CONFIDENTIAL VERIFICATION REPORT

MIDWEST SMART CONTRACTS & AUDIT SOLUTIONS

BASEDBID

Consolidated Contract Verification Report

Four implementation families | Four networks | Fourteen deployments

PREPARED FOR	**BASEDBID**
PREPARED BY	Midwest Smart Contracts & Audit Solutions
REPORT	MSC-BB-CVR-2026-001 | Version 1.0
ISSUE DATE	4 September 2026

STATUS Verification complete for all fourteen listed deployments. Deep security review was not included in this engagement.

1. Executive summary

BASEDBID requested a consolidated verification of its published lock-contract deployments, with a deep review of every contract expressly deferred. The engagement covers fourteen deployments across BNB Smart Chain, Robinhood Chain, Base, and Ethereum.

For each listed deployment, the work recorded explorer source-verification metadata and compiler configuration, decoded the constructor-configured position manager, retrieved deployed runtime bytecode, and compared equivalent implementations across networks. Immutable position-manager values were normalized only when necessary to separate expected constructor configuration from executable-code differences.

14	4	4	1
DEPLOYMENTS	NETWORKS	CONTRACT FAMILIES	DOCUMENTED VARIANCE

VERIFICATION CONCLUSION All fourteen listed deployments were recorded and compared. Equivalent deployments were runtime-identical directly or after normalization of the documented immutable manager address, except for the localized BNB Smart Chain BasedV3UniswapLock variance documented in Section 6.

IMPORTANT This conclusion establishes deployment consistency under the stated methodology. It does not establish that the underlying source is secure, functionally correct, or suitable for every token, pool, hook, or user condition.

Implementation families

- LpLockCore - shared permanent-custody and fee-receiver controls.
- BasedV3UniswapLock / V3LpLockBase - Uniswap V3-style position-manager integration.
- BasedV3PCSLock / V3LpLockBase - PancakeSwap V3-style position-manager integration.
- BasedV4Lock - Uniswap V4 position-manager integration.
- BasedInfinityLock - PancakeSwap Infinity CL position-manager integration.

2. Verification scope

Included objectives

- Independently confirm the listed contract addresses and published source-verification status.

- Record published compiler, optimizer, and EVM-version metadata where available.
- Decode and record immutable constructor-configured position-manager addresses.
- Retrieve and compare deployed runtime bytecode for equivalent implementations.
- Normalize only documented immutable position-manager values when cross-chain configurations differ.
- Identify and document any remaining runtime variance.
- Perform a limited structural screen sufficient to categorize implementation families and interpret comparison results.
- Document the precise boundaries and limitations of the verification engagement.

Meaning of the results

Explorer exact match: The applicable explorer reports that its published source compiles to the deployed bytecode under the displayed settings. This is not a security rating.

Runtime identical: The retrieved deployed runtimes were byte-for-byte equal.

Identical after immutable normalization: The runtimes became equal after substituting only the documented constructor-set position-manager address. No other executable bytes were disregarded.

Deployment consistency: This does not prove correctness, protocol compatibility, economic soundness, or freedom from vulnerabilities.

3. Exclusions and intentional deferrals

SCOPE BOUNDARY The activities below were intentionally excluded by the agreed verification-first scope. Their absence is not an omission or incomplete performance; each requires a separately authorized security-review engagement.

- Comprehensive or line-by-line security auditing of every execution path.
- Vulnerability discovery intended to provide full coverage, a security grade, or a no-findings conclusion.
- Exploit development, adversarial attack simulation, penetration testing, or transaction replay.
- Formal verification or mathematical proof of correctness.
- Automated static-analysis coverage, exhaustive fuzzing, invariant testing, unit or integration testing, fork testing, gas testing, or economic simulation.
- Deep review of third-party position managers, pool managers, token implementations, hooks, or imported OpenZeppelin code.
- Confirmation that a configured third-party manager is the correct or safest protocol deployment beyond recording its on-chain address.
- Compatibility testing for individual pools, tokens, hooks, NFTs, or user-supplied position data.
- Validation of live balances, existing locks, historical transactions, fee accounting, liquidity positions, or future runtime behavior.
- Deployment-factory, salt, creation-transaction, repository-history, commit-history, or developer-build provenance analysis beyond the recorded explorer and runtime evidence.
- Review of the BASEDBID launchpad, front end, back end, treasury, governance, operational keys, custody procedures, or unrelated contracts.
- Review of any contract address not expressly listed in the deployment register.
- Legal, regulatory, tokenomic, market, financial, or business-model review.
- Guarantees concerning future chain upgrades, contract replacements, source changes, or third-party protocol changes.

