The Haskell Security Response Team

Current status and future evolution

Fraser Tweedale
@hackuador@functional.cafe

2024-06-06

Outline

- SRT: why / what / who
- Advisory database status
- Other high-priority work
- Evolving the SRT
- ZuriHac goals / ideas

SRT: why / what / who

SRT - motivation

- HF tech proposal 37: advisory database
- Primary motivation: enterprise adoption
- Without a security response structure and artifacts, Haskell is a **non-starter** for many companies
- Makes Haskell an easier choice even without hard regulatory/compliance requirements
- We should care about security of our ecosystem anyway :)

SRT - scope

- Manage the advisory database and associated tooling
- Triage, assess and admit issue reports
- Coordinate repsonse with maintainers of affected packages (high-impact issues)
- Collaborate and respond to needs of downstream tools that consume our advisories
- Quarterly report

SRT - current members

- Fraser Tweedale
- Gautier Di Folco
- Mihai Maruseac
- Tristan de Cacqueray
- Casey Mattingly
- Jose Calderon (observer/overseer)

Advisory database

Advisory database - structure

- https://github.com/haskell/security-advisories
- TOML metadata + CommonMark description
- arranged by namespace and package/component name
- symlinks for multi-package advisories
- quirk: dates are derived from Git commit times

```toml
[advisory]
id = "HSEC-2023-0001"
cwe = [328, 400]
keywords = ["json", "dos", "historical"]
aliases = ["CVE-2022-3433"]

[[affected]]
package = "aeson"
cvss = "CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H"

[[affected.versions]]
introduced = "0.4.0.0"
fixed = "2.0.1.0"
```

Hash flooding vulnerability in aeson

aeson was vulnerable to hash flooding (a.k.a. hash DoS). The
issue is a consequence of the HashMap implementation from
unordered-containers. It results in a denial of service through
CPU consumption. This technique has been used in real-world attacks
against a variety of languages, libraries and frameworks over the
years.

Advisory database - outputs

- OSV (ingested by osv.dev[1])
- HTML index: https://haskell.github.io/security-advisories/
- "snapshot" format designed for syncing with tools[2]

[1] https://osv.dev/list?ecosystem=Hackage

[2] https://github.com/haskell/security-advisories/pull/179

Advisory database - libraries

Libraries and tools for processing advisory data are on Hackage:

- https://hackage.haskell.org/package/cvss
- https://hackage.haskell.org/package/osv
- https://hackage.haskell.org/package/hsec-core
- https://hackage.haskell.org/package/hsec-tools

Expect churn as more consumers/users arrive, give feedback.

CWE[3] library is coming.

[3] *Common Weakness Enumeration*—https://cwe.mitre.org/

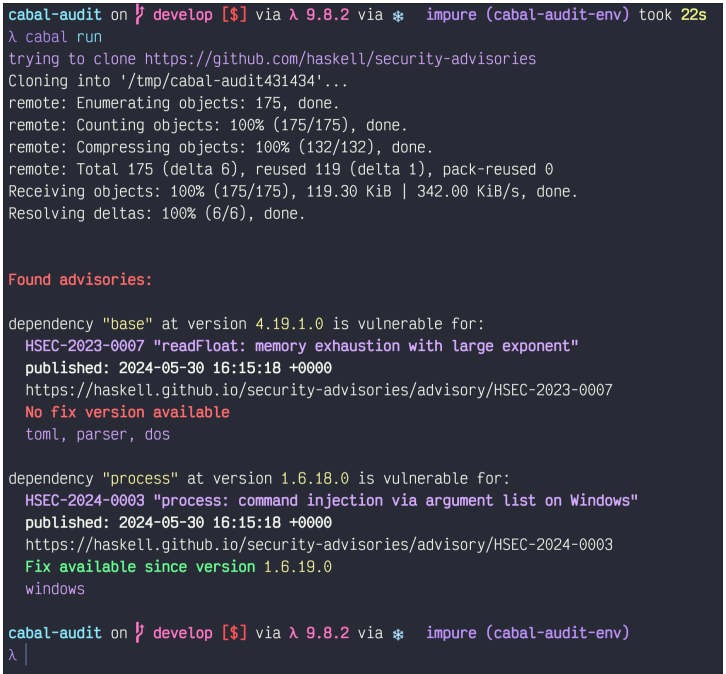
Advisory database - low growth

- SRT advisory triage/assess/add workload is **very low**. Why?
- Submission process is too hard? (improve the tools)
- Haskell just has fewer security bugs? (not significantly, IMO)
- People don't know about it? (increase visibility)
- People don't care?

