Threat Model Report - Zuhsyn Edubox v1.0.0

Prepared by: Ibrahim Idris Cybersecurity Analyst

Date: July 21, 2025

1. Objectives

The goal of this threat model is to:

- Identify potential security threats to Zuhsyn Edubox.
- Highlight assets that must be protected (user data, tokens, quizzes).
- Recommend mitigations to reduce risks.
- Help developers securely build and scale the platform.

2. Assets to Protect

- User credentials: Email, password
- JWT tokens: Access and refresh tokens stored on client
- User profile data: Levels, points, subject progress
- Quiz submissions: Lesson answers, scores
- Leaderboard data: Rankings, performance metrics
- Email communications: Password reset tokens, verification links
- API endpoints: Auth, lessons, profiles, leaderboard

3. Architecture Overview

- Frontend: React Native app (JavaScript)
- Backend: Django REST API with JWT Authentication
- Database: PostgreSQL or MySQL
- Storage: AsyncStorage on mobile

- Transport Layer: HTTPS (API communication)
- Email Service: Gmail SMTP for password reset

4. STRIDE Threat Modeling

S - Spoofing: Prevent with strong passwords, rate-limiting, 2FA

T - Tampering: Validate input, audit logs, hash integrity

R - Repudiation: Log actions, use timestamps/IPs

I - Info Disclosure: Enforce HTTPS, expire tokens

D - DoS: Use throttling, caching

E - Privilege Escalation: RBAC, secure role checks

5. Key Threat Scenarios & Mitigations

- Auth Bypass: Use refresh flow, rotate tokens
- Token Storage: Prefer secure APIs, refresh often
- Insecure API: Enforce HTTPS, HSTS
- Quiz Forgery: Server-side validation, hash user actions
- Leaderboard Tampering: Avoid client-side score logic
- Email Abuse: Rate-limit reset requests

6. Mitigation Summary

- JWT: Short TTL, refresh securely
- Storage: SecureStore/EncryptedStorage over AsyncStorage
- API: DRF permissions, input sanitization
- DB: Use ORM, validate inputs
- Email: Sign and expire reset links

7. Security Testing Plan

- API: Postman + OWASP ZAP
- Mobile: MobSF scans
- Rate Tests: ab, wrk
- Monitoring: Django logs, Sentry

8. Future Work

- Add 2FA, security headers
- Implement RBAC
- Integrate with SIEM tools
- Regular automated scans

9. Appendix

- OWASP Mobile Top 10
- JWT Best Practices
- OWASP Secure Coding Standards