Standardizing FOSS package identifiers using Package URL

Towards (mostly) universal SCA tools integration

Philippe Ombredanne, AboutCode.org nexB Inc.

Philippe Ombredanne

- ▷ **ScanCode, VulnerableCode and AboutCode 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., makers 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 license comments)
  - Unrepentant **code hoarder**. Had 60,000+ GH forks now down only to 20K forks
- ▷ pombredanne@gmail.com irc:pombreda

Software things are getting hairy and complex!!

- ▷ Ever **more FOSS packages** are reused
  - *10x to 100x more than a few years ago*
  - ***But YEAH!*** *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!**

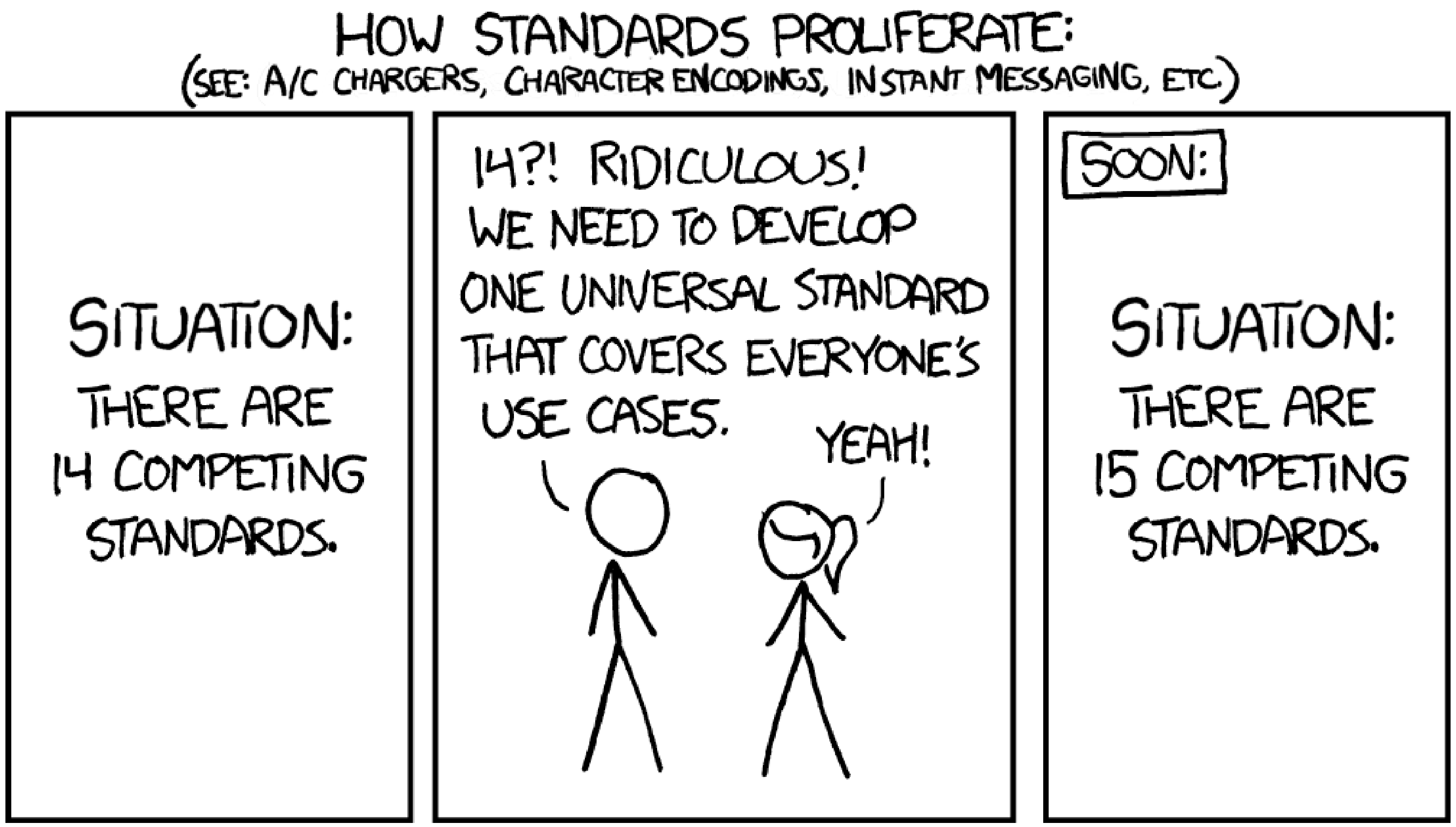
But wait! SBOMs anyone?

