VulnerableCode

Because a vulnerability database should not be about Vulnerabilities

Philippe Ombredanne

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

Agenda

- ▷ The state of vulnerability databases (open or not)
- ▷ How do we search vulnerabilities? By package first!
- ▷ A better approach: package first
- ▷ Why VulnerableCode?
- ▷ VulnerableCode Solution
- ▷ How to create a package vulnerability database
  - ○ Aggregate and correlate many data sources
  - ○ Multi level data refinement
- ▷ Issues with vulnerability data
- ▷ Future plans
- ▷ Next steps: we need your help!

State of vulnerability databases (1)

- Databases with ghost **packages**
  - DBs reference packages that do not exist anywhere
- Databases ghost **vulnerable versions**
  - Even though these are not vulnerable or the opposite
- Databases "Crying wolf" - **improbable** vulnerabilities
  - DBs report a package as vulnerable if anything in the dependency tree may be vulnerable (Log4j)
- Impossible, self-contradictory version ranges
  - Resolved to nothing or everything
- Redundant and noisy duplicated vulnerabilities
- Vulnerabilities mapped to hard-to-find CPEs

State of vulnerability databases (2)
The Telephone Game problem

- Everyone is making something up a little by trying to improve data
- Each of them makes something up slightly differently
  - Too much reliance on automated tools on top of bad data
- Many DBs base their content on another DB’s content
  - At each step the data is transformed (and damaged) in subtle ways
- You can have as many vulnerable ranges as there are DB interpretations.
  - None of them is entirely faithful to the upstream data
  - Over time this turns into The Telephone game
- **Upstream has better data**

The true-true vulnerabilities are **UPSTREAM!**

Credit https://live.staticflickr.com/3798/10142017736_7f69d9f472_h.jpg
"Brown Bears at Brooks Camp, Katmai National Park"
by Christoph Strässler https://www.flickr.com/photos/christoph_straessler/
This work is licensed under a Creative Commons Attribution-2.50 License.
https://creativecommons.org/licenses/by-sa/2.0/
https://www.youtube.com/watch?v=V4kWQSpCbvo

State of vulnerability databases (3)

- ▷ Databases of known FOSS software vulnerabilities are mostly **proprietary** and/or **privately** maintained using proprietary tools, processes and data.
- ▷ Why not open data? FOSS code likes open data about FOSS!
  - ○ Some new entrants are now using open licenses: 🎉
  - ○ GHSA (GitHub), OSV (Google), GitLab (one month delay)
- ▷ Emerging support for **Package URL** promotes interoperability
  - ○ OSSF OSV, Sonatype OSSINDEX
- ▷ Emerging common format promotes interoperability
  - ○ **OSSF OSV**
- ▷ But open formats do not mean common data identifiers

How do we search? By package first!

Questions to answer:

- ▷ Is package foo@1.0 known to be vulnerable?
  - ○ What are the vulnerabilities?
  - ○ What is the severity of the vulnerability?
  - ○ Which version has a fix?
- ▷ More rarely: do I have any package vulnerable to this vulnerability?

A better approach: package first

~~Lookup **vulnerabilities**, find packages~~

Find **packages**, lookup vulnerabilities

Why VulnerableCode? accuracy and correctness

- Vulnerabilities are important!
- Code is more important! **Package first**!
- There is no Free Software Vulnerability Database that is
  - Open!!
  - Comprehensive for most ecosystems (system+package)
  - Curated by expert humans
  - Validated: Trusted, correlated and verified data
  - **Working towards correctness**

VulnerableCode Solution

- ▷ Find packages with scanning, matching and tracing
  - ○ Leverage all tools that report package-url (we support CPE too)
  - ○ **ScanCode**.io and **ScanCode**, ORT, Tern, OWASP Dependency Track and many more .... or an **SBOM** (SPDX, CycloneDX)
- ▷ Lookup package vulnerabilities in an open database that aggregates them all
- ▷ Query by **purl**!
- ▷ Open data and open source tools are better for open source!
- ▷ Eventually review by experts to curate all the data.

How to create a package vulnerability database?

