Security Audit Report

# Cider.Trade Platform

Asfalia Audit on June 14, 2026

# Table of Contents

# Executive Summary

This report presents the findings of a comprehensive security audit of the Cider.trade platform across four operational domains: API and webhooks, CIDER staking, USDC deposit and withdrawal flows, and OTC barrel exchange contracts.

The audit encompassed full source-code review of the Next.js application, on-chain smart contract analysis, integration testing, and verification of production deployment on Base mainnet (chainId 8453).

All four audited domains completed successfully. Security controls are implemented, tested, and operational. No critical or high-severity vulnerabilities were identified in the current codebase.

| Domain | Audit Result |
|---|---|
| API / Webhook Audit | **PASSED** |
| CIDER Staking Audit | **PASSED** |
| USDC Deposit / Withdrawal Audit | **PASSED** |
| OTC Contracts Audit (Barrel Exchange) | **PASSED** |

## Automated Verification

- Platform unit tests: 54 / 54 passed
- Smart contract tests: 9 / 9 passed
- Production build: Successful

## Deployed Infrastructure - Base Mainnet

| | |
|---|---|
| **CIDER token** | 0x47cECFC5FF6209C1e1D7492D77ADcF90130cE1fB |
| **CiderStaking** | 0x50A992c3166BF42DEBbDc306e5cc0Db70121931e |
| **BarrelEscrow** | 0x94cf32807E8f0501271EF82932aD04881097e891 |
| **Platform treasury** | 0xe6c046A0675F662dA2a7eCBc5Be0DACDa2A39278 |

**OVERALL VERDICT: AUDIT SUCCESSFUL - PLATFORM SECURE FOR PRODUCTION OPERATION**

# Project Summary

## Overview

| | |
|---|---|
| **Project Name** | Cider.Trade Platform |
| **Platform** | The AI Trading Orchard |
| **Production URL** | https://cider.trade |
| **Repositories** | cider-platform (application); contracts/cider-trade (Base contracts) |
| **Chain** | Base Network (chainId 8453) |
| **Languages** | TypeScript, Solidity |
| **Document ID** | CIDER-SEC-AUDIT-2026-001 |
| **Assessment Date** | June 14, 2026 |
| **Report Version** | 1.0 - Final |
| **Classification** | Official - Stakeholder Distribution |
| **Audit Methodology** | Source-code review, smart contract analysis, integration testing, production verification |

## Audit Summary

| Total Domains | Pending | Acknowledged | Unresolved | Partially Resolved | Resolved / Passed |
|---|---|---|---|---|---|
| 4 | 0 | 0 | 0 | 0 | 4 |

Excellence Security Scoring: 100 / 100

## Vulnerability Summary

| Critical | High | Medium | Low | Minor | Informational |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |

# 1. API / Webhook Audit

**AUDIT RESULT: PASSED**

## 1.1 Scope

Review of all HTTP API routes under app/api/, webhook ingestion endpoints, API key authentication middleware, signal processing pipeline, and cron job authorization. Fifteen route modules and eighteen HTTP handlers were examined.

## 1.2 Authentication & Authorization

The platform implements a layered authentication model appropriate to each endpoint class:

| Endpoint Class | Mechanism | Status |
|---|---|---|
| TradingView webhooks | Per-bot webhook secret | PASSED |
| AI signal webhooks | Per-bot webhook secret | PASSED |
| Cron / admin jobs | Bearer CRON_SECRET | PASSED |
| Agent / v1 API | SHA-256 hashed API keys | PASSED |
| OTC trade mutations | API key + minimum trade scope | PASSED |
| User sessions | NextAuth JWT (httpOnly cookies) | PASSED |

Per-bot webhook secrets are enforced in production: bots without an individual webhookSecret configured reject incoming signals rather than falling back to a shared global credential. Secret comparison uses constant-time equality (safeEqualStrings) to prevent timing side-channel attacks.

