JadongPay

$JPAY

Security Audit Report

VERSION 4.0-COMPLIANCE — JUNE 2026

JadongPay (JPAY) — Security Audit Report

Version 4.0-compliance | June 2026

> ***Internal security review.*** *This report documents the internal audit process applied to the JadongPay smart contract (v4.0-compliance). Security relies on internal review, automated testing, and AI-assisted static analysis. No external third-party audit has been commissioned. This document is provided for transparency and does not constitute a guarantee; no software is free of risk.*

Executive Summary

The JadongPay smart contract underwent four internal audit passes over its development cycle, during which 42 vulnerabilities of varying severity were identified and resolved. The contract reached its frozen compliance release (v4.0-compliance) after the removal of the jackpot module (for legal compliance) and the implementation of a real token burn.

The final v4.0 contract passes 97 automated tests (91 unit + 4 invariant + 2 integration), with clean static analysis from both Slither and Mythril. The runtime bytecode is 19,341 bytes, well under the 24,576-byte EVM limit.

Metric	Value
Contract version	v4.0-compliance
Solidity	0.8.26 (via-ir)
Total tests	97 (91 unit + 4 invariant + 2 integration)
Audit passes	4 internal
Vulnerabilities resolved	42
Slither	0 actionable High (1 documented false positive)
Mythril	0 issues detected in JadongPay
Runtime size	19,341 bytes (margin 5,235 under limit)

Scope

The audit covered the full JadongPay contract (`src/JadongPay.sol`), including: - ERC20 token logic (OpenZeppelin 5.x base) - Sell-fee mechanism (10%: 7% redistribution, 2% burn, 0.5% reserve, 0.5%

marketing) - USDT redistribution (Dividend-Per-Share accounting model) - Real token burn (reduces totalSupply) - Presale lifecycle (start, contribute, finalize, refund) - Liquidity creation and locking (trusted-locker pattern) - Vesting (presale and team schedules) - Ownership (Ownable2Step) and bounded admin parameters - Anti-manipulation (max-sell limit, anti-bot)

Test Coverage

The v4.0 test suite comprises 97 automated tests: - **91 unit tests** — individual function behaviour, access control, bounds, edge cases - **4 invariant tests** — properties that must always hold (e.g. tax conservation, reward-pool accounting), validated over thousands of randomized calls - **2 integration tests** — full lifecycle end-to-end (Scenario A: success path softcap → mint → LP → vesting → sell tax → rewards; Scenario B: failed presale → refund)

All 97 tests pass against the frozen v4.0 bytecode.

Findings Summary

Across four audit passes, 42 findings were identified and resolved. The table below reflects the findings applicable to the final v4.0 architecture. Findings that were specific to the jackpot/VRF module are no longer applicable, as that module was entirely removed in v4.0.

Severity	Identified	Resolved	Applicable to v4.0
Critical	6	6	All resolved
High	8	8	All resolved
Medium	7	7	All resolved
Low / Informational	5	5	All resolved/accepted

Critical Findings (All Resolved)

C-01 — Double Presale Finalization (RESOLVED)

`finalizePresale` could theoretically be triggered through inconsistent state. Resolved via a strict state machine and the one-shot `prepareFinalization` flag (`finalizationPrepared`), which reverts on a second call (`JPAY__AlreadyPrepared`).

C-02 — MAX_CONTRIBUTION Bypass via Partial Fill (RESOLVED)

A partial fill near the hardcap could allow a wallet to exceed its per-wallet maximum. Resolved by enforcing the per-wallet cap on cumulative contribution.

C-03 — LP Tokens Recoverable via recoverERC20 (RESOLVED)

The generic ERC20 recovery function could have extracted LP tokens. Resolved by explicitly blocking the LP token address in `recoverERC20`.

C-04 — emergencyWithdrawBNB Conflicts With Refunds (RESOLVED)

An emergency withdrawal path could have conflicted with pending refunds. Resolved by restricting BNB withdrawal so it cannot touch refundable contributions.

C-05 — LP Lock Destination Not Pre-Approved (RESOLVED)

LP tokens could have been sent to an arbitrary destination. Resolved with the trusted-locker pattern: the locker is set once during SETUP (`setTrustedLocker`, one-shot) and frozen thereafter; `transferLPForLocking` can only send to that pre-approved address.

C-06 — Surplus / Burn Accounting Integrity (RESOLVED)

Ensured that unsold presale supply is never minted (tracked via `unmintedSupply` + `SurplusNotMinted` event) and is never conflated with real burns. Real burns (sell-fee burn + LP-creation deficit) genuinely reduce `totalSupply()` via the standard ERC20 burn.

High Severity Findings (All Resolved)

H-01 — setMinSwapThreshold Without Bounds (RESOLVED)

Resolved by bounding the setter within `[10 ; 5,000] JPAY` (hard-coded floor/ceiling).

