Package URL and Version Range spec

Towards (mostly) universal dependency resolution
Package URL
and Version Range spec

Towards (mostly) universal dependency resolution
Philippe Ombredanne

▷ ScanCode and AboutCode projects lead and maintainer
▷ Creator of **Package URL**, co-founder of **SPDX**, ClearlyDefined
▷ FOSS veteran, long time **Google Summer of Code** mentor
▷ Co-founder and CTO of nexB Inc., makes of DejaCode
▷ Weird facts and claims to fame
  • Signed off on the **largest deletion of lines of code** in the **Linux kernel** (but these were only comments)
  • Unrepentant **code hoarder**. Had 60,000+ GH forks now down only to 20K forks
▷ pombredanne@gmail.com irc:pombreda
Things are getting hairy and complex!!

▷ Ever more FOSS packages are reused
  • 10x to 100x more than a few years ago
  • Yes! we can really build applications from components!

▷ Complex stacks with multiple tech and languages
  • Deep dependency trees
  • Dependencies on both application and system packages

▷ Unstated dependencies across
  • package ecosystem boundaries
  • system and application boundaries

▷ More bugs and vulnerabilities!
How can we deal with this complexity?

▷ Hand crafted, good old README with installation instructions
  • Install these debian packages, these npm packages and these Python packages

▷ Replace all package managers with one to rule them all
  Gain total control, all the way down with ...

▷ ... massive mono repo and "hermetic" build system
  • Big tech use these with Bazel or Buck

▷ ... general purpose package managers
  • Like Spack or Conda

▷ ... "functional" package managers
  • Like nix or guix
What is dependency resolution?

- Start with **package** you directly depend on
  - name & version (or version range)
- Access some **package** index or metadata source
  - collect all known versions of the package
- Select one **version** that matches the range you need
- For this version, collect the packages it **depends** on
  - names and versions (or version range)
- **Repeat recursively**
- Update selected **names and versions** as needed until consistent
  - Can be involved algos using sat solvers, backtracking, etc.
- **Finally install these packages (not today's scope)**
The many curious ways of versioning

▷ Resolving package dependencies
  • "I require package <ABC>, version 2.0 or later versions"

▷ Affected vulnerable versions
  • "vulnerability CVE-2021-1 affects <XYZ>, version 3.1 and version 4.2 but not version 5"

▷ Version numbers should be boring

▷ Yet each ecosystem has its own way for version and range!
  Debian, RPM, npm, PyPI, Ruby, etc. have their different notations using comparators >, < or =, or tilde ~ or caret ^ or star *

▷ Each resolve a dependency version in a range differently
What if ....

▷ We could express a dependency in a *mostly universal* way?
▷ And not replace all package managers BUT rather rule them all

▷ Use **Package URL** "purl" to name a package across ecosystem
▷ Add new "**vers**" **Version Range Spec** for ranges
  • For any ecosystem, building on Package URL "package type"
  • Simplified comparators set: >, <, =, !, <=, >=
  • Ecosystem-specific version comparison
▷ Designed for **dependency** ranges AND **vulnerability** ranges
We need new standards to rule them all!

**How Standards Proliferate:**

(See: A/C chargers, character encodings, instant messaging, etc)

**Situation:**

There are 14 competing standards.

**14?! Ridiculous!**

We need to develop one universal standard that covers everyone's use cases. **Yeah!**

**Soon:**

There are 15 competing standards.

Credits: https://xkcd.com/927/

xkcd.com is best viewed with Netscape Navigator 4.0 or below on a Pentium 3±1 emulated in Javascript on an Apple IIICS at a screen resolution of 1024×1. Please enable your ad blockers, disable high-heat drying, and remove your device from Airplane Mode and set it to Boat Mode. For security reasons, please leave caps lock on while browsing.

