netwIRC

audited.xyz
by zack.eth

Date
Mar 22, 2026 08:14 UTC

Commit
main
305 files, 22,498 lines

Summary

netwIRC is a real-time chat platform built on Ruby on Rails with a native iOS client and a companion Node.js CLI tool (PromptSwap) for peer-to-peer LLM inference trading. The Rails backend powers IRC-style chat rooms with end-to-end encryption via AES-256-GCM, a marketplace with escrow-based job transactions, wallet and Stripe payment processing, and AI-powered content moderation. The architecture spans approximately 135 application-layer files, 25 database tables, and 25 configuration files on the server side, complemented by a 17-file Swift iOS client using CryptoKit and a 15-file Node.js CLI. Real-time messaging is delivered through ActionCable WebSockets, background processing through Sidekiq, and persistent state through PostgreSQL with Redis for caching and pub/sub.

The codebase demonstrates deliberate security engineering across multiple layers. Authentication supports three independent flows — username/password via Devise with bcrypt at 12 stretches and account lockout, WebAuthn passkeys with server-derived Relying Party IDs, and Ethereum wallet signature verification — all protected by Rack::Attack rate limiting on authentication endpoints. Transport security is enforced through production SSL with `force_ssl` and `assume_ssl`, a well-structured Content Security Policy using script nonces, and restricted ActionCable origins. The iOS client stores authentication tokens and E2E identity keys in the Keychain with `kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly` protection and implements E2E encryption through a Swift actor for thread-safe cryptographic operations. The database layer consistently applies foreign key constraints and unique indexes on critical join tables, and financial column integrity is enforced at the application layer with proper validation boundaries.

The overall security posture of netwIRC is strong. No low-severity or above vulnerabilities were identified across the full codebase during this audit. The application demonstrates mature patterns throughout: parameterized queries prevent SQL injection, server-side configuration derivation for WebAuthn eliminates client-controlled origin attacks, restrictive file permissions on CLI credentials follow least-privilege principles, and the clean separation between the iOS thin client and the Rails authorization layer places trust boundaries at appropriate architectural seams. The E2E encryption implementation correctly uses authenticated encryption with proper nonce and tag handling, and the migration from Lightning/sats to Stripe/USD-cents was executed cleanly without leaving residual attack surface.

Conclusion

The netwIRC codebase reflects a security-conscious development methodology across all layers. The Rails application enforces access controls consistently through Devise authentication, role-based room memberships, and session management with appropriate cookie security attributes. Configuration management follows environment-specific hardening — Sentry suppresses PII, parameter filtering covers sensitive fields, and host authorization whitelisting is scoped per deployment target. The test suite provides comprehensive coverage of security-sensitive paths including authentication bypass attempts, insufficient balance checks, prompt injection sanitization, and escrow lifecycle state transitions, indicating that security considerations are integrated into the development workflow rather than applied retroactively.

No vulnerabilities at low-severity or above were identified in this audit. The codebase's primary risk surfaces — E2E key exchange, marketplace escrow state machines, wallet balance operations, and multi-factor authentication flows — were reviewed in detail and found to implement appropriate controls. The PromptSwap CLI, while architecturally sound in its token storage and API authentication patterns, operates in a higher-risk domain as a peer-to-peer inference marketplace, but its current deployment scope and authentication requirements provide adequate containment. The iOS client's use of platform cryptographic primitives and its delegation of authorization decisions to the server reflect correct mobile security architecture.

netwIRC is suitable for production deployment in its current state. The application's layered security model — transport encryption, authenticated sessions, role-based access control, E2E message encryption, and content moderation — provides defense in depth across its chat, marketplace, and payment subsystems. To further harden the platform, the development team should consider adding certificate pinning to the iOS client's transport layer and expanding Rack::Attack coverage to include unauthenticated API traffic. These are incremental improvements to an already solid security foundation rather than prerequisites for deployment.