

# Security Hardening Plan and Implementation

## 1) Research Summary (Current Security Standards)

This hardening plan was based on primary guidance from OWASP, NIST, and framework docs.

### Key protocols and coding practices used

- Authentication and session management
- Use signed, time-bounded sessions.
- Use secure cookie attributes (HttpOnly, SameSite, Secure in TLS environments).
- Rotate sessions on login.
- Credential handling
- Use adaptive password hashing (PBKDF2/Argon2/bcrypt), never plaintext.
- Add lockout/rate limiting on login flows.
- CSRF protection
- Enforce anti-CSRF tokens for state-changing browser requests.
- Protect API state-change calls when cookie auth is used.
- SSRF protection
- Restrict URL schemes and destination networks.
- Block localhost/private/reserved IP resolution.
- Restrict LinkedIn enrichment to explicit linkedin.com profile URLs when opt-in is enabled.
- Security headers
- Add CSP, X-Frame-Options, X-Content-Type-Options, Referrer-Policy, Permissions-Policy.
- Add HSTS in secure deployments.
- Access control
- Enforce server-side RBAC checks for every sensitive route/API.
- Avoid client-side-only authorization.
- Gate destructive organizer actions (attendee deletion) behind explicit typed confirmation.
- Gate bulk attendee imports behind organizer auth + CSRF + rate limiting with strict row validation.
- Auditability
- Log security-relevant actions (login, denied actions, enrichment attempts, exports, admin operations).

## 2) Sources

- OWASP ASVS project: <https://owasp.org/www-project-application-security-verification-standard/>
- OWASP Cheat Sheet Series (index): <https://cheatsheetseries.owasp.org/>
- OWASP Session Management Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/Session\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)
- OWASP Authentication Cheat Sheet: [https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)
- OWASP Password Storage Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/Password\\_Storage\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)
- OWASP CSRF Prevention Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
- OWASP SSRF Prevention Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/Server\\_Side\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html)
- NIST SP 800-63B (Digital Identity Guidelines - Authentication): <https://pages.nist.gov/800-63-4/sp800-63b.html>
- FastAPI security docs: <https://fastapi.tiangolo.com/reference/security/>
- Starlette middleware docs: <https://www.starlette.io/middleware/>

- MDN HTTP security headers overview: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP>

### 3) Security Hardening Plan

#### Phase A: Immediate controls (implemented)

- Replace plaintext organizer auth with hashed credentials in DB.
- Add signed, expiring auth session cookie.
- Add CSRF checks on all state-changing form routes.
- Add CSRF checks for state-changing API routes using cookie auth.
- Add RBAC for UI and API routes.
- Add login rate limiting and organizer lockout controls.
- Add SSRF guardrails for external enrichment endpoint.
- Add security headers middleware.
- Add audit logging for security-sensitive actions.

#### Phase B: Near-term controls (next)

- Add per-user and per-endpoint distributed rate limiting (Redis-backed).
- Add account recovery and mandatory credential rotation policies.
- Add breached-password blocklist checks.
- Add API token service for machine-to-machine usage (instead of session cookie for API clients).

#### Phase C: Production-grade controls (next)

- Migrate auth to enterprise IdP/SSO (OIDC/SAML) with MFA.
- Add centralized audit pipeline (SIEM).
- Add secret manager integration and key rotation.
- Add threat-detection and alerting (anomaly/risk-based events).

### 4) Implemented Controls in This App

#### Authentication and session hardening

- Added AppUser table for organizer and attendee credentials (hashed password).
- Password hashing uses PBKDF2-HMAC-SHA256 with high iteration count.
- Sessions are signed and include expiration.
- Login sets signed auth cookie + CSRF cookie.
- Added startup guardrails that block insecure config in APP\_ENV=production.

Files:

- app/models.py (AppUser)
- app/services/security.py
- app/main.py

#### RBAC

- Role map defined for organizer and attendee.
- Route and API permissions enforced server-side.
- Attendee scoped to their own record for protected operations.

Files:

- app/main.py

#### CSRF protection

- Hidden CSRF token added to form POST flows.
- Double-submit verification for browser forms.

- X-CSRF-Token enforcement for state-changing API routes.

Files:

- app/services/security.py
- app/main.py
- app/templates/login.html
- app/templates/organizer.html
- app/templates/attendee.html

## Rate limiting and login lockout

- In-memory IP-based rate limiter on login, attendee create, intros, feedback, and enrichment flows.
- Organizer account lockout window after repeated failures.
- Added attendee account lockout window after repeated failures.

Files:

- app/services/security.py
- app/main.py

## SSRF protections

- Only https allowed.
- Hostname validation and resolution.
- Blocks private/loopback/link-local/reserved/multicast targets.
- Redirect hops are validated and bounded.
- HTML response size is bounded to prevent memory abuse.
- LinkedIn enrichment allows only linkedin.com profile URLs (/in/...) and requires attendee opt-in.

Files:

- app/services/external\_enrichment.py

## Security headers and host/https controls

- Added CSP and baseline hardening headers.
- Optional TrustedHostMiddleware via ALLOWED\_HOSTS.
- Optional HTTPSRedirectMiddleware via FORCE\_HTTPS.

Files:

- app/main.py

## Container/runtime hardening

- Added Docker image running as non-root user.
- Added container health checks via /health.
- Added deployment env template to enforce secure defaults in production.

Files:

- Dockerfile
- docker-compose.yml
- .env.example

## Audit logging

- Added AuditLog table.
- Logs login outcomes, feedback writes, intro actions, attendee creation/deletion, enrichment attempts, and exports.
- Logs login outcomes, feedback writes, intro actions, attendee creation/deletion, bulk imports, enrichment attempts, and exports.
- Added organizer audit view.

Files:

- app/models.py (AuditLog)
- app/services/audit.py
- app/main.py
- app/templates/organizer\_audit.html

## 5) Environment and Security Configuration

- AUTH\_SECRET
- COOKIE\_SECURE
- FORCE\_HTTPS
- ALLOWED\_HOSTS
- ORGANIZER\_EMAIL
- ORGANIZER\_PASSWORD
- ATTENDEE\_BOOTSTRAP\_PASSWORD
- APP\_ENV
- DATABASE\_URL
- PASSWORD\_ITERATIONS
- TOKEN\_TTL\_SECONDS
- LOCKOUT\_SECONDS

## 6) Validation Performed

- Existing tests still pass (11 passed).
- Verified:
  - unauthenticated API access is denied,
  - role restrictions are enforced,
  - CSRF-protected form requests fail without token,
  - API state changes require CSRF header,
  - security headers are present,
  - audit page is accessible to organizer role,
  - attendee login uses per-attendee passcode pattern (attendee123-<id>),
  - organizer bulk import enforces row-level validation and partial-failure reporting,
  - feedback tampering across attendee match scopes is blocked.

## 7) Residual Risk and Next Actions

Current hardening is strong for MVP/demo, but still not full enterprise production. Priority next actions:

- Replace in-memory rate limiter with Redis/distributed limit service.
- Add enterprise SSO + MFA and remove static attendee passcode model.
- Add database migrations/versioning for security schema changes.
- Add structured audit export to SIEM and alerting.
- Add automated security test suite (DAST/SAST + dependency scanning).