

Security Audit Complete

OTCSwap V4 — All Critical Issues Resolved

We recently completed a focused security audit of the OTCSwap smart contract (the core escrow engine behind SwapNFT.xyz). Three critical vulnerabilities were identified in earlier versions. All three have been fully fixed in OTCSwap V4.

□ Outcome: The protocol now enforces strict deposit validation, uses safe pull-based ETH handling, and rejects fake NFT offers. Your assets are protected by these hardened mechanisms.



Audit tooling: Paradigm EVM Bench

(Tooling credit only — not an endorsement)

What Was Found & Fixed

1. ERC20 Deposit Validation

Fee-on-transfer tokens could cause under-collateralized orders and drain shared balances. Fixed: V4 now strictly verifies the exact amount received before accepting any deposit.

2. ETH Transfer Locking

ETH sent to non-payable contract addresses could become permanently stuck. Fixed: We moved to a pull-based withdrawal system so users can always claim their ETH safely.

3. Fake NFT Offers

Malicious makers could advertise non-existent NFTs and steal taker deposits. Fixed: Every NFT leg now requires a real contract + proof that the NFT was actually escrowed.

These fixes were implemented with minimal, targeted changes and follow industry best practices for escrow contracts. The V4 contract has been thoroughly reviewed against the original findings.

Full technical audit report (10 pages, with code snippets and detailed analysis) is available for developers and auditors. Download link at the bottom of this summary.

How the Fixes Work (Visual Overview)

Below are simple before/after diagrams showing the problems we fixed and how OTCSwap V4 now protects users.

1. ERC20 Deposits — Preventing Under-Collateralization

VULNERABILITY #1 — ERC20 Deposit Validation

Fee-on-transfer / non-standard tokens could under-collateralize orders and drain shared balances

VULNERABLE (OTCSwapV2 / V3)	FIXED in OTCSwapV4
1. Maker calls createOrder() with ERC20 offer 2. _depositFrom() executes SafeERC20.safeTransferFrom() → Token contract may deduct fee or transfer less 3. No balanceBefore / balanceAfter validation 4. Order stored with declared asset.amount (e.g. 1000) 5. Later acceptOrder() or cancelOrder() calls _transferOut() 6. Payout uses stored amount — may consume other users' escrowed tokens to cover the shortfall IMPACT: • Under-collateralized escrow • Cross-order balance drainage possible • Honest users' cancels/fills can revert • Permanent lock risk on depleted pools RISK	1. createOrder() / acceptOrder() → _depositFrom() 2. _requireContract(token) — ensures real contract 3. uint256 balanceBefore = IERC20(token).balanceOf(this) 4. safeTransferFrom(from, this, amount) 5. uint256 balanceAfter = IERC20(token).balanceOf(this) 6. uint256 received = balanceAfter - balanceBefore 7. require(received == asset.amount, "ERC20 amount mismatch") → Reverts immediately on any fee/shortfall/non-standard RESULT: • Only exact-amount deposits are accepted • No under-collateralization possible • No cross-order drainage or shared-balance attacks • All subsequent payouts/cancels are fully backed SECURE
Key Fix: _depositFrom() lines ~318-325 — balance delta + exact match require()	

Result: Only orders backed by the exact declared amount of tokens are created. No more silent shortfalls or draining of other users' funds.

2. ETH Handling — No More Locked Funds

VULNERABILITY #2 — Escrowed ETH Permanently Lockable

Push-payment model to non-payable recipients (contracts without receive/fallback) traps ETH forever

VULNERABLE (V2 / V3)	FIXED in OTCSwapV4
cancelOrder() or acceptOrder() → _transferOut(ETH) _transferOut ETH branch: (bool success,) = payable(to).call{value: payout}(""); require(success, "ETH transfer failed"); If 'to' is a contract without payable receive()/fallback(): • .call reverts (or returns false) • require() fails → entire cancel/fill reverts • No alternative withdrawal path exists • No way to change payout address IMPACT: • ETH escrowed in order is PERMANENTLY TRAPPED • Maker cannot cancel if non-payable • Fills to non-payable takers also blocked • No pull-based recovery mechanism X LOCKED	_transferOut() for ETH no longer pushes: <pre>if (payout > 0) { claimableEth[to] += payout; emit EthCredited(to, payout); }</pre> return; // never reverts on non-payable New public function: <pre>function withdrawEth() external nonReentrant { uint256 amount = claimableEth[msg.sender]; require(amount > 0, "No claimable ETH"); claimableEth[msg.sender] = 0; (bool ok,) = payable(msg.sender).call{value: amount}(""); require(ok, "ETH withdraw failed"); }</pre> RESULT: Pull-based, always recoverable by the rightful owner. PULL

Key Fix: claimableEth mapping + withdrawEth() – replaces all direct ETH pushes in fills/cancels

3. NFT Offers — Blocking Fake Assets

VULNERABILITY #3 — Fake ERC721/ERC1155 NFT Offers

Malicious makers could advertise non-existent NFTs, receive real taker assets, and deliver nothing

VULNERABLE (V3)	FIXED in OTCSwapV4
Maker creates order: <pre>offer: [{assetType: ERC721, token: address(0), tokenId: 1}] want: [{assetType: ERC20, token: USDC, amount: 1000e6}]</pre> createOrder() → _depositFrom(): <pre>IERC721(address(0)).transferFrom(...) // no-op, succeeds! No _requireContract, no ownerOf check after Order stored as Open with fake NFT "escrowed"</pre> Taker calls acceptOrder(): <pre>Deposits real 1000 USDC to contract _transferOut sends USDC to maker (real value!) _transferOut tries NFT to taker → no-op (nothing received)</pre> IMPACT: Maker steals taker's assets with zero-risk fake offer X THEFT	_requireContract(asset.token) added for all NFT legs: <pre>require(token != address(0), "Invalid token"); require(token.code.length > 0, "Token is not a contract");</pre> ERC721 deposit path: <pre>IERC721(token).transferFrom(from, this, tokenId); require(!IERC721(token).ownerOf(tokenId) == address(this), "ERC721 deposit failed");</pre> ERC1155 deposit path: balanceBefore/After + exact delta check Same post-condition checks added to _transferOut() payouts RESULT: • Fake token addresses (0x0, EOAs) immediately rejected • Non-compliant contracts that don't actually transfer are caught • Only real, escrowed NFTs can be offered in orders • Takers are protected — they only send value for real assets ✓ VERIFIED

Key Fix: _depositFrom() + _transferOut() + new _requireContract() for ERC721/ERC1155 (lines ~327-371)

□ Your assets are now protected by these improvements.

OTCSwap V4 rejects unsafe deposits, prevents locked ETH, and only allows real escrowed NFTs to be traded. We take security seriously and will continue to monitor and improve the protocol.

Download the Full Technical Report

For developers, auditors, or anyone who wants the complete details (including exact code changes and proof-of-concept analysis), you can download the full 10-page technical audit report here:

[📄 Full Technical Audit Report \(PDF\)](#)
OTCSwapV4_Security_Audit_Report.pdf — 10 pages with code-level details

This public summary and the full technical report were prepared in July 2026. SwapNFT.xyz is committed to transparency and the ongoing security of user funds.