H-02 — setSwapSlippage(0) Disables All Swaps (RESOLVED)

Resolved by enforcing a minimum slippage bound so swaps cannot be silently disabled.

H-03 — setMaxSellBps Without Ceiling (RESOLVED)

Resolved by bounding the max-sell setter within `[10 ; 1,000] bps` (0.1% to 10%).

H-04 — minimumHoldingForRewards Without Ceiling (RESOLVED)

Resolved by bounding the eligibility-threshold setter within `[100 ; 10,000] JPAY`.

H-05 — Pause Blocks Vesting Claims (RESOLVED)

Resolved so that pause only affects external transfers; claims and refunds remain active while paused.

H-06 — LP Tokens Drainable via transferLPForLocking (RESOLVED)

Resolved by constraining `transferLPForLocking` to the pre-approved trusted locker only.

H-07 — Ownership Transfer Safety (RESOLVED)

Adopted `Ownable2Step`: ownership transfer is a two-step process (propose + accept), preventing accidental loss of ownership to a mistyped address. Relevant for the planned transfer to a Gnosis Safe multisig.

H-08 — Reward Accounting Precision (RESOLVED)

Resolved precision/rounding handling in the Dividend-Per-Share accumulator to avoid drift in `rewardPerShareStored`.

Medium Severity Findings (All Resolved)

M-01 — Constructor Duplicate Exclusions (RESOLVED)

Deduplicated fee/reward exclusions set in the constructor.

M-02 — Removing Reward Exclusion Doesn't Update Array (RESOLVED)

Resolved array bookkeeping when toggling reward exclusions.

M-03 — _eligibleSupply Underflow Risk (RESOLVED)

Added explicit guards (`s = s > x ? s - x : 0`) on both the AMM and excluded-wallet branches; confirmed underflow-free by invariant testing. The real burn correctly reduces eligible supply without underflow.

M-04 — contribute() Without nonReentrant (RESOLVED)

Added `nonReentrant` to the contribution path.

M-05 — transferLPForLocking Accepts Arbitrary ERC20 (RESOLVED)

Constrained to the LP token and trusted locker only.

M-06 — Tax Conservation Invariant (RESOLVED)

Added an explicit conservation check: `netAmount + burn + reserve + marketing + reward == value`, validated by invariant testing.

M-07 — Real Burn vs Cosmetic Burn (RESOLVED)

Replaced the earlier cosmetic burn (tokens parked in contract) with a real burn via `super._update(..., address(0), ...)`, which genuinely decrements `totalSupply()`. Applies to both the sell-fee burn and any LP-creation deficit.

Low / Informational Findings

L-01 — HARDCAP / PRESALE_RATE Rounding (Accepted)

Minor rounding in token math; impact negligible and accepted.

L-02 — VestingSchedule.startTime Semantics (Accepted)

Documented the meaning of the vesting start reference for clarity.

L-03 — No TOTAL_TAX_BPS Compile-Time Invariant (Resolved)

Added a runtime conservation check ensuring the tax components always sum to 1000 bps (10%).

L-04 — Constructor Atomicity for Wallet Exclusions (Resolved)

Ensured wallet exclusions are set atomically in the constructor.

L-05 — Solidity Version Pragma (Informational)

Pinned to Solidity 0.8.26; OpenZeppelin `^0.8.20` compatibility noted.

Static Analysis Results

Slither (Slither 0.11.5, via-ir, --filter-paths lib/)

61 results, all low/informational (pragma, solc-version, low-level-call in `_sendBNB`, naming conventions, unindexed events, cache-array-length, block.timestamp usage, divide-before-multiply). The only reentrancy notes are on `createLiquidityPool` and `_swapJpayForRewards`, both protected by `nonReentrant` and following the Checks-Effects-Interactions pattern — documented as a false positive. **0 actionable High findings.**

Mythril (v0.24.x — WSL, Python 3.12, execution-timeout 120, max-depth 8)

No issues detected within the JadongPay contract. One out-of-scope note relating to an OpenZeppelin/external base was reviewed and is not applicable to JadongPay's logic.

Conclusion

The JadongPay v4.0-compliance contract reflects four passes of internal review, with 42 findings resolved across all severity levels. The removal of the jackpot module eliminated an entire class of complexity (and its associated findings), and the move to a real burn plus `Ownable2Step` strengthened both transparency and operational safety. The contract is frozen (immutable) and verified on BscScan.

This is an internal review, not a substitute for an external audit. Participants should conduct their own research and understand that no smart contract is risk-free.

JadongPay Security Audit Report — v4.0-compliance — June 2026 Internal review. The authoritative source is the on-chain, verified contract.