VORTEX PROTOCOL / ENGINEERING

REPORT 01 — V2 FINAL

Smart contract security *review.*

Vortex Wallet Register V2 Final

A source-based assessment of the deployed contract, its economic controls, migration boundaries and operational dependencies.

NETWORK	REVIEW DATE	EVIDENCE BLOCK
BNB Smart Chain · 56	08 September 2026	120,676,009

TECHNICAL REVIEW · REV 1.0

SOURCE / DEPLOYMENT / TESTS

01 Executive summary

The reviewed build matches the deployed runtime after masking compiler-declared immutable locations. Migration is sealed.

This revision combines manual source inspection, automated contract tests and a public BSC RPC snapshot. It documents verified behavior and material trust assumptions; it does not provide an independent audit-firm attestation.

99	77	23,935
V2 Final tests passed	Selected inherited checks passed	Deployed runtime bytes

Controls present in the source

Reentrancy guards on token-moving entry points, SafeERC20 transfers, caller-supplied slippage bounds, one-way migration sealing, pool-reserve separation and ranged ROI claims.

Material dependencies

Privileged configuration, external reserve funding, price-feed integrity, correct migration inputs and the gas cost of large referral structures. These remain relevant when ordinary functional tests pass.

Assessment position

The executed tests support the covered migration, referral, reward and withdrawal behaviors. Findings below are review observations with explicit conditions, not claims of demonstrated public exploits.

No aggregate solvency calculation, independent token/pair audit, formal verification or adversarial mainnet-fork assessment was completed for this report.

02 Deployment & build

Deployment fields were supplied from BscScan and cross-checked against the transaction receipt and block returned by BSC RPC. Compiler details below come from the matching build.

Deployment identity and reproducible build settings

Contract	**VortexWalletRegisterV2Final**
Address	`0x715EeB0DA4586a7c803301d8b304AfA234d9f803`
Network	BNB Smart Chain mainnet · Chain ID 56
Deployment transaction	`0xfae6dfdee7f3671f3f509a4be2501f6efe4cf0c8988af76f13cd21f51cfd8217`
Receipt	Success · Contract creation · Block 118,245,276
Deployment time	26 August 2026 · 19:10:05 UTC
Deployer	`0x6F5A0083CaE993B21Ad77034dCb00ea484A3c6e5`
Gas used / effective price	6,168,955 gas / 0.05 Gwei
Transaction fee	0.00030844775 BNB · gas used × effective gas price
Value sent	0 BNB · supplied BscScan transaction detail
Compiler	`v0.8.35+commit.47b9dedd`
Optimization	Enabled · 1 run · `viaIR: true`
EVM target / source pragma	`paris` / `^0.8.20`
License / dependencies	MIT / OpenZeppelin IERC20, SafeERC20, ReentrancyGuard, Ownable2Step

> **Build-to-chain comparison: match**
>
> The build input contains the exact reviewed source. Deployed runtime and compiled output are equal after replacing compiler-declared immutable byte ranges with zero. This comparison includes the remaining runtime and metadata. Immutable getters were recorded separately; creation bytecode was not independently reconstructed.

The runtime occupies 23,935 of 24,576 bytes, leaving 641 bytes below the size limit used by the project. Relative ages, changing confirmation counts and fiat fee estimates are deliberately omitted from this fixed record.

03 Protocol mechanics

Deposits create USD-denominated positions. Payouts are funded from a separate VTR reserve.

DEPOSIT PATH

Registered user
Minimum USD value: 100

↓

70% VTR + 30% LVTR
USD value split at priceUsd()

↓

Collector wallet
Both tokens leave the user

The contract records the position and referral credits. Deposit transfers do not automatically replenish the contract's VTR payout reserve.

WITHDRAWAL PATH

USD balance
Minimum withdrawal: 10

↓

VTR reserve
Gross amount = USD ÷ price

↓

95% net to user
3% brotherhood · 2% pool

Integer rounding applies. The 2% pool portion remains in the contract and is reserved for pool claims.

