Security Layers (Defense in Depth)

27 Evidence-Based Security Mechanisms

PREVENT - Proactive Security Controls (14 mechanisms)

1. Argon2id Password Hashing
auth_service.py:22
OWASP params: time_cost=2, memory_cost=19456, parallelism=1

2. Password Validation
validators.py:28
NIST SP 800-63B: min 12 chars, max 128, diversity check, common block

3. Breach Detection
validators.py:65
HaveIBeenPwned API integration with k-anonymity model

4. CSRF Protection
app_auth.py:29
Flask-WTF CSRF tokens on all state-changing requests

5. Content Security Policy
security_headers.py:24
Strict CSP with self-origin and whitelisted CDNs

6. XSS Prevention
sanitization.py:8
HTML sanitization with bleach library

7. Session Security
app_auth.py:24
Secure, HttpOnly, SameSite=Lax cookies

8. Session Fixation Prevention
decorators.py:23
Session ID regeneration after authentication

9. TOTP Secret Encryption
totp_service.py:87
Fernet symmetric encryption with PBKDF2-derived keys

10. PKCE for OAuth2
oauth2_service.py:98
Mandatory S256 code challenge method

11. Exact Redirect URI Matching
oauth2_service.py:96
No wildcards or pattern matching allowed

12. HSTS Header
security_headers.py:36
1 year max-age with includeSubDomains

13. X-Frame-Options
security_headers.py:34
DENY to prevent clickjacking

14. X-Content-Type-Options
security_headers.py:33

DETECT - Monitoring & Alerting (6 mechanisms)

1. Login Attempt Tracking
security_service.py:63
Log all login attempts with timestamp, IP, user agent

2. Security Event Logging
security_service.py:24
Comprehensive audit log with severity levels

3. Failed Login Counting
security_service.py:135
Count recent failures within lockout window

4. TOTP Replay Detection
totp_service.py:159
In-memory cache of used codes per time window

5. Refresh Token Reuse Detection
oauth2_service.py:320
Track refresh_token_used flag, revoke family on reuse

6. Authorization Code Single-Use
oauth2_service.py:197
Mark code as used in transaction, reject if already used

RESPOND - Incident Response (6 mechanisms)

1. Account Lockout
security_service.py:159
15-minute lockout after 3 failed attempts

2. CAPTCHA Challenge
auth_routes.py:78
Google reCAPTCHA v2 after 3 failures

3. Rate Limiting
rate_limiter.py:28
429 response when rate limit exceeded

4. Token Family Revocation
oauth2_service.py:397
Revoke all tokens in family if reuse detected

5. Lockout Clearing
security_service.py:219
Clear lockout after successful login

6. Backup Code Depletion Warning
twofa_routes.py:120
Show remaining backup codes after use

Layered Defense Architecture

Prevent → Detect → Respond: Each layer complements the others for defense in depth

Transaction Safety: BEGIN IMMEDIATE used for race condition prevention in rate limiting, OAuth2 code validation, and token rotation