API key scope hierarchy (read -> trade -> full) is enforced on all OTC mutation endpoints. Keys below the required scope receive HTTP 403 Forbidden.

Production startup validates that NEXTAUTH_SECRET and CRON_SECRET are set to secure, non-default values. The application refuses to boot in production if either secret matches a development placeholder.

## 1.3 Input Validation & Rate Limiting

OTC order creation and take endpoints validate request bodies with Zod schemas (lib/otc/schemas.ts), rejecting malformed addresses, non-positive amounts, and invalid expiry values before any database or on-chain operation.

Shared schemas govern both the v1 REST API and the agent-facing OTC API, ensuring consistent validation across all order entry surfaces.

Rate limiting is active on:

- TradingView and AI signal webhook endpoints (IP-based, 120 requests/minute)
- All authenticated OTC trade POST endpoints (tier-based: standard 60/min, premium 300/min, unlimited 10,000/min)
- Ourbit relay API (API key tier limits)

## 1.4 Webhook Security

Signal ingestion applies the following controls on every webhook delivery:

| Control | Implementation | Status |
|---|---|---|
| Secret verification | Per-bot timing-safe compare | PASSED |
| Payload validation | TradingView schema parser (validator.ts) | PASSED |

| Control | Implementation | Status |
|---|---|---|
| Secret non-persistence | Webhook secret stripped before MongoDB write | PASSED |
| Replay deduplication | 60-second window on (botId, action, pair, orderId) | PASSED |
| Rate limiting | IP-based throttle on both webhook routes | PASSED |
| Malformed JSON handling | HTTP 400 with no signal side-effects | PASSED |

The signal processor creates an audit record for every inbound webhook, tracks execution status, and routes trades through an idempotent execution pipeline with deduplication keys to prevent double-execution on alert replay.

## 1.5 Cron & Admin Endpoints

Scheduled job endpoints (/api/cron) and the Telegram smoke-test endpoint (/api/admin/telegram/test) require a valid Bearer CRON_SECRET validated with constant-time comparison. The Telegram test endpoint restricts outbound chatId values to configured allowlist entries (TELEGRAM_SIGNALS_CHAT_ID, TELEGRAM_HFT_CHAT_ID).

## 1.6 OTC API Integrity

Take-order operations use atomic database updates (findOneAndUpdate with status: pending and expiresAt > now preconditions) to guarantee exactly one taker per order under concurrent requests. On-chain taker assignment is rolled back if the escrow contract call fails, preserving database and chain consistency.

## 1.7 Test Coverage

| Coverage Area | Verification | Result |
|---|---|---|
| API key scope enforcement | 11 unit tests | PASSED |
| Timing-safe string comparison | 3 unit tests | PASSED |
| Signal payload validation | 18 unit tests | PASSED |
| OTC schema and token registry | Audit-remediation suite | PASSED |

## 1.8 Verdict

The API and webhook layer demonstrates production-grade authentication, authorization, input validation, rate limiting, and replay protection. All examined controls function as designed.

**API / WEBHOOK AUDIT: PASSED**

# 2. CIDER Staking Audit

**AUDIT RESULT: PASSED**

## 2.1 Scope

Review of CiderStaking.sol (deployed at 0x50A992c3166BF42DEBbDc306e5cc0Db70121931e on Base mainnet), the live CIDER ERC-20 token  0x47cECFC5FF6209C1e1D7492D77ADcF90130cE1fB), platform integration (hooks/useCiderStaking.ts), and on-chain eligibility reads for funding and revenue distribution.

## 2.2 Contract Architecture

CiderStaking implements a Synthetix-style multi-pool reward system:

| Function | Purpose |
|---|---|
| stake(amount) | Lock CIDER tokens; begin earning rewards |
| requestUnstake(amt) | Initiate 7-day cooldown; stop reward accrual |
| completeUnstake() | Withdraw after cooldown period elapsed |
| getReward() | Claim accrued rewards from all active pools |
| addRewardPool(token) | Owner adds a new ERC-20 reward token pool |
| startRewardPool(...) | Owner funds and activates a reward distribution |
| extendRewardPool(...) | Owner extends an active pool duration / emission |

Staking token: live CIDER (CIDER.TRADE, 18 decimals) on Base mainnet.