Parameters defined by the reviewed source; mutable defaults are not a snapshot of live configuration

MECHANISM	SOURCE BEHAVIOR
ROI tiers	Below $1,000: 0.60–0.80% daily; $1,000–4,999.99: 0.60–1.00%; $5,000+: 0.60–1.20%. Rates are a deterministic hash of UTC day and tier, not unpredictable randomness.
Position cap	ROI credits stop at 250% of position USD value. The position then becomes inactive when settled. Principal is not a separately withdrawable balance.
Direct referral bonus	5% base rate; 7%, 8% or 10% can unlock during a 72-hour accelerator window. The manager may restart that window.
Matching / leadership	Matching examines up to 10 levels, unlocked by active directs. Leadership traverses up to 50 ancestors and uses effective rank plus big-leg exclusion.
Rank progression	Earned ranks only increase. Effective rank is max(earned, manual). Manual grants can be reduced or revoked without lowering the earned rank.
Pool accounting	Seven-day cycles; per-weight accumulator, user debt and credit. Pool weights in rank configuration freeze when migration is sealed.
Oracle	Eight-observation ring per feed; at most one sample per UTC day; minimum six-day elapsed window. Token/WBNB quotes are combined with WBNB/USDT.

What V2 Final adds

`setReferrer` moves a subtree and updates volumes; `importState` imports positions and monetary state; `setUserRank` grants manual rank; `resetAccelerator` reopens the referral window. The manager can also update rank configuration and fallback prices within their respective guards.

04 Control & permissions

Snapshot: block 120,676,009 · 08 September 2026, 11:11:06 UTC. All state calls in this record used that explicit block number.

Owner	`0x6F5A0083CaE993B21Ad77034dCb00ea484A3c6e5`
Pending owner	Zero address
Account manager	`0xEfbd812a7c654cb035871a2E829DdfFb5B4CE28B`
Collector (immutable)	`0xEfbd812a7c654cb035871a2E829DdfFb5B4CE28B`
Brotherhood	`0xaEfDd2FB1bB1f0848BeC19fD1ccA1344f028E90b`

Migration sealed	`true`
Price fallback enabled	`true`
Registered / imported	`3,743 / 3,569`

Payable reserve	`5,494,879.9168 VTR`
Pool reserved	`10,740.8138 VTR`
Total pool weight	`165`

Token figures use 18-decimal formatting, consistent with the contract's arithmetic. Reserves alone do not establish coverage of all liabilities.

Access rules in the reviewed contract

OPERATION	AUTHORITY	AFTER OWNER RENUNCIATION
Set account manager	Owner or manager	Manager can still transfer or retire its role.
Move referral subtree; grant rank; reset accelerator	Manager; after seal	Remain available to the manager.
Set rank configuration	Owner or manager	Manager retains control. Pool weight changes are blocked after seal.
Set fallback price	Owner or manager	Manager retains control while fallback is enabled.
Set feeds; disable fallback	Owner	Unavailable. Disable requires mature nonzero feeds.
Set brotherhood; matching levels	Owner	Unavailable.

OPERATION	AUTHORITY	AFTER OWNER RENUNCIATION
Import structure / state; seal	Owner; before seal	Unavailable; migration is already sealed in this snapshot.
Poke; refresh big leg; fund reserve	Anyone	Remain public.

A renounced owner would not remove manager authority. The current manager and immutable collector are the same address. Key custody, multisignature configuration and signer practices were not assessed.

05 Review observations

Priority describes review attention and potential consequence under the stated conditions. These entries distinguish intentional trust assumptions, operational limits and code-level concerns. None is presented as a reproduced unauthorized exploit.

V2F-01 **GOVERNANCE CONSIDERATION** Privileged control

Administrative configuration

`setFallbackPrice` accepts any uint256 price while fallback is enabled. If the TWAP path cannot supply a nonzero quote, that configured value prices deposits and withdrawals. A malicious or erroneous value can distort the exchange between tokens and credited USD; zero can make that fallback path revert. Mature TWAP quotes take precedence.

The manager may also alter rank thresholds and leadership basis points, assign manual ranks and move referral subtrees. Ownership renunciation would preserve these manager powers. The recorded permissions and fallback setting are listed in section 04.

> **Operational action.** Review custody and change procedures for both roles. Confirm feed readiness before deciding whether to permanently disable fallback; complete owner-only configuration before any renunciation.

