Hackage security

Known issues and opportunities

Fraser Tweedale
@hackuador@functional.cafe

2026-06-04

Rules

- I will discuss open security issues (non-critical)
- Not to be shared except with relevant maintainers
- *Opportunities* are OK to share!
- Want to help and need more info? Talk to me or Gautier or email `security-advisories@haskell.org`

The good news

- Several **CRITICAL** XSS and CSRF vulnerabilities were fixed over last 2 years
- We are hopeful no more catastrophic issues of this kind lurking.

Issue 1: HSEC-2026-0001 - CAPTCHA bypass

- CAPTCHA bypass due to client-forgeable hash and timestamp.
  Impact: LOW
- **To fix:** store CAPTCHA value and expiry time server-side, or propagate *signed* (digest,timestamp) via client and validate.
  Complexity: MEDIUM

Issue 2: HSEC-2026-0003 - missing authz in `putAliasEdit`

- Missing authorisation check allows unauthenticated clients to create or modify global tag alias mappings.
  Impact: MEDIUM
- **To fix:** add authorisation check.
  Complexity: LOW

Issue 3: HSEC-2026-0005 - sensitive info in scrubbed backup

- `userDetailsToCSV` does not scrub the "notes" field.
  Impact: LOW (NONE?)
- **To fix:** redact except when `FullBackup`
  Complexity: LOW

Issue 4: HSEC-2024-0005 - insecure password storage

- HTTP `Digest` authn mechanism requires insecure password storage. Attacker with DB access can compromise any user account.
  Impact: HIGH
- **To fix:** switch to Basic (or web form) authn + re-hash passwords upon login (e.g. Argon2) + **comms** & timeline to disable un-migrated accounts
  Complexity: HIGH
- Compatibility changes in *cabal-install* already released

Opportunity 1: HTML login form + cookie-based session

- HTTP authn mechanisms (Basic, Digest) are awkward, don't support MFA, don't support session timeout
- Ability to force-expire all sessions is a useful Incident Response capability
- **To implement:** web form + set cookie (signed?) + validate and reinstate session context
  Complexity: MEDIUM
- https://github.com/haskell/hackage-server/issues/91

Opportunity 2: MFA - TOTP

- **To implement:** Schema to store shared secret + QR code enrolment workflow + Login UX
  Complexity: MEDIUM
- **Policy questions:**
  - When to enforce? Upon whom?
  - Staged rollout: opt-in at first, then require for some accounts based on {metrics,vibes}, then all?
- https://github.com/haskell/hackage-server/issues/1265

Opportunity 3: passkey / WebAuthn support

- Modern phishing-resistant 2FA using crypto token
- Lower priority than TOTP
- **To implement:** ? I haven't looked deeply into it.
  Complexity: MEDIUM?

Opportunity 4: integrate advisory DB info

- Show vulnerable versions and show (or link to) vulnerability info
- Show info about (potentially) vulnerable deps
- **To implement:** periodically retrieve advisory DB *snapshot*; load data on package pages; update HTML template(s) to display
  Complexity: HIGH

Opportunity 5: scoped tokens

- API tokens with restricted permissions or scope
- **To implement:** store associated scope in backend, **OR** issue signed token with scope metadata
  Complexity: MEDIUM
- https://github.com/haskell/hackage-server/issues/1378

Opportunity 6: trusted publishing

- Grant short-lived publish token to trusted pipeline/workflow on trusted forges (GHA, etc)
- **To implement:** OpenID Connect workflow + scoped token issuance
  Complexity: HIGH
- https://github.com/haskell/hackage-server/issues/1443
- Read more:
  https://repos.openssf.org/trusted-publishers-for-all-package-repositories

Opportunity 7: report security issues

- Wanted: a better way to report security issues in a particular package
- **To implement:** Web form? Email SRT? Open to ideas...
  Complexity: MEDIUM?

Hackage modernisation proposal

- https://github.com/haskellfoundation/tech-proposals/pull/67
- Re-implementation of *hackage-server* API with PostgreSQL storage
- Defer the larger "opportunity" work until after cut-over?
  (expected timeline: 3 months)

Rallying cry

- Who wants to help?
- What should we try to do this week(end)?
- Prioritise and focus