4. Methodology

- Explorer inspection of published verification status and compiler metadata.
- Constructor-argument decoding and recording of the configured position manager.
- Direct retrieval of deployed runtime bytecode from accessible on-chain records.
- Byte-for-byte comparison for deployments with identical immutable configuration.
- Normalized comparison for equivalent deployments whose documented immutable manager addresses differ.
- Runtime-length and SHA-256 evidence recording for material comparisons.
- Limited manual source and disassembly inspection solely to identify implementation families, immutable locations, and the context of observed deployment differences.

5. Observed architecture

The following characteristics were visible during the limited structural screen used to support deployment comparison. They are contextual descriptions, not the result of a comprehensive security audit.

Permanent custody

lock(tokenId, feeReceiver) records a nonzero fee receiver and transfers the position NFT into the locker. The published implementation exposes no method that transfers a locked NFT out of the contract.

Fee rights

Anyone may trigger fee collection, while proceeds are directed to the recorded fee receiver. The current receiver may transfer that right to another nonzero address. No owner override or recovery authority was visible.

Liquidity additions

Anyone may add liquidity using their own funds. V3 lockers temporarily custody funding, approve the manager, add liquidity, revoke approvals, and refund remaining token balances. V4 and Infinity use batched manager actions.

Additional visible characteristics

- No owner or administrator privilege was visible in the published implementation.
- No proxy, implementation slot, upgrade function, or delegatecall-based upgrade path was identified during the structural screen.
- No unlock or NFT rescue function was visible; permanent custody is implemented at the contract layer.
- A zero address cannot be recorded as the fee receiver.
- Only the current fee receiver may change the recorded receiver.
- Fee collection is permissionless while the payout destination remains constrained.
- Reentrancy protection is applied to locking, fee collection, and liquidity increases.
- V3 token approvals are set for the requested amount and reset after the external manager call.

6. Deployment verification register

All explorer exact-match entries were observed with Solidity 0.8.36+commit.8a079791, Prague EVM settings, optimization enabled, and 200 runs. Full deployment and position-manager addresses are recorded below.

BNB Smart Chain

CONTRACT / RESULT	ON-CHAIN CONFIGURATION
BasedV3UniswapLock Explorer exact match; localized runtime variance versus Base documented in Section 6.	DEPLOYMENT `0xb8f4eDA82c2c9D514FFF646e88e7297480C5dEaD` MANAGER `0x7b8A01B39D58278b5DE7e48c8449c9f4F5170613`
BasedV3PCSLock Explorer exact match; runtime byte-for-byte identical to Base.	DEPLOYMENT `0x82306A00f4Dd54482F708066142492f96DeBdEaD` MANAGER `0x46A15B0b27311cedF172AB29E4f4766fbE7F4364`
BasedV4Lock Explorer exact match; runtime matches Base after immutable normalization.	DEPLOYMENT `0xD77cBAcE1E6dEE8230Ac38294CEf13fA6871dEaD` MANAGER `0x7A4a5c919aE2541AeD11041A1AEeE68f1287f95b`
BasedInfinityLock Explorer exact match; runtime byte-for-byte identical to Base.	DEPLOYMENT `0xDdfa9b6EB5b6f5cFF64289c9391820889BabdEaD` MANAGER `0x55f4c8abA71A1e923edC303eb4fEfF14608cC226`

Robinhood Chain

CONTRACT / RESULT	ON-CHAIN CONFIGURATION
BasedV3UniswapLock Verified source; runtime matches Base after immutable normalization.	DEPLOYMENT `0x635f5b6A566573bef2E4E631DFf004baf3fEdEaD` MANAGER `0x73991a25C818Bf1f1128dEAaB1492D45638DE0D3`
BasedV3PCSLock Verified source; runtime byte-for-byte identical to Base.	DEPLOYMENT `0x82306A00f4Dd54482F708066142492f96DeBdEaD` MANAGER `0x46A15B0b27311cedF172AB29E4f4766fbE7F4364`
BasedV4Lock Verified source; runtime matches Base after immutable normalization.	DEPLOYMENT `0xF49Ddb31213C75C41776EdB41E06B821c4DadEaD` MANAGER `0x58daec3116aae6D93017bAAea7749052E8a04fA7`

6. Deployment verification register - continued

Base

CONTRACT / RESULT	ON-CHAIN CONFIGURATION
BasedV3UniswapLock Explorer exact match; Base reference implementation.	DEPLOYMENT `0xE91FC414B872859b4d3a37B3309EEdd44820dEaD` MANAGER `0x03a520b32C04BF3bEEf7BEb72E919cf822Ed34f1`
BasedV3PCSLock Explorer exact match; Base reference implementation.	DEPLOYMENT `0x82306A00f4Dd54482F708066142492f96DeBdEaD` MANAGER `0x46A15B0b27311cedF172AB29E4f4766fbE7F4364`
BasedV4Lock Explorer exact match; Base reference implementation.	DEPLOYMENT `0xF537120Ec2fF398AD11F07DA7421C7B043A0dEaD` MANAGER `0x7C5f5A4bBd8fD63184577525326123B519429bDc`
BasedInfinityLock Explorer exact match; Base reference implementation.	DEPLOYMENT `0xDdfa9b6EB5b6f5cFF64289c9391820889BabdEaD` MANAGER `0x55f4c8abA71A1e923edC303eb4fEfF14608cC226`

