## **2FA Login Sequence - Complete Authentication Flow** Evidence: auth\_routes.py:116, twofa\_routes.py:106,133, totp\_service.py:159-173 /twofa/verify-2fa **AuthService SecurityService** twofa\_routes.py check\_account\_lqckout(username) SELECT \* FROM account\_lockouts lockout data or null

## /auth/login **TOTPService Auth Database** User auth\_routes.py **Phase 1: Password Authentication** POST username + password | requires\_captaha(username) SELECT COUNT(\*) FROM login\_attempts failure\_count If failure\_count ≥ 3: reCAPTCHA v2 verification required Google reCAPTCHA API integration authenticate(username, password) SELECT \* FROM users WHERE username = ? user row (password hash totp\_enabled, etc.) **Argon2id Password Verification** Argon2id.verify(stored\_hash, password) eters: time\_cost=2, memory\_cost=19456, parallelism=1 ~120ms constant-time operation Success: True, user\_dict log\_login\_attempt(username, IP, UA, success=True) clear\_account\_lockout(username) 13 ☐ SESSION REGENERATION POINT #1 regenerate\_session() Prevents session fixation after password auth Evidence: auth\_routes.py:116 decorators.py:23 - Session ID regeneration implemented 14 Check user['totp\_enabled'] If totp\_enabled == 1: session['pending\_2fa\_user\_id'] = user['id'] session['pending\_2fa\_username'] = username → Step-up authentication required 302 Redirect to /twofa/verify-2fa **15 Phase 2: TOTP Verification (Step-Up Authentication) GET /twofa/verify-2fa** 16 Show 2FA form (6-digit code + backup code option) 17 **POST code=123456, use\_backup=false** 18 verify\_totp(user\_id, code) SELECT totp\_secret, totp\_enabled 20 encrypted\_secret, totp\_enabled=1 21 **TOTP Verification Process:** 1. Decrypt TOTP Secret Fernet symmetric encryption (AES-128-CBC + HMAC) PBKDF2 key derivation: 100,000 iterations 2. Replay Attack Prevention cache\_key = f"{user\_id}:{code}:{window}" Check in-memory used\_codes\_cache Evidence: totp\_service.py:159-173 3. TOTP Verification pyotp.TOTP(secret).verify(code, valid\_window=1) True (TOTP valid) 22 23 session.pop('pending\_2fa\_user\_id') session.pop('pending\_2fa\_username') ☐ SESSION REGENERATION POINT #2 regenerate\_session() Prevents session fixation after 2FA verification Evidence: twofa\_routes.py:133 decorators.py:23 - Session ID regeneration implemented 24 session['user\_id'] = user\_id session['username'] = username log\_security\_event('2fa\_verified', **302 Redirect to /home (Login Complete ✓)** 26 **Alternative Flow: Backup Code Verification Backup Code Verification Process:** 1. User enters backup code in format: ABCD-EFGH 8 alphanumeric characters with dash separator 2. Hash provided code: code\_hash = SHA256(code.upper().encode()) Evidence: totp\_service.py:201 3. Verify against stored backup codes: SELECT backup codes FROM users WHERE id = ? JSON array of SHA256 hashes (10 codes generated initially) 4. Single-Use Enforcement: Remove used code\_hash from array → UPDATE users SET backup\_codes = ? → Same session regeneration flow as TOTP (twofa routes.py:106) → Log security event: '2fa\_backup\_used' with remaining count After-spacessful backup code verification: **2FA Security Implementation Summary** ☐ Session Fixation Prevention (IMPLEMENTED): • Regeneration Point #1: After password authentication (auth\_routes.py:116) • Regeneration Point #2: After TOTP verification (twofa\_routes.py:133) or backup code (line 106) • Implementation: decorators.py:23 - regenerate\_session() decorator • Security: Prevents attacker from fixing session ID before authentication completes **5 TOTP Replay Attack Prevention: □** Additional Security Measures: • In-memory cache: {user\_id}:{code}:{time\_window} (totp\_service.py:159-173) • Rate Limiting: 5 req/min per-IP on /verify-2fa • Backup Codes: SHA256 hashed, single-use (10 total) • Time window: 30-second intervals (TOTP standard) • TOTP Secret: Fernet encrypted at rest • Cache cleanup: Removes entries older than 90 seconds (3 windows) • PBKDF2: 100,000 iterations for key derivation • Prevention: Same code cannot be used twice in same time window • Security Logging: All 2FA events audited

## **☐ TOTP Time Window Tolerance:** • valid\_window=1: Accepts codes from current window and ±1 adjacent windows • Tolerance: ±30 seconds (1 step before and after current window)

• Balances security with clock skew tolerance (RFC 6238 recommendation)

- ☐ Step-Up Authentication Pattern:
- Phase 1: Password validates identity • Phase 2: TOTP/Backup code validates possession

• Pending session: Temporary storage between phases • Final session: Only created after both factors pass