What is missing?

cabal-audit

- Scans the build plan for vulnerable dependencies
- The author, **MangoIV**, is here!
- Long-term goal: integrated with `cabal-install` (as plugin)
  - Ship via GHCUp?
- https://github.com/MangoIV/cabal-audit


cabal-audit on develop [$] via λ 9.8.2 via ❄ impure (cabal-audit-env) took 22s
λ cabal run
trying to clone https://github.com/haskell/security-advisories
Cloning into '/tmp/cabal-audit431434'...
remote: Enumerating objects: 175, done.
remote: Counting objects: 100% (175/175), done.
remote: Compressing objects: 100% (132/132), done.
remote: Total 175 (delta 6), reused 119 (delta 1), pack-reused 0
Receiving objects: 100% (175/175), 119.30 KiB | 342.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.
Found advisories:
dependency "base" at version 4.19.1.0 is vulnerable for:
HSEC-2023-0007 "readFloat: memory exhaustion with large exponent"
published: 2024-05-30 16:15:18 +0000
https://haskell.github.io/security-advisories/advisory/HSEC-2023-0007
No fix version available
toml, parser, dos
dependency "process" at version 1.6.18.0 is vulnerable for:
HSEC-2024-0003 "process: command injection via argument list on Windows"
published: 2024-05-30 16:15:18 +0000
https://haskell.github.io/security-advisories/advisory/HSEC-2024-0003
Fix available since version 1.6.19.0
windows
cabal-audit on develop [$] via λ 9.8.2 via ❄ impure (cabal-audit-env)
λ

Hackage - integrate with advisory db

- Show info about vulnerable versions
- Show info about (potentially) vulnerable deps
- Add "how to report" info / helpers
- Flora.pm might want to do these things too

Hackage - other security enhancements

- password / token storage improvements
- 2FA (TOTP for a start)

SRT tooling

- **GitHub bot** to help define / review advisories
  - explain CVSS, CWE values; suggest keywords; etc
- **Web form** for advisory submission
  - . . . or other ways to make it easier

Exploitability information

- As audit tooling matures, we must suppress false positives
  - e.g. HSEC-2023-0007 `readFloat` memory exhaustion in ***base***
- VEX - *Vulnerability Exploitability eXchange*
  - statements that an issue is(n't) exploitable in the dependent
  - **data model** by CISA.gov
  - implementations: OpenVEX, SPDX 3.0, OASIS CSAF 2.0, CycloneDX
- We don't have to use VEX *per se*

VEX statements - generation

- Written by human
- Generated by machine (call analysis)
  - Tristan's experiment: https://github.com/TristanCacqueray/cabal-audit
  - typeclass methods seem to be the tricky part
  - relies on declaration of affected functions/symbols in the advisory

VEX statements - distribution

- Cabal package description (**dependent**)
  - supplied by maintainer, or Hackage trustees
  - distributed in Hackage snapshots
  - metadata revisions $\rightarrow$ new version not required to update VEX statements
- Advisory DB as a VEX clearing-house
  - supplied by SRT, or community with SRT review
  - distributed in Advisory DB snapshots
- Both?
  - requires conflict resolution (preferred source)

Dependency graph analysis

- Tools to analyse the dependency graph (of a single project or whole ecosystem) are increasingly important
- That xkcd[4]
- Identify the **load-bearing projects / juicy targets**?
  - Are they maintained? Sustainably?
  - What are the main risks?
- Query projects exposed to **external risks**
  - cbits? vendored/bundled code? out of date?
  - using external libraries?

[4] https://xkcd.com/2347/

Dependency graph analysis - Open Source Insights

- A fair bit of this tooling exists in *Open Source Insights*
  - https://deps.dev; Google project
  - Web, visualisations, API, BigQuery
- Haskell is not supported yet
- Development is not public (currently)
- I have reached out to find out more and offer support
- *acme-everything* becomes useful?

But wait there's more...

- SBOM artifacts? (e.g. SPDX)
- Add Haskell call analysis support to osv-scanner?
- Increase issue discovery efforts
  - OSS-Fuzz support?
- OpenSSF Best Practices checking?
- **Verified crypto** libs
  - ...and other compliance things that matter in various sectors

Software Bill of Materials (SBOM)

- Blog post published yesterday:
  - *SBOM Generation and Vulnerability Monitoring for the Hackage/Haskell Ecosystem*[5]—Timesys (cybersecurity vendor)
  - Cabal freeze file → CycloneDX SBOM via *syft*[6]
- Build SBOM generation into `cabal-install`? As plugin?

[5] https://www.timesys.com/security/sbom-generation-and-vulnerability-monitoring-for-the-hackage-haskell-ecosystem/

[6] https://github.com/anchore/syft

Evolution of the SRT

SRT scope change

- Workload w.r.t. current charter is low
- Much to do that's *not* in the charter
- Therefore: **expand the charter** and grow the team
- Gather feedback/ideas this week about:
  - how scope of SRT should change
  - topology (sub-teams? separate efforts? how many people?)

SRT nominations

- Soon: call for nominations for new SRT members
- Number of people and responsibilities depends on feedback
- Casey is retiring from SRT. **Thank you** for your efforts!

Ecosystem Workshop / ZuriHac goals

Ecosystem Workshop / ZuriHac goals

- Collaborate with anyone on anything that improves Haskell security posture
- Discussions about SRT evolution
- I will *default* to working on Hackage
- Gautier will work on advisory db snapshots
- MangoIV will work on cabal-audit
- Beginner-friendly tasks:
  - *cvss*/*osv*/*hsec-core* CVSS 4.0 support
  - Haskell.org security page - https://github.com/haskell-infra/www.haskell.org/issues/293

SRT code github.com/haskell/security-advisories

My blog frasertweedale.github.io/blog-fp

Fediverse @hackuador@functional.cafe