## 2.3 Security Controls - On-Chain

| Control | Status | Evidence |
|---|---|---|
| ReentrancyGuard | PASSED | All state-changing functions protected |
| 7-day unstake cooldown | PASSED | completeUnstake reverts during cooldown |
| Reward accrual halt on unstake | PASSED | balanceOf returns 0 after requestUnstake |
| rescue() staking token block | PASSED | Cannot rescue staked CIDER |
| rescue() reward token block | PASSED | isRewardToken mapping guards all pools |
| Owner-only pool management | PASSED | addRewardPool / startRewardPool restricted |
| SafeERC20 token transfers | PASSED | OpenZeppelin-style safe transfer pattern |
| Zero-address checks | PASSED | Constructor and transfer guards present |

The rescue() function - intended for recovering mistakenly sent tokens - is correctly restricted from accessing both the staking token and any token registered as an active reward pool, protecting staker entitlements.

## 2.4 Security Controls - Platform

| Control | Status | Evidence |
|---|---|---|
| On-chain stake reads | PASSED | readOnChainStakedAmount when contract set |
| Eligibility gate | PASSED | userMeetsFundingGate uses on-chain balance |
| Revenue distribution | PASSED | On-chain stakes used for allocation weight |
| Client staking UI | PASSED | useCiderStaking hook: stake / unstake / claim |
| Wagmi + Base chain enforcement | PASSED | switchChainAsync to Base before writes |
| ERC-20 approve before stake | PASSED | Allowance check and approval flow |

When STAKING_CONTRACT_ADDRESS is configured, the platform reads stake balances exclusively from the on-chain CiderStaking contract, ensuring eligibility and revenue calculations reflect live staking state.

## 2.5 Hardhat Test Results

| Test | Result |
|---|---|
| Stakes and reports stakedOf | PASSED |
| Enforces unstake cooldown | PASSED |
| Completes unstake after cooldown; stops earning | PASSED |
| Accrues and claims rewards from reward pool | PASSED |
| Extends an active reward pool | PASSED |
| Blocks rescue of reward tokens | PASSED |

CiderStaking contract tests: 6 / 6 passed

## 2.6 Verdict

CiderStaking contract logic is sound, fully tested, and deployed on Base mainnet. Reward pool mechanics, cooldown enforcement, and rescue protections operate correctly. Platform integration reads on-chain state for all funding and revenue decisions.

**CIDER STAKING AUDIT: PASSED**

# 3. USDC Deposit / Withdrawal Audit

**AUDIT RESULT: PASSED**

## 3.1 Scope

Review of the complete USDC funding lifecycle: user deposit verification, FundingAttempt audit trail, Allocation creation, orphan reconciliation cron, funding eligibility gates, tier caps, and admin-only withdrawal execution.

## 3.2 Deposit Flow

The fundBot server action implements a multi-stage verification pipeline:

| Stage | Check | Status |
|---|---|---|
| 1 | Session authentication | PASSED |
| 2 | Funding eligibility (stake threshold or code) | PASSED |
| 3 | Duplicate txHash rejection (DB + unique index) | PASSED |
| 4 | Bot exists, active, and public | PASSED |
| 5 | Pool capacity remaining | PASSED |
| 6 | On-chain USDC transfer verification | PASSED |
| 7 | Sender wallet matches connected user address | PASSED |
| 8 | Credit capped to min(actual, declared, remaining) | PASSED |
| 9 | Per-bot tier cap enforcement | PASSED |
| 10 | Allocation created + bot.totalFunded updated | PASSED |
| 11 | FundingAttempt marked verified with audit metadata | PASSED |

### On-Chain Verification

verifyDeposit.ts confirms:

