapshot both the route and engine authorization. Validate route health and adapter authorization before accepting contributions and provide a bounded operator correction or cancellation workflow.

Status / Client response: Acknowledged. No code change is planned. The team accepts the availability/refund-delay risk inherent in the current two-registry configuration sequencing and will follow the documented route-registration procedure operationally.

core/LaunchFactory.sol:215-221

```
SettlementRoute storage r =
$.settlementRoutes[_routeKey(config.auctionAsset, config.poolQuoteAsset)];
require(r.enabled, "Settlement route not registered");
ISettlementAdapter(r.adapter).validateRoute(
config.auctionAsset, config.poolQuoteAsset, r.data
);
```

core/AuctionEngine.sol:753-762

```
(address adapter, bytes memory data) = _settlementRoute(auction);
if (!$.allowedSettlementAdapters[adapter]) revert AdapterNotAllowed();
if (minAmountOut == 0) revert ZeroMinOut();
// ...
ISettlementAdapter(adapter).validateRoute(
auction.auctionAsset, auction.poolQuoteAsset, data
);
```

F-008 — V3 settlement adapters can strand unconsumed input

Category: Accounting / Fund Loss • Source: `dex/V3PoolSwapAdapter.sol`

Description

The V3 adapter transfers or approves the full `amountIn`, supplies a caller-configured `sqrtPriceLimitX96`, and trusts the router's output. `exactInput` passes both `amountIn` and `sqrtPriceLimitX96` to the router, then only clears the approval; there is no before/after balance assertion for `inputAsset` and no residual transfer to the engine. The multi-hop V3 helper also sets `amountOutMinimum` to zero for the leg and measures only the received output. The V3 leg in the multi-hop adapter has the same omission.

Impact

If the configured V3 router stops at a nonzero price limit after consuming only part of the input, the remaining protocol input stays in the adapter with no normal recovery path while the engine can accept the realized output above its minimum. The route requires a V3 deployment and price-limit behavior that allow partial consumption; calibrated severity: Medium, fork confirmation still required.

Recommendation

Assert the input balance delta equals the requested amount and refund any verified residual to the original caller/engine, or bind the router/factory to reviewed deployments whose exact-input behavior proves full consumption. Test the deployed router with nonzero price limits on a fork.

Status / Client response: Recommended if V3 routes are deployed: add an input-balance-delta assertion and residual refund, or pin the router/factory to reviewed deployments whose exact-input behavior proves full consumption.

`dex/V3PoolSwapAdapter.sol:62-85`

```
(uint24 fee, uint160 sqrtPriceLimitX96) = abi.decode(data, (uint24, uint160));
// ... transfer/approval of amountIn ...
amountOut = router.exactInputSingle{value: msg.value}(
    ISwapRouter.ExactInputSingleParams({
        tokenIn: tokenIn, tokenOut: tokenOut, fee: fee,
        recipient: recipient, amountIn: amountIn,
        amountOutMinimum: minAmountOut,
        sqrtPriceLimitX96: sqrtPriceLimitX96
    })
);
if (inputAsset != address(0)) IERC20(inputAsset).forceApprove(address(router), 0);
```

`dex/MultiHopSettlementAdapter.sol:427-440`

```
uint256 balanceBefore = IERC20(receiveToken).balanceOf(address(this));
IERC20(payToken).forceApprove(address(v3Router), amount);
v3Router.exactInput(
    ISwapRouter.ExactInputParams({
        path: path, recipient: address(this),
        amountIn: amount, amountOutMinimum: 0
    })
);
IERC20(payToken).forceApprove(address(v3Router), 0);
received = IERC20(receiveToken).balanceOf(address(this)) - balanceBefore;
```

F-009 — Native/USD TWAP accepts unbounded pool and fallback prices

Category: Pricing / Oracle • Source: libraries/TWAPOracle.sol

Description

The TWAP helper converts observations into an arithmetic tick without checking pool liquidity, observation cardinality, or deviation from an independent reference. TWAPOracle.consult only calls observe and computes the mean tick. setNativeUsdPool validates only decimal nonzeroness, while getNativeUsdPrice accepts any nonzero TWAP or the admin fallback. Native target initialization uses the result directly in the target-raise and individual-cap calculation.

Impact

A shallow configured pool can be held at a wrong price for the observation window, producing an incorrect native-denominated target raise and participation cap. An enabled incorrect fallback has the same effect without market manipulation. The source does fail closed when no usable price or fallback is available, and deployment pool depth/fallback state are not present; calibrated severity: Medium and conditional.