- ▷ Emerging imperative for appsec
  - ● Convey the "making of" a software system or app
- ▷ Central principle: track inventory of packages or components
- ▷ So you **must to identify software packages used**
- ▷ ... and their license (SPDX license expression!)
- ▷ ... and known security bugs (CVEs)

What if ....

- ▷ We could name a package j**ust by looking at it?**
  - ● It's true name, not a "given" name.
- ▷ Make it so that the name is obvious for human and machines?
- ▷ Use this to id packages in SCA, vulnerability and dependency management in **a mostly universal** way?
- ▷ And not replace all package managers BUT rather rule them all

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:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

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

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

Some PURL history

- ▷ Started to solve simple problems for ScanCode and VulnerableCode in 2017
  - How can we identify all packages a the simplest way?
  - mish mash of Maven, npm, python, Ruby
- ▷ Key insight
  - Each package ecosystem guarantees each name is unique
  - The essence of ids is a name + a version (plus some extra)
  - file 5.3 in npm is not file 5.3 in Rubygems
  - **`npm/file@5.3`** vs. **`gem/file@5.3`** removes the ambiguity
  - **`pkg:`** prefix makes it a valid URL: **`npm/file@5.3`**

Some PURL history

- ▷ Many other projects were facing similar problems
  - ORT, Openshift, Google grafeas, Libraries.io
  - Borrowed concept from a similar approach at JFrog
- ▷ Eventually we moved the spec to its own org
  - Invited as co-org owners to share control
- ▷ Contributions of programming language parsers poured in
- ▷ Adoption started quickly across many open source tools
- ▷ Also in vulnerability databases

Who is using PURL?

- ▷ **Open source SCA tools**
  - ScanCode, MatchCode, Tern, ORT, Syft, Fosslight, Anchore, Microsoft SBOM tool, DependencyTrack, etc.
  - Most other FOSS and proprietary SCA and Infosec/Appsec tools
- ▷ **SBOM and VEX specs**
  - CycloneDX
  - SPDX
  - CSAF, Open VEX
- ▷ **Mostly all SBOM Tools**
  - GitHub SPDX SBOM
- ▷ **Vulnerability databases**
  - Google OSV
  - Sonatype OSSIndex
  - VulnerableCode
  - GitHub Advisory DB
  - Global Security DB
  - NVD in v5.1!
- ▷ **Databases of packages metadata**
  - PurlDB, Ecosystems, Osselot

SBOM anyone?

- SBOMs are everywhere
  - GitHub can even create these directly from a repo
  - But what about data quality (depth and breadth)?
  - But what about using proper machine readable identifiers (license, PURL)?
- Hi-Fi or Lo-Fi SBOMs?
- Every tool creates SBOMs but then what?
  - 2 out of 50+ folks were effectively consuming SBOMs
- Big gaps in tool-to-tool integration
- Too much over engineering, and under-specification
- Advice: Ignore the SPDX vs. CycloneDX feud and **embrace both, with PURL**
  - SBOM is just a reporting format
  - PURL is the key unifying id between them

Package URL in the news

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

https://linuxfoundation.org/wp-content/uploads/LFResearch_SBOM_Report_final.pdf

Software Bill of Materials (SBOM) and Cybersecurity Readiness

January 2022

Stephen Hendrick, VP Research, The Linux Foundation

PURL is the essential glue

PURL is emerging as the glue to avoid lock-in!

- Key vector for interop as a universal id
  - if two tools speak PURL, integration is made easier
- **You must demand** its adoption by your vendors and projects
- The benefits are
  - less lock-in
  - mix and match best in class tool
  - can compare performance of tools objectively

Package URL quote

"Package URLs have profoundly transformed the landscape of appsec and infosec tooling, for the better."

