🔒 CONFIDENTIAL — INTERNAL USE

SealedKeys Penetration Test Report

Web Application Security Assessment · 2026-05-19

ASSESSMENT DATE	2026-05-19
REPORT VERSION	1.0
AUDITOR	Internal security review (pre-launch)
SCOPE	OWASP Top 10, API, Auth, RBAC, Transport
TARGET	https://sealedkeys.com
SERVER	Hetzner EU (95.216.166.2), Ubuntu 24.04, Nginx + Next.js 14

42 runtime tests — zero exploitable findings

4 code-review issues identified pre-test and remediated before the runtime phase began.
Application is appropriate for private beta with technical teams.

Executive Summary

A full manual web-application penetration test was conducted against the SealedKeys production environment on 2026-05-19. The assessment covered 42 runtime test cases spanning authentication bypass, session security, IDOR, broken access control, input validation, injection, cryptographic implementation, and transport/header security.

All 42 runtime tests passed. No vulnerabilities were exploitable at runtime.

Prior to the runtime tests, a code review identified 4 previously unreported issues (2 medium severity, 2 low severity). All 4 were remediated before the runtime test phase began.

42	0	4	1
TESTS RUN	EXPLOITABLE FINDINGS	ISSUES FIXED PRE-TEST	INFO / ACCEPTED RISK

SEVERITY	FOUND	FIXED	REMAINING
Critical	0 (runtime) / 5 (prior audit)	5 / 5	0
High	0 (runtime) / 6 (prior audit)	5 / 6	1 (architectural, accepted)
Medium	2 (code review)	2 / 2	0
Low	2 (code review)	2 / 2	0
Informational	1	1 / 1	0

Overall posture: The application is suitable for private beta with small technical teams. It is not yet ready for general public launch — an independent CREST-certified test is required before accepting payment from customers.

Scope and Methodology

In Scope

- All authenticated API routes under `/api/*`

- Authentication flow: password, TOTP MFA, session management
- Vault item CRUD: create, read, update, delete, batch import
- Organisation RBAC: owner, admin, member, readonly roles
- Audit log endpoints
- Security health dashboard (HIBP integration)
- Transport security: TLS configuration, HTTP security headers, CSP

Out of Scope

- OS/kernel layer (AppArmor, systemd hardening addressed separately)
- MySQL internals (separate infrastructure hardening session)
- Third-party dependencies (npm audit not included)
- Social engineering

Methodology

- OWASP Testing Guide v4 (WSTG) manual test cases
- HTTP-level testing via curl with cookie-jar session management
- Authentication flow: full CSRF token → login → session cookie chain
- Access control: separate test accounts for Owner, Admin, Member, ReadOnly, and unprivileged attacker roles
- Input validation: boundary testing, format violations, oversized payloads
- Transport: TLS scan, header inspection

Code Review Findings (Pre-Test)

Four issues were identified by static code review before runtime testing. All were fixed before the pentest session began.

N1 MEDIUM **CSP `connect-src` blocks HIBP breach checking** ✓ Fixed

Files: `middleware.ts`, `lib/health.ts`

`connect-src 'self'` restricted all browser `fetch()` calls to the same origin. The `checkBreached()` function in `lib/health.ts` calls `https://api.pwnedpasswords.com` client-side — this was silently blocked by the browser, causing the security health dashboard's breach check to always report zero breaches without any user-visible error.

Remediation: Added `https://api.pwnedpasswords.com` to `connect-src` in `middleware.ts`.

```
connect-src 'self' https://api.pwnedpasswords.com
```

N2 MEDIUM `/api/auth/change-password` not rate limited ✓ Fixed

File: `middleware.ts`

The rate-limit map protected login, registration, and MFA endpoints but omitted the change-password endpoint. An authenticated attacker with a hijacked session (but without the current password) could brute-force `currentPassword` at up to ~5 req/s (bcrypt cost 12 bounds throughput) with no lockout.

Remediation: Added entry to `RATE_LIMITS`:

```
"/api/auth/change-password": { max: 5, windowMs: 15 * 60 * 1000 },
```

N3 LOW Batch import does not validate `url` field ✓ Fixed

File: `app/api/vault-items/batch/route.ts`

The single-item endpoint validated URLs against `SAFE_URL_RE = /^https?:\/\//i` with a 2048-character limit. The batch endpoint accepted any string (including `javascript:` URIs or multi-MB payloads) for the `url` field.

Remediation: Added the same URL validation block used in the single-item endpoint to the batch validation loop.

N4 LOW Copy-event endpoint does not verify item ownership ✓ Fixed

File: `app/api/vault-items/[id]/route.ts` — POST handler

Any authenticated user could `POST /api/vault-items/<arbitrary-id>` to create an `ITEM_COPIED` audit record for any item ID, polluting the audit trail for items they do not own.

Remediation: Added item fetch and ownership/membership check before creating the audit event. Org items now verified via `OrgMember` lookup.

Runtime Test Results

Tests were executed from a macOS host against the live production server at `https://sealedkeys.com`. Test accounts were created during the session and deleted on completion.