This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. https://creativecommons.org/licenses/by-nc/2.5/
Package URL (purl)

- Problem: Each package type/ecosystem has its own conventions to identify, locate and provision software packages.
- Solution: An expressive **package-url string**, minimalist yet obvious.
- Identify & locate software packages reliably across tools and languages.
  
  pkg:npm/foobar@12.3.1
  
  pkg:pypi/django@1.11.1

- Started with ScanCode and VulnerableCode and now adopted in many places.
- Now a **de-facto standard** used in ORT, OSSF OSV, CycloneDX, SPDX, Sonatype OSSIndex, GitHub and many places.
- Libraries in Java (multiple), PHP, Go, Python, JavaScript, Ruby, Swift, Rust, .Net,
- Recommended by the US NTIA as an SBOM package identifier.
- See https://github.com/package-url/purl-spec
Package URL in the news this week

"Component verification and vulnerability reporting are supported by some SBOM data formats today. Globally unique identifiers is a work in process supported by the leading data formats for package URLs (PURLs)."

Software Bill of Materials (SBOM) and Cybersecurity Readiness
January 2022
Stephen Hendrick, VP Research, The Linux Foundation
Version Range Spec (vers)

▷ Problem: Each package type/ecosystem has its own convention to specify version ranges
▷ Solution: An expressive version range string, minimalist yet obvious
▷ Specify version ranges reliably across tools and languages for deps and vulnerabilities.

\texttt{vers:npn/1.2.3|>=2.0.0|<5.0.0}
\texttt{vers:pypv/0.0.1|0.0.2|0.0.3|1.0|2.0pre1}

▷ A version range specifier ("vers") is a URI string using the vers scheme and this syntax:
\texttt{vers:<versioning-scheme>/<version-constraint>|<version-constraint>|...}

▷ Started with VulnerableCode with "univers" library and now used in CycloneDX
  • Goal is to be a useful adjunct to purl

▷ Can pave the way to universal dependency resolution engines
  • Would still need to have access to all the package versions... working on it!

▷ See https://github.com/package-url/purl-spec/blob/version-range-spec/VERSION-RANGE-SPEC.rst
Putting it all together

A mostly universal package name (**purl**) with a mostly universal version range (**vers**) opens up many possibilities:

▷ **Store vulnerable version ranges and evaluate later if a version is vulnerable**
  • Prototype in univers library and VulnerableCode
  • https://github.com/nexb/vulnerablecode
  • https://github.com/nexB/univers

▷ **Build a multi package installer for many ecosystems**
  • Prototype toy at https://gist.github.com/pombredanne/d3585617882f91d9316be5ce5edd190

▷ **Write mostly universal dependency declarations**
  • Soon in https://github.com/nexB/scancode-toolkit

▷ **Write a mostly universal dependency resolver?**
  • Say goodbye to README installation instructions!

An incremental approach instead of replacing everything
SCA AUTOMATION IS HARD

▷ But it is nearly **impossible** if no one speaks the same language
▷ To de-babelize this, **we need** shared names for:
  • Licenses
  • Packages
  • Versions
  • Vulnerabilities
  • Version control references
The Naming of Cats is a difficult matter

- **License names**
  - Mostly solved with **SPDX license ids** and **expressions**
  - Plus scancode-licensedb DB of most FOSS licenses

- **Software package names**
  - Mostly solved with **Package URL** emerging as a de-facto standard

- **Version range notation** for dependencies and vulnerable ranges
  - New mini spec for **Version Range Specifiers**

- **Vulnerability identifiers**
  - Mostly solved with NVD's **CVE** and aliases

- **Version control system references**
  - Likely solved with **VCS URLs** adapted from Python pip, now in SPDX
If you want to help

You can contribute code, time, docs (or cash?)
▷ Use these fine FOSS tools and specs
  ● https://github.com/package-url
  ● https://www.aboutcode.org/
  ● https://github.com/nexB/
▷ Join the conversation at
  ● https://gitter.im/aboutcode-org
▷ Donate at
  ● https://opencollective.com/aboutcode
Credits

Special thanks to all the people who made and released these excellent free resources:

▷ Presentation template by SlidesCarnival
▷ Photographs by Unsplash
▷ All the open source software authors that made ScanCode and AboutCode possible