- Transaction succeeded on Base (status === "success")
- Transfer is USDC to the platform treasury address
- Amount meets minimum threshold (at least 99% of declared amount)
- Returns sender (from) address for wallet binding

## 3.3 Deposit Attribution & Anti-Fraud

| Control | Implementation | Status |
|---|---|---|
| Sender wallet binding | fundBot compares verification.from to user.walletAddress (case-normalized) | PASSED |
| TxHash uniqueness | Unique MongoDB index on Allocation.txHash | PASSED |
| Race condition handling | E11000 duplicate key catch on concurrent credit | PASSED |
| Attempt identity protection | recordFundingAttempt uses $setOnInsert for userId / botId / walletAddress; rejects cross-user rebinding of an existing txHash | PASSED |
| Credit amount cap | min(on-chain amount, declared amount, pool headroom) | PASSED |

These controls ensure that a USDC deposit is credited exclusively to the wallet that sent it, exactly once, for the correct amount within pool limits.

## 3.4 Orphan Reconciliation

The fundingReconciliation cron (runs every 2 minutes) provides automated recovery for deposits where the user's USDC arrived on-chain but the fund dialog did not complete server-side verification:

| Capability | Status |
|---|---|
| Base USDC Transfer log scanning | PASSED |
| Reorg-safe cursor (5-block buffer) | PASSED |
| TxHash direct match to FundingAttempt | PASSED |
| Wallet + amount heuristic (24h window) | PASSED |
| Sender wallet verification | PASSED |
| Funding eligibility gate | PASSED |
| Credit amount cap | PASSED |
| Tier cap enforcement | PASSED |
| Ambiguous / unmatched -> admin alert | PASSED |
| Telegram notification on orphan | PASSED |

Shared eligibility logic lives in lib/funding/eligibilityCore.ts and is used by both the interactive fundBot path and the reconciliation cron, ensuring consistent guardrails across all credit paths.

## 3.5 Withdrawal Controls

User-initiated withdrawals are disabled at the application layer. All USDC returns to users are processed exclusively through the admin withdrawal flow:

| Control | Status |
|---|---|
| requireAdmin() gate | PASSED |
| On-chain USDC send via treasury key | PASSED |
| WithdrawalAttempt audit record | PASSED |
| Receipt verification before database update | PASSED |
| Unique allocationId on attempt rows | PASSED |
| Allocation status -> withdrawn | PASSED |

The executeWithdrawal module validates treasury key configuration, checks USDC balance, broadcasts the transfer, and waits for on-chain receipt confirmation before marking the withdrawal complete.

## 3.6 Allocation Transfer Security

Inter-bot allocation transfers require wallet-signed intents with nonce-based replay protection and timing-safe signature verification - preventing unauthorized movement of user allocations between bots.

## 3.7 Test Coverage

| Coverage Area | Verification | Result |
|---|---|---|
| Deposit sender matching | 7 unit tests | PASSED |
| Funding eligibility core | Verified | PASSED |
| Allocation transfer math | 4 unit tests | PASSED |
| Execution idempotency keys | 3 unit tests | PASSED |

## 3.8 Verdict

USDC deposit flows implement production-grade on-chain verification, sender attribution, duplicate prevention, and automated orphan recovery. Withdrawals are correctly restricted to admin processing with full audit trail. All funding paths enforce eligibility, pool capacity, and tier caps consistently.

**USDC DEPOSIT / WITHDRAWAL AUDIT: PASSED**

# 4. OTC Contracts Audit (Barrel Exchange)

**AUDIT RESULT: PASSED**

## 4.1 Scope

Review of BarrelEscrow.sol (deployed at 0x94cf32807E8f0501271EF82932aD04881097e891 on Base mainnet), platform escrow integration (lib/barrel/escrow.ts), OTC order lifecycle (app/actions/otc.ts), cron monitors, client deposit hooks, and API endpoints for agent and user order management.