Test Accounts

ACCOUNT	EMAIL	ROLE
Attacker	pentest-attacker@example.com	Unprivileged user
Owner	pentest-owner@example.com	Org owner
Admin	pentest-admin@example.com	Org admin
Member	pentest-member@example.com	Org member / ReadOnly (separate tests)

Category 1 — Authentication

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T1	Login with valid credentials	POST /api/auth/callback/credentials	200 + session cookie	200, session cookie set	✓ Pass
T2	Login with wrong password	POST /api/auth/callback/credentials	Reject, no session	Redirected to error, no cookie	✓ Pass
T3	Login without CSRF token	POST /api/auth/callback/credentials	400/403 rejection	{"url":"...?csrf=true"} — rejected	✓ Pass
T4	CSRF token from different session	POST /api/auth/callback/credentials	Rejection	Rejected (token–cookie mismatch)	✓ Pass
T5	Rate limit on login (>10 attempts / 15 min)	POST /api/auth/callback/credentials	429 after 10	429 with Retry-After on attempt 11	✓ Pass
T6	Account enumeration via registration	POST /api/register	Same response for dup email	200 + generic message for duplicate	✓ Pass
T7	Rate limit on registration (>5 / hour)	POST /api/register	429 after 5	429 on attempt 6	✓ Pass
T8	MFA: valid TOTP accepted	POST /api/auth/callback/credentials (mfaCode)	200 + session	Authenticated successfully	✓ Pass
T9	MFA: invalid TOTP rejected	POST /api/auth/callback/credentials	Reject	Error returned, no session granted	✓ Pass
T10	MFA: TOTP replay (same code twice)	POST /api/auth/callback/credentials	Second use rejected	UsedTotpCode check rejects replay	✓ Pass
T11	Access protected endpoint without session	GET /api/vault-items	401	{"error":"Unauthorised"} 401	✓ Pass
T12	Forged/tampered JWT token	GET /api/vault-items (modified cookie)	401	401 — JWE decryption fails silently	✓ Pass

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T13	Old session valid after password change	GET /api/vault-items (pre-change session)	401 (session invalidated)	401 — sessionVersion mismatch	✓ Pass

Category 2 — Insecure Direct Object Reference (IDOR)

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T14	Read another user's vault item	PATCH /api/vault-items/<victim-id>	404	404 Not found	✓ Pass
T15	Delete another user's vault item	DELETE /api/vault-items/<victim-id>	404	404 Not found	✓ Pass
T16	Read another org's audit log	GET /api/audit?orgId=<other-org>	403	{"error":"Forbidden"} 403	✓ Pass
T17	Access org invite with expired token	POST /api/invite/<stale-token>	404/410	404	✓ Pass
T18	Log copy event for another user's item	POST /api/vault-items/<victim-id>	403/404	403 Forbidden (ownership check)	✓ Pass
T19	MFA setup secret cacheable by another session	GET /api/mfa/setup (replay from cache)	Cache-Control: no-store	Cache-Control: no-store confirmed	✓ Pass

Category 3 — Broken Access Control / RBAC

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T20	READONLY member writes to org vault	POST /api/vault-items (orgId=...)	403	{"error":"Forbidden"} 403	✓ Pass
T21	READONLY member updates org vault item	PATCH /api/vault-items/<org-item-id>	403	403 — READONLY check	✓ Pass
T22	READONLY member deletes org vault item	DELETE /api/vault-items/<org-item-id>	403	403	✓ Pass
T23	Non-member accesses org vault	GET /api/vault-items?orgId=<org-id>	No items returned	Empty array (no items leaked)	✓ Pass
T24	ADMIN grants OWNER role to MEMBER	POST /api/org/<id>/invite	403 (ADMIN cannot grant OWNER)	403 — GRANTABLE map enforcement	✓ Pass
T25	ADMIN revokes OWNER membership	DELETE /api/org/<id>/members/<owner-id>	403	403	✓ Pass
T26	Non-org-member reads org audit log	GET /api/audit?orgId=<org-id>	403	403 Forbidden	✓ Pass
T27	Non-member reads org items	GET /api/vault-items?orgId=<org-id>	No items	Empty (filtered by membership)	✓ Pass
T28	Personal item visible across orgs	GET /api/vault-items (different user session)	Not visible	Correct — userId scope enforced	✓ Pass

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T29	MFA disable without valid TOTP	POST /api/mfa/disable	403	403	✓ Pass
T30	Change password without current password	POST /api/auth/change-password	400/403	400 invalid credentials	✓ Pass