Contract functions: `setAccountManager`, `setReferrer`, `setFallbackPrice`, `priceUsd`, `setRank`, `setUserRank`. Status: retained capability; no misuse demonstrated.

V2F-02 **LOW PRIORITY** Economic dependency

Recorded rewards require external reserve funding

`deposit` transfers both tokens directly to the collector. It records USD positions and bonuses without requiring equivalent VTR to be held by the register. ROI claims increase accounting balances; `withdraw` later checks the available reserve and reverts if it is insufficient.

The contract snapshot shows a positive reserve, but the contract does not expose an aggregate measure covering all present and future reward liabilities. A positive balance is therefore not a solvency proof.

> **Operational action.** Reconcile positions, reward balances, pending accruals, token prices and reserves in a separate liability model. Define how the collector replenishes VTR and how reserve shortfalls are handled.

Contract functions: `deposit`, `_claim`, `payableReserve`, `fundReserve`, `withdraw`. Status: funding model / trust assumption; coverage not calculated.

V2F-03 **MEDIUM PRIORITY** Migration integrity

Import correctness depends on the supplied snapshot

`importStructure` rejects self-referrals but does not perform a full cycle check. `importState` accepts supplied financial and volume values without enforcing all cross-field invariants. `sealMigration` checks the expected structure import count, not that every registered user received valid state.

Rank and pool accounting are intentionally omitted from `StateRow` under the assumption that those fields were zero in the source snapshot. Invalid volumes would be particularly sensitive because detachment performs unchecked subtraction under consistency assumptions. Sealing closes the import entry points but does not prove historic inputs were correct.

> **Verification action.** Preserve an acyclic source snapshot and a field-by-field reconciliation, including omitted state and reconstructed branch volume. The executed migration tests demonstrate correct sample imports; the complete production migration was not replayed here.

Contract functions: `importStructure`, `importState`, `sealMigration`, `_detachVolume`. Status: privileged input-validation boundary; migration sealed on-chain.

V2F-04 **MEDIUM PRIORITY** Oracle dependency

Pricing depends on pair integrity and observation history

Feed setup validates the token and quote-token addresses but does not authenticate the pair against a factory. The cumulative-price calculation relies on the configured pair and has no explicit minimum-liquidity guard or maximum observation age. An old observation can produce an averaging interval longer than six days.

Sampling calls external pairs during deposits, ROI claims and withdrawals. A reverting configured pair can therefore interrupt those operations. Disabling fallback is permanent and also freezes feed changes; renouncing ownership prevents later use of either owner-only control.

> **Verification action.** Check configured pair provenance, liquidity, cumulative observations and sampling health. This revision did not inspect pair storage or determine which price path served each live token quote.

Contract functions: `setFeed`, `_sampleAll`, `_sample`, `_cumulativeNow`, `_feedPrice`, `priceUsd`, `disablePriceFallback`. Status: oracle trust and availability assumptions.

V2F-05 **MEDIUM PRIORITY** Accounting boundary

Deep subtree moves can differ from fresh deposit propagation

Deposits propagate volume for at most 50 ancestors. Detachment clamps the removed amount to each recorded branch, whereas attachment credits the moved subtree's aggregate volume to each new ancestor within its walk. For sufficiently deep subtrees, the new ancestor can receive volume that individual fresh deposits at those depths would not have propagated to it.

The source documents this asymmetry. The new sponsor path is checked, but the entire moved subtree depth is not constrained. Ranks already earned are not reduced when volume leaves.

> **Operational action.** Model depth and volume changes before manager-driven moves. Treat the 50-hop boundary as an accounting rule; avoid assuming invariance with a rebuilt deposit history beyond that boundary.

Contract functions: `setReferrer`, `_detachVolume`, `_attachVolume`, `_bubbleUp`, `_refreshRank`. Status: documented source behavior; no production impact quantified.

V2F-06 **MEDIUM PRIORITY** Token assumptions

Nominal transfer amounts are trusted by the accounting

Deposit pricing scales token amounts by `1e18` and records the requested USD value after SafeERC20 transfers. It does not compare recipient balance changes against nominal amounts. Tokens with transfer fees, rebasing behavior or incompatible decimal conventions can therefore diverge from this accounting model. SafeERC20 handles call results; it does not validate economic transfer semantics.