## 4.2 Contract Architecture

BarrelEscrow implements dual-sided programmable escrow for OTC token swaps:

| Phase | On-Chain Function | Actor |
|---|---|---|
| Registration | registerOrder | Platform operator |
| Taker assignment | assignTaker | Platform operator |
| Maker deposit | depositOffer | Maker wallet |
| Taker deposit | depositRequest | Taker wallet |
| Settlement | settle | Platform operator |
| Cancellation | cancelOrder | Platform operator |
| Expiry refund | refundExpired | Platform operator or cron |

Fee: 1% (100 basis points) deducted from each side on settlement, routed to the configured treasury address (0xe6c046A0675F662dA2a7eCBc5Be0DACDa2A39278).

Supported tokens: USDC, USDT (Base USDC alias), CIDER - validated at order creation on both server and client.

## 4.3 Security Controls - On-Chain

| Control | Status | Detail |
|---|---|---|
| ReentrancyGuard on all mutations | **PASSED** | deposit / settle / cancel / refund / rescue |
| assignTaker immutability | **PASSED** | Cannot overwrite after assignment |
| assignTaker post-deposit block | **PASSED** | Reverts if taker already deposited |
| Dual-deposit settlement gate | **PASSED** | settle requires both sides funded |
| Expired order refund | **PASSED** | refundExpired returns deposits to parties |
| Cancel with automatic refund | **PASSED** | cancelOrder triggers internal _refund |
| Owner-only registration / settlement | **PASSED** | onlyOwner on register / assign / settle / cancel |
| Zero-address and amount validation | **PASSED** | Constructor and registerOrder guards |

## 4.4 Security Controls - Platform

| Control | Status | Detail |
|---|---|---|
| Escrow registration rollback | PASSED | Database order deleted if chain registration fails |
| Atomic take-order | PASSED | findOneAndUpdate on pending status |
| Chain rollback on assign failure | PASSED | Database reverted to pending if assignTaker fails |
| Cancel chain-before-database | PASSED | On-chain cancel must succeed before database update |
| Expired order refund cron | PASSED | otcDepositMonitor dual-pass (active + expired) |
| Stale order cleanup with chain refund | PASSED | staleOrderCleanup calls refundExpiredOnChain |
| Token pair validation | PASSED | validateOtcTokenPair on order create |
| Client deposit flow | PASSED | useBarrelEscrowDeposit + EscrowDepositButton |
| Zod validation on API routes | PASSED | CreateOrderSchema / TakeOrderSchema |
| API key scope on trade endpoints | PASSED | Minimum trade scope required |
| Rate limiting on trade endpoints | PASSED | Tier-based limits enforced |

## 4.5 Order Lifecycle Integrity

The platform maintains database and on-chain state in sync through a coordinated lifecycle:

- User or agent creates order -> MongoDB record + registerOrder on-chain
- Counterparty takes order -> atomic database match + assignTaker on-chain
- Maker deposits offer token -> depositOffer via wallet hook
- Taker deposits request token -> depositRequest via wallet hook
- Cron detects both deposits -> settle on-chain -> order marked completed
- Expired / cancelled orders -> refundExpired / cancelOrder -> deposits returned

The otcDepositMonitor cron runs every 2 minutes, syncing on-chain deposit flags, triggering settlement when both sides are funded, and initiating on-chain refunds for expired orders that still hold deposits.

## 4.6 Hardhat Test Results

| Test | Result |
|---|---|
| Registers and tracks deposits | PASSED |
| Prevents assignTaker overwrite after taker deposit | PASSED |
| Prevents assignTaker when taker already assigned | PASSED |

BarrelEscrow contract tests: 3 / 3 passed

## 4.7 Verdict

BarrelEscrow contract logic is secure, deployed on Base mainnet, and fully integrated with the platform. Order creation, matching, deposit, settlement, cancellation, and expiry refund flows are implemented end-to-end with database and on-chain consistency guarantees. Client-side deposit hooks enable makers and takers to complete escrow deposits directly from the Barrel Exchange UI.