A leader for an SBOM specs, March 2023

In conclusion: 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

T.S. Eliot: 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** in purl project
- ▷ **Vulnerability** identifiers
  - Mostly solved with NVD's **CVE** and their many aliases
- ▷ **Version control system** references
  - Likely solved with **VCS URLs** adapted from Python pip, now in SPDX

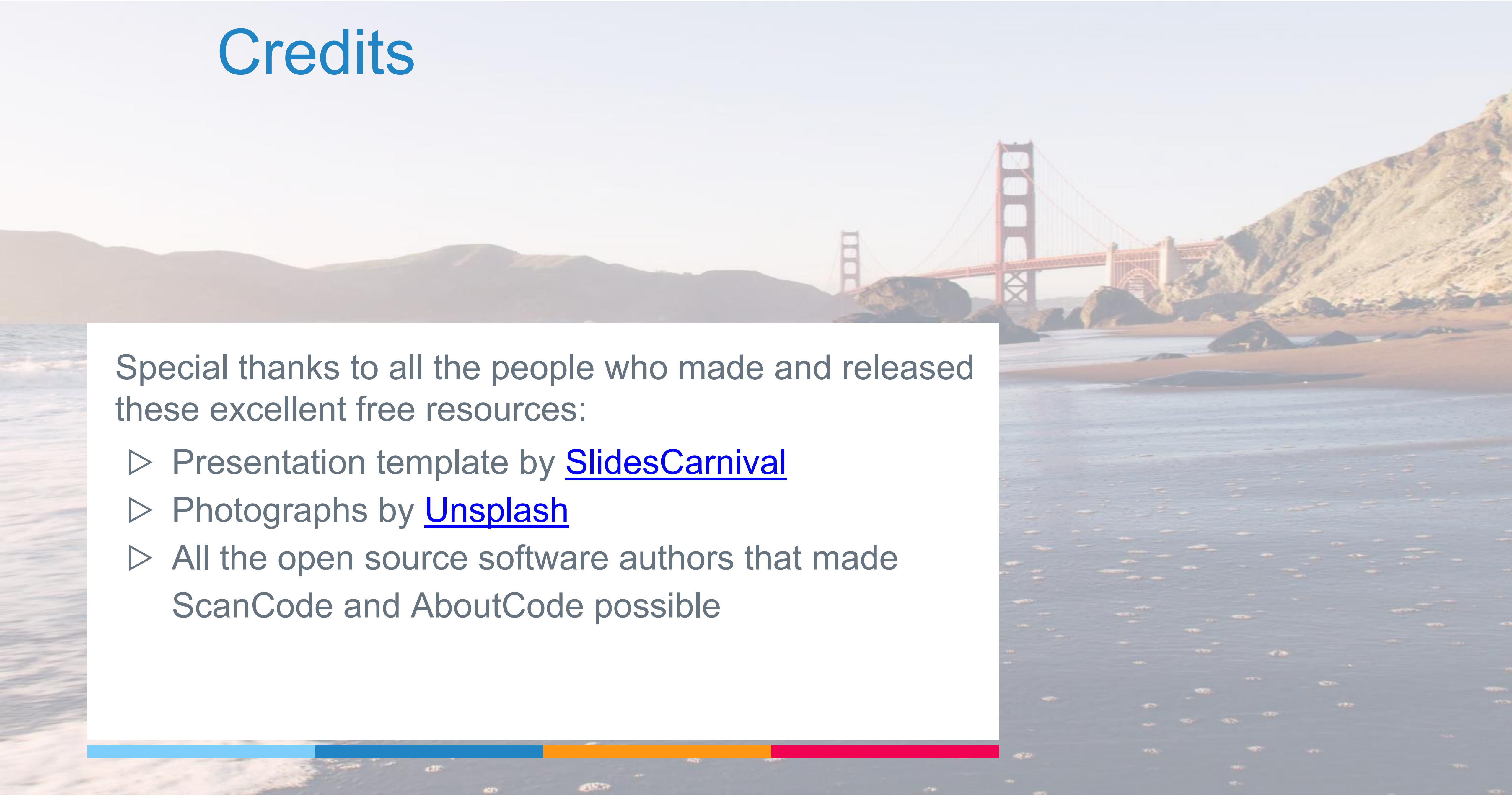
Credit https://www.severnedgevets.co.uk/pets/advice/advice-new-kitten-owners
Credit: T.S. Eliot, "Old Possum's Book of Practical Cats"

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

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.

```
vers:npm/1.2.3|>=2.0.0|<5.0.0
vers:pypi/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:
- ▷ `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