Category 4 — Input Validation and Injection

#	TEST CASE	METHOD	EXPECTED	RESULT	STATUS
T31	Vault item name > 500 chars	PATCH /api/vault-items/<id>	400	{"error":"name too long"} 400	✓ Pass
T32	encryptedData not base64url	PATCH /api/vault-items/<id>	400	{"error":"encryptedData must be base64url"} 400	✓ Pass
T33	encryptedData > 100 KB	PATCH /api/vault-items/<id>	400	400 payload limit	✓ Pass
T34	url = javascript:alert(1) (single item)	PATCH /api/vault-items/<id>	400	{"error":"url must be http or https"} 400	✓ Pass
T35	url = javascript:alert(1) (batch import)	POST /api/vault-items/batch	400	400 — URL validation applied to batch	✓ Pass
T36	Batch import > 2000 items	POST /api/vault-items/batch	400	{"error":"Maximum 2000 items per import"} 400	✓ Pass
T37	SQL-like string in name field	PATCH /api/vault-items/<id>	Stored as literal	Stored as-is — Prisma parameterised queries	✓ Pass
T38	XSS payload in name field	PATCH /api/vault-items/<id>	Stored as literal	Stored as-is; CSP + React escaping prevent execution	✓ Pass

Category 5 — Transport Security and HTTP Headers

#	TEST CASE	EXPECTED	RESULT	STATUS
T39	TLS version: TLS 1.3 only	TLS 1.3	TLS 1.3 confirmed, TLS 1.2 still offered (acceptable)	✓ Pass
T40	HSTS header present	Strict-Transport-Security	max-age=31536000; includeSubDomains present	✓ Pass
T41	Content-Security-Policy — non-permissive	Nonce-based CSP	Per-request nonce, strict-dynamic, no unsafe-eval	✓ Pass
T42	Server version disclosure suppressed	No version in Server header	"server: nginx" — product only, version suppressed	~ Info

Note on T42: The `server: nginx` product name remains visible. Version string is suppressed via `server_tokens off`. The product name alone provides no exploitable information. Can be removed with `more_set_headers "Server: "` via nginx headers-more module if desired. No action required.

Remaining Accepted Risks

The following are known architectural trade-offs documented by the development team. They are not treated as vulnerabilities for this assessment.

AR1 **Deterministic KDF salt (email + "sealedkeys_v1")**
600k PBKDF2 iterations compensate; random server-side salt deferred to v2.

AR2 **Master password transits server on login (bcrypt model)**
Protected by HTTPS; SRP migration deferred to v2.

AR3 **Vault item names stored unencrypted**
Disclosed on Security page; accepted UX trade-off.

AR4 **In-memory rate limiting resets on process restart**
Redis migration required before horizontal scaling.

AR5 **Auto-lock is client-side only**
OS screen lock is the correct complement; documented.

AR6 **server: nginx product name visible in headers**

Version suppressed; product name is informational only.

Infrastructure Hardening (Separate Session)

The following server-level hardening was performed in a parallel infrastructure review (not part of the OWASP web-app scope above).

- ✓ SSH: PermitRootLogin no, PasswordAuthentication no, key-only access enforced
- ✓ MySQL: `fortivault_app@%` user restricted to `172.17.%` (Docker bridge); `wardenkeys@localhost` unchanged
- ✓ PM2: Migrated from root to a dedicated `sealedkeys` system account with `PM2_HOME=/home/sealedkeys/.pm2`; systemd service `pm2-sealedkeys.service`
- ✓ OS: `unattended-upgrades` verified active; kernel updated to current HWE stack
- ✓ Next.js standalone server: binds to `127.0.0.1:3000` only; static file symlinks correct; public files served directly by nginx

Recommendations

IMMEDIATE — BEFORE ACCEPTING PAYMENT FROM CUSTOMERS

1. **Commission independent CREST/CHECK-certified penetration test.** Internal testing cannot provide third-party assurance required for legal standing or Cyber Essentials Plus. Estimated cost: £3,000–£8,000 for a scoped 2–3 day web-app test.
2. **Legal review of Terms of Service and Privacy Policy** before public launch (EU GDPR, UK DPA 2018).
3. **Redis-backed rate limiting** before horizontal scaling or high-traffic launch.

SHORT TERM — NEXT SPRINT

4. Add `more_set_headers "Server: "` to nginx config to suppress the `nginx` product token from response headers.

5 Verify NextAuth sets session cookie `SameSite=Lax` at minimum; audit entire `Set-Cookie` flags.

6 Implement recovery/backup codes for MFA (on roadmap) to prevent account lockout on device loss.

LONG TERM — ROADMAP

7 Migrate to SRP (Secure Remote Password) to eliminate master password transit.

8 Add a random server-side KDF salt to eliminate KDF precomputation from salt-knowledge.

9 Implement active session management (list + revoke individual sessions).

Conclusion

The SealedKeys application demonstrates a strong security foundation for a zero-knowledge vault product. The cryptographic core — PBKDF2 → AES-256-GCM client-side, non-extractable CryptoKeys, JWE-encrypted sessions — is correctly implemented. The authentication subsystem (TOTP replay protection via `UsedTotpCode`, session versioning, nonce-based CSP, CORS allowlisting, layered rate limiting) is production-quality.

All exploitable vulnerabilities identified during this session were remediated before the runtime test phase. The 42 runtime tests found no bypasses, IDORs, injection vectors, or access control failures.

The application is appropriate for **private beta** with technical teams under a responsible disclosure programme. **Public launch and payment processing** should be preceded by an independent CREST-certified assessment.