Ethereum

CONTRACT / RESULT	ON-CHAIN CONFIGURATION
BasedV3UniswapLock Explorer exact match.	DEPLOYMENT `0x08740c80287603A8071287AD8d259e70e1C3dEaD` MANAGER `0xC36442b4a4522E871399CD717aBDD847Ab11FE88`
BasedV3PCSLock Explorer exact match.	DEPLOYMENT `0x82306A00f4Dd54482F708066142492f96DeBdEaD` MANAGER `0x46A15B0b27311cedF172AB29E4f4766fbE7F4364`
BasedV4Lock Explorer exact match.	DEPLOYMENT `0x92746C1fc7eD907Cb678D09425d67062054BdEaD` MANAGER `0xbD216513d74C8cf14cf4747E6AaA6420FF64ee9e`

Comparison evidence

Robinhood / Base V3 Uniswap | Immutable manager normalized | 6,216 hexadecimal characters
`SHA-256 2ea26772cee2d7dc07bd69de4b6160201be93f527ec5e88d210f4ce1bdbe29c3`

Robinhood / Base V3 PCS | Direct byte comparison | 6,208 hexadecimal characters
`SHA-256 8407df58c3d053f0e33e7c62b56ff229acb977e0e1b365f13adc605f20ff0cc0`

Robinhood / Base V4 | Immutable manager normalized | 7,002 hexadecimal characters
`SHA-256 6e7534c5a8101a495f6bbb871fa2adc487a782576cc38d5ee630186dbafdcd70`

BNB / Base Infinity | Direct byte comparison | 7,046 hexadecimal characters
`SHA-256 9900d07b30fb2e5e3319425e4131678e9b3511cdaf09fd67ab4fc5568f3df16d`

DOCUMENTED VARIANCE After immutable normalization, the BNB Smart Chain BasedV3UniswapLock contains one localized three-byte sequence (0x526024) in ZeroAddress() revert-data construction. Removing that sequence produces an otherwise exact runtime match to Base. The variance is recorded without making a broader behavioral-equivalence or security conclusion.

The repeated addresses 0x82306A00f4Dd54482F708066142492f96DeBdEaD and 0xDdfa9b6EB5b6f5cFF64289c9391820889BabdEaD occur on multiple networks. Each network deployment was treated as a separate on-chain record.

7. Observations reserved for future review

NOT SECURITY FINDINGS These items were noticed during the limited structural screen. They were not required for deployment verification, have not received comprehensive validation, and are not assigned security severities in this report.

BBL-01 Hook requirements may affect fee-collection compatibility

Affected: BasedV4Lock and BasedInfinityLock

The fee-collection batches encode empty hook data for the zero-liquidity decrease used to realize accrued fees. Positions whose hooks require nonempty data may be incompatible. A separately scoped review should validate supported pool and hook conditions before assigning impact or severity.

BBL-02 Whole-balance refund and manager-balance assumptions merit validation

Affected: V3LpLockBase; related V4 and Infinity behavior

The V3 refund path transfers the locker's remaining balance of each pool token to the liquidity contributor. V4 and Infinity rely on external manager balance-settlement behavior. A deeper review should validate unsolicited-balance and accounting assumptions across the configured managers.

BBL-03 Fee-receiver handoff is one-step and lacks recovery

Affected: All implementations

The recorded fee receiver may transfer that role immediately to another nonzero address. No two-step acceptance or administrator recovery path was visible. Operational procedures should require destination confirmation; any design recommendation belongs to a separately scoped review.

8. Limitations and use of report

This report reflects only the fourteen listed addresses, published explorer metadata, and accessible on-chain records observed as of the review date. It verifies specified deployment facts and cross-deployment runtime consistency under the stated methodology.

It does not certify source-code security, test functional correctness, determine whether all intended contracts were disclosed, or predict behavior under every token, pool, hook, protocol, chain, or user condition. Third-party protocol behavior, chain changes, token behavior, hooks, compromised keys, deployment mistakes, contract replacements, and undisclosed components can materially affect outcomes.

RESTRICTION This report must not be described or marketed as a comprehensive security audit, formal verification, warranty, guarantee, insurance policy, endorsement, or assurance of financial performance. A comprehensive security conclusion requires a separately defined and authorized deep-review engagement.

Issuance record

REPORT IDENTIFIER	**MSC-BB-CVR-2026-001**
VERSION	1.0
ISSUED	4 September 2026
PREPARED BY	Midwest Smart Contracts & Audit Solutions

END OF REPORT