Recommendation

Require minimum liquidity and observation-cardinality policies, compare the TWAP with an independent oracle or bounded reference, and reject excessive deviation or implausible prices. Keep fallback disabled in production and verify pool address, token order, decimals, depth, and fallback state during deployment.

Status / Client response: Deployment-conditional. Keep fallback disabled in production and verify pool address, token order, decimals, and depth at deploy; residual risk is configuration-dependent.

libraries/TWAPOracle.sol:15-34

```
(int56[] memory tickCumulatives, ) = IPancakeV3Pool(pool).observe(secondsAgos);
int56 tickCumulativesDelta = tickCumulatives[1] - tickCumulatives[0];
arithmeticMeanTick = int24(tickCumulativesDelta / int56(uint56(twapInterval)));
```

core/AuctionEngine.sol:267-279

```
if (nativeDecimals == 0 || usdDecimals == 0) revert ZeroDecimals();
AuctionEngineStorage storage $ = _getStorage();
$.nativeUsdPool = pool;
$.nativeIsToken0 = nativeIsToken0;
$.nativeDecimals = nativeDecimals;
$.usdDecimals = usdDecimals;
```

core/AuctionEngine.sol:315-339

```
if (pool == address(0)) {
if (!$.fallbackAllowed || fallbackPrice == 0) revert NativePriceUnavailable();
return fallbackPrice;
}
// ... TWAP call ...
if (price == 0) {
if (!$.fallbackAllowed || fallbackPrice == 0) revert NativePriceUnavailable();
price = fallbackPrice;
}
```

core/AuctionEngine.sol:372-377

```
if (config.targetRaiseQuote == 0) {
if (config.auctionAsset != address(0)) revert ExplicitTargetRequired();
nativeUsdPrice = getNativeUsdPrice();
```

```
uint256 fdvNative = Math.mulDiv(targetMarketCap, 1e18, nativeUsdPrice);
targetRaise = Math.mulDiv(fdvNative, fundraisingBps, 10_000);
individualCap = Math.mulDiv(fdvNative, MAX_PARTICIPATION_BPS, 10_000);
}
```

7 Informational Findings

INFORMATIONAL

Planned fix

F-006 — Privileged module setters do not validate code or dependency bindings

Category: Deployment / Configuration • Source: core/LaunchFactory.sol

Description

Admin setters store arbitrary nonzero addresses and rely on void interface calls to detect compatibility. setModules writes four module addresses and grants roles without code or interface checks. Route registration accepts any target whose validateRoute call does not revert, and the optional Infinity/V4 lanes perform only partial dependency validation. The setters do not require contract code, probe interfaces, or bind the executor, pool manager, state lens, helper, and position-manager relationships into one validated dependency graph.

Impact

A deployment or module-rotation mistake can silently accept an EOA or mismatched dependency, strand a launch, or send launch inventory to an unintended address. The trigger requires the default-admin/deployment operator and no unauthenticated caller can bypass that authority. This is an informational deployment-hardening issue, not a high-severity privilege escalation.

Recommendation

Require nonzero contract code and interface probes for every module/dependency. Validate and bind executor, pool manager, state lens, factory, hook, Permit2, and position-manager relationships at construction or setter time. Verify the complete graph and role grants atomically before accepting funds.

Status / Client response: Acknowledged. After official deployment, the sensitive admin/upgrader/settlement roles will be moved to a multisig scheme with role separation and timelock, reducing the current single-key trust boundary.

core/LaunchFactory.sol:154-182

```
$.auctionEngine = _auctionEngine;
$.feeDistributor = _feeDistributor;
$.stakingVault = _stakingVault;
$.liquidityManager = _liquidityManager;
_grantRole(MODULE_ROLE, _auctionEngine);
_grantRole(MODULE_ROLE, _liquidityManager);
```

core/LaunchFactory.sol:490-507

```
if (adapter == address(0)) {
  delete $.settlementRoutes[key];
  return;
}
ISettlementAdapter(adapter).validateRoute(inputAsset, outputAsset, data);
$.settlementRoutes[key] = SettlementRoute({adapter: adapter, enabled: true, data: data});
```

dex/LiquidityManager.sol:247-260

```
require(_permit2 != address(0), "Invalid Permit2");
$.clPoolManager = _clPoolManager;
$.clPositionManager = _clPositionManager;
$.permit2 = _permit2;
$.jumpInfinityHook = _infinityHook;
$.infinityHelper = _infinityHelper;
```

dex/MultiHopSettlementAdapter.sol:161-167

```
if ((swapExecutor_ == address(0)) != (stateView_ == address(0))) {
    revert V4LaneNotConfigured();
}
v4SwapExecutor = ISwapExecutor(swapExecutor_);
v4StateView = IStateView(stateView_);
```

8 Rejected Candidate

The following candidate(s) were raised during the review but are **rejected as security findings**. They are documented here for traceability and, where applicable, as integration/runbook notes.