**OTC CONTRACTS AUDIT (BARREL EXCHANGE): PASSED**

# Conclusion

This security audit examined the Cider.trade platform across four operational domains covering all user-facing financial flows: API ingress, staking, USDC funding, and OTC escrow trading.

Every domain completed successfully. Security controls - including authentication, authorization, input validation, rate limiting, on-chain verification, sender attribution, duplicate prevention, atomic order matching, escrow lifecycle management, and automated reconciliation - are implemented, tested, and operational on Base mainnet.

| Domain | Result |
|---|---|
| API / Webhook Audit | **PASSED** |
| CIDER Staking Audit | **PASSED** |
| USDC Deposit / Withdrawal Audit | **PASSED** |
| OTC Contracts Audit (Barrel Exchange) | **PASSED** |

- Automated tests: 63 / 63 passed (54 platform + 9 contract)
- Production build: Successful
- On-chain deployment: CiderStaking + BarrelEscrow live on Base mainnet

**OVERALL AUDIT VERDICT: SUCCESSFUL - ALL DOMAINS PASSED**

Document prepared: June 14, 2026

Cider.trade Platform Security Assessment - Final Release

END OF REPORT

# Appendix

## Audit Reference

| | |
|---|---|
| **Production URL** | https://cider.trade |
| **Application Repository** | cider-platform |
| **Contract Repository** | contracts/cider-trade |
| **Chain** | Base mainnet (chainId 8453) |
| **CIDER Token** | 0x47cECFC5FF6209C1e1D7492D77ADcF90130cE1fB |
| **CiderStaking** | 0x50A992c3166BF42DEBbDc306e5cc0Db70121931e |
| **BarrelEscrow** | 0x94cf32807E8f0501271EF82932aD04881097e891 |
| **Platform Treasury** | 0xe6c046A0675F662dA2a7eCBc5Be0DACDa2A39278 |
| **Assessment Date** | June 14, 2026 |
| **Document ID** | CIDER-SEC-AUDIT-2026-001 |

## Audit Coverage Categories

| Category | Coverage | Resolution |
|---|---|---|
| Authentication / authorization | API keys, JWT sessions, webhook secrets, admin gates | **PASSED** |
| Input validation | Zod schemas, address / amount / expiry validation | **PASSED** |
| Rate limiting | Webhook, OTC, agent and relay throttling | **PASSED** |
| Replay / idempotency | Signal deduplication, transfer nonce controls | **PASSED** |
| Smart contract security | Reentrancy, ownership, escrow, rescue restrictions | **PASSED** |
| Financial integrity | USDC verification, duplicate prevention, sender binding | **PASSED** |
| Operational recovery | Orphan reconciliation, stale order cleanup, refunds | **PASSED** |
| Build and tests | 54 platform tests + 9 contract tests; production build | **PASSED** |

## Final Resolution Matrix

| Domain | Resolution | Status |
|---|---|---|
| API / Webhook | No critical, high, medium, low or unresolved findings | **PASSED** |
| CIDER Staking | All controls and six contract tests completed successfully | **PASSED** |
| USDC Deposit / Withdrawal | All deposit, reconciliation and withdrawal controls verified | **PASSED** |
| OTC Barrel Exchange | All escrow lifecycle and three contract tests completed successfully | **PASSED** |

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Asfalia's prior written consent in each instance.This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Asfalia to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-freenature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. Asfalia's position is that each company and individual are responsible for their own due diligence and continuous security. Asfalia's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Asfalia is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Project is potentially vulnerable to 3rd party failures of service - namely in the form of APIs providing the price for the currencies used by the project. Project could become at risk if these APIs provided incorrect pricing.

Audit does not claim to address any off-chain functions utilized by the project.


The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone.

With over 30 years of combined experience in the DeFi space, our team is highly dedicated to delivering a product that is as streamlined and secure as possible. Our mission is to set a new standard for security in the auditing sector, while increasing accessibility to top tier audits for all projects in the crypto space. Our dedication and passion to continuously improve the DeFi space is second to none.