> **Verification action.** Assess the immutable VTR and LVTR contracts and their configuration for compatible transfer behavior and units. Those token implementations were not independently reviewed in this report.

Contract functions: `constructor`, `deposit`, `payableReserve`, `fundReserve`, `withdraw`. Status: integration assumption; addresses recorded in section 07.

V2F-07 **OPERATIONAL PRIORITY** Gas & maintenance

Large accounts and wide trees need bounded operations

Full ROI claims iterate positions and accrued days; ranged claims provide an existing way to split that work. Big-leg rescans during a move share a 1,000-branch budget. When it is exceeded, the code pins big-leg volume to total direct volume and emits `BigLegStale`. `refreshBigLeg` is public but scans the full direct list in one call.

Stale big-leg data is designed to suppress new small-leg rank qualification at that moment, but should not be assumed harmless for every downstream use: leadership also reads `bigLegDirect`. Incorrect unlink hints may trigger an additional unbounded direct-list scan.

> **Operational action.** Use `claimRoiRange`, supply accurate old-index hints and monitor `BigLegStale`. Simulate unusually wide repairs; the tested shapes are examples, not a universal gas ceiling.

Contract functions: `_unlinkDirect`, `_rescanBigLeg`, `refreshBigLeg`, `_accrue`, `claimRoiRange`, `_payLeadership`. Status: tested limits and maintenance dependency.

06 Validation results

176 successful checks across the reported validation scenarios.

Tests ran in a controlled environment with simulated tokens and liquidity pairs, without submitting mainnet transactions. The inherited functional tests exercised V2 Final.

V2 Final: accelerator, import state, manual rank, reparenting, renounced ownership, simulation and gas (7 test groups)	99 PASSED
Inherited: deposit, ROI, withdraw, oracle, matching, leadership, pool, referral, ranks, limits and claim ranges (11 test groups)	77 PASSED

The reported totals exclude the historical V2 minimum-gas benchmark, which is not a correctness criterion for V2 Final.

Selected measurements from the executed tests

SCENARIO	GAS USED	INTERPRETATION
Constructed deep / wide move	12,065,736	71.9% of the suite's 16,777,216 comparison ceiling.
Ordinary subtree move	284,552	Baseline for the tested scenario.
Repair 1,001 direct branches	4,737,873	Public full-list big-leg refresh.
Correct / incorrect hint	131,897 / 2,563,302	1,001-entry direct list.
120 positions / 90 days	17,487,046 total	Successfully settled in three ranged transactions.

Coverage caveat: the inherited bytecode-size check evaluates the original V2 contract. Its passing status is not size evidence for Final. Final's 23,935-byte runtime was measured separately from the on-chain code and matching build.

No Slither, Mythril, formal proof or independent coverage analysis was run.

07 Scope & evidence

Reviewed source

VortexWalletRegisterV2Final · Solidity smart contract. The reviewed source is identified by the SHA-256 digest below. The deployment and contract code can be inspected on BscScan.

SOURCE SHA-256

`a2b218397eeca9cc31a1db141271704b41382c345946ef74de1a5e787cb55603`

Deployment snapshot

BSC RPC endpoint: bsc-dataseed1.binance.org. The block number was fetched immediately before a batch of calls pinned to that block. Data is a historical snapshot and does not update when this HTML is opened.

BLOCK 120,676,009 HASH

`0x1813e01f13bef9545fd7d96f709c0951b8270103533e9e2705ca0ee50976b180`

DEPLOYED RUNTIME KECCAK-256

`0x0bc2223b4ca9060572c90da39976396b19b303e4397f1d50d99496dcba76ac3c`

Token getters recorded at the evidence block

VTR	`0x620833798C1596C5C8201a0b2C2d36E992A17Efb`
LVTR	`0x2dCb495f5BDF5BFc06Aa866710d1c7d118b9631a`
WBNB	`0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c`
USDT	`0x55d398326f99059fF775485246999027B3197955`

Assessment boundary. Based on Solidity source review, build comparison, automated contract tests and public RPC evidence. This is not a signed third-party audit or certification. Token/pair internals, full production migration replay, frontend and backend security, key custody, aggregate solvency and economic sustainability remain outside the completed work. No numerical safety score is assigned. Additional defects may exist outside the executed checks.