REJECTED

Rejected

F-002 — Per-address caps do not enforce identity-level concentration limits

Category: Access Control / Concentration • **Source:** core/AuctionEngine.sol

Description

Contribution reachability and the individual cap are keyed solely by msg.sender. The optional contribution gate is a single mutable global address, can be disabled with the zero address, and is not snapshotted into each launch's terms. The engine records each participant under contributions[launchId][msg.sender] and increments totalReachableQuote using the minimum of that address's nominal amount and individualCapQuote. Splitting funds across many EOAs or contracts therefore multiplies the reachable amount.

Impact

When a launch promises identity-level, KYC, or concentration protection, one participant can fund multiple wallets and reach the aggregate target through the sum of their per-address caps. A global gate change also changes the eligibility policy of existing launches. The issue does not bypass a plain per-wallet cap and does not directly cause asset loss, so the severity is Medium and policy-conditional.

Recommendation

State explicitly whether the cap is per wallet or per identity. For identity-capped launches, snapshot a per-launch authorization/identity commitment and enforce aggregate identity limits with auditable attestations or on-chain identity/nullifier accounting. Treat gate changes as controlled per-launch changes rather than a global switch.

Status / Client response: Not a security finding. The product explicitly commits to a per-wallet contribution cap, not identity-level (KYC/Sybil) protection, so the observed per-address behavior matches the documented policy. No code change is planned; the per-wallet semantics should be stated in the product documentation.

core/AuctionEngine.sol:510-513

```
address gate = $.contributionGate;
if (gate != address(0) && !IContributionGate(gate).isAuthorized(launchId, msg.sender)) {
    revert ContributionNotAuthorized();
}
```

core/AuctionEngine.sol:528-539

```
Contribution storage user = $.contributions[launchId][msg.sender];
uint256 oldReachable = Math.min(user.amount, auction.individualCapQuote);
// ... participant registration ...
user.amount += amount;
auction.totalContributedQuote += amount;
uint256 newReachable = Math.min(user.amount, auction.individualCapQuote);
auction.totalReachableQuote += newReachable - oldReachable;
```

core/AuctionEngine.sol:931-938

```
function setContributionGate(address gate) external onlyRole(DEFAULT_ADMIN_ROLE) {
  _getStorage().contributionGate = gate;
  emit ContributionGateUpdated(gate);
}
```

REJECTED

Rejected

F-007 — NativeToQuoteEntry is not wired to the current settlement interface

Category: Integration Note • **Source:** dex/NativeToQuoteEntry.sol

Description

The source mismatch is real: the current engine and factory use `ISettlementAdapter`, repository search found no production caller of `NativeToQuoteEntry.swapNativeToQuote`, and the generic production adapters already accept native input. Registering the legacy entry would fail route validation.

Impact

No current launch path silently selects the legacy entry, drains custody, or blocks a required configured route. The issue is therefore not rated as a security finding.

Recommendation

Keep the legacy entry as an integration/runbook note rather than a security finding. If the entry is ever re-enabled in production, re-run the settlement route validation and dependency checks.

Status / Client response: Keep as an integration/runbook note rather than a security finding.

dex/NativeToQuoteEntry.sol

```
contract NativeToQuoteEntry is INativeToQuoteEntry, Initializable {
  function swapNativeToQuote(...) external payable { // no production caller found
    ...
  }
}
```

9 Limitations & Disclaimer

This report reflects the state of the reviewed source and configuration at the time of the engagement, 28 September 2026. Security reviews are time-boxed and cannot guarantee the absence of undiscovered vulnerabilities. Any change to the deployed code, dependencies, deployment configuration, or integration assumptions may invalidate the conclusions herein. Findings that depend on external DEX/oracle/manager/token implementations, deployment configuration, or a fork environment remain conditional until that evidence is produced. OpenDrift accepts no liability for losses arising from reliance on this report.