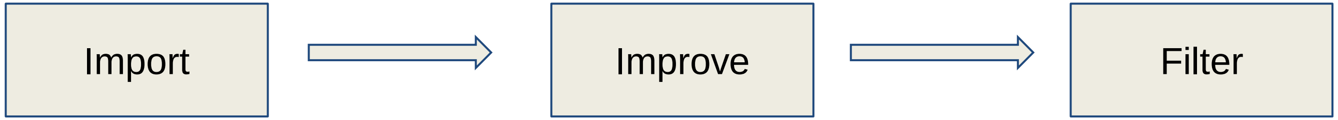
- Use data from upstream, at the **source of the source**!
  - From the package maintainers and authors themselves
- Employ a **confidence** based system: not all data are equally trusted and of the same quality
- Aggregate and correlate many data sources to enrich, cross-check and validate
- Discover of new relations between vulnerabilities and packages from mining the graph
- Curate and review for correctness with experts (AI is nice but does not fix the problems)

Aggregate and correlate many data sources

- Collect and parse **many sources**
  - Store in a common data model
  - Cross-reference to create a graph
- **Project-specific trackers**
  - Apache, OpenSSL, nginx...
  - Bug trackers, commit logs, projects CHANGELOGs.
- **Linux distro trackers** (Debian, Ubuntu, RedHat, SUSE, Gentoo, ...)
  - Custom or standard formats (CVRF, OVAL)
- **Application package trackers**
  - NuGet, Rust, RubyGems, Pysec, RustSec, npm, ...
- **NVD**, and other aggregators: **OSV, GHSA, GitLab, GSD and more**.

Multi level data refinement

- The data is always imported in an “**Advisory**” staging area
- "Advisory" data are converted to "**Vulnerability**" and "**Package**" data and their relationships using "**Improvers**"
- "Advisory" data that cannot be converted are kept with a log to investigate and resolve issue
- Specific improvers can mine the graph, cross check with other data sources, resolve updated version ranges



Issue: Ghost packages

- **Some packages do not exist anywhere**
  - Including versions that may not exist: they were never released

- **Solution: Look up upstream in the package registries and repositories**
  - We can look up in the registries and repositories to validate that the Package URLs and versions are correct and really exist upstream

Issue: Lesser data quality

- **Some vulnerability sources cannot be trusted**
  - Known to make incorrect or inaccurate assertions about packages and versions
- **Solution: store confidence level**
  - Confidence level ensure we keep all inferred data, even lesser quality data
  - We do not trust others: We can discount the data sources we trust less
  - And we do not trust ourselves: we can discount the automated inferences we do if we are not 100% sure about their correctness

Issue: Incorrect or missing versions

- **Some package versions are missing or incorrect**
  - Affected version statements are often ambiguous
  - ***"All the versions of package foo are vulnerable to CVE XZY"*** really means all versions of foo known at the time this advisory was published were vulnerable.
- **Solution: store version range, resolve range and "time travel"**
  - We store version ranges as a compact string (using new purl "vers" spec)
  - We expand and resolve ranges with "univers" version handling library
  - Package version can be re/checked for being in a vulnerable range as needed
    - In the past and in the future
  - Improvers can do "**time travel**" based on version publication dates and determine if a package version was vulnerable **in the past when published**.

Issue: Duplicated data

- **Some vulnerabilities are duplicates**
  - Leads to many noisy relationships and lesser correlation abilities
- **Solution: Introduce a new set of vulnerabilities id aliases**
  - Before, we tracked by CVE id; only if there was none we created a VULCOID id. (VulnerableCode ID for a vulnerability)
  - We now always use a **VULCOID** id and track many aliases (including a CVE id when available) for each vulnerability
  - Aliases are used for data reconciliation during the second step of "Improvers" meaning that we avoid a large number of duplicates
  - Improver jobs will further merge additional duplicates

Other issues

- ▷ **Many data sources - redundant, unstructured, messy, incomplete**
    - ○ We grew to appreciate the complexity of the task and why commercial vendors currently dominate the space
    - ○ **Solution**: integrate them all (all the data sources) to cross-check them
- ▷ **Old, obsolete, or less useful data**
    - ○ More is not always better - e.g. old vulnerabilities on Windows 95
    - ○ Commercial-only software (Windows, etc.) or hardware is less interesting
    - ○ **Solution**: let go of some of the past! and ignore the legacy

Future plans

- More **primary data sources**, going upstream
- **More data**: Actual commits fixing and introducing vulnerabilities
- YARA Rules: enable finer grain detection of actual vulnerable code
- Community peer **curation system** including curation UI
- AI/ML for data quality improvements
- .... and good ole heuristics
- **VulnTotal**: Tool to compare all the Vulnerabilities DB
    - Think Virustotal for vulnerabilities

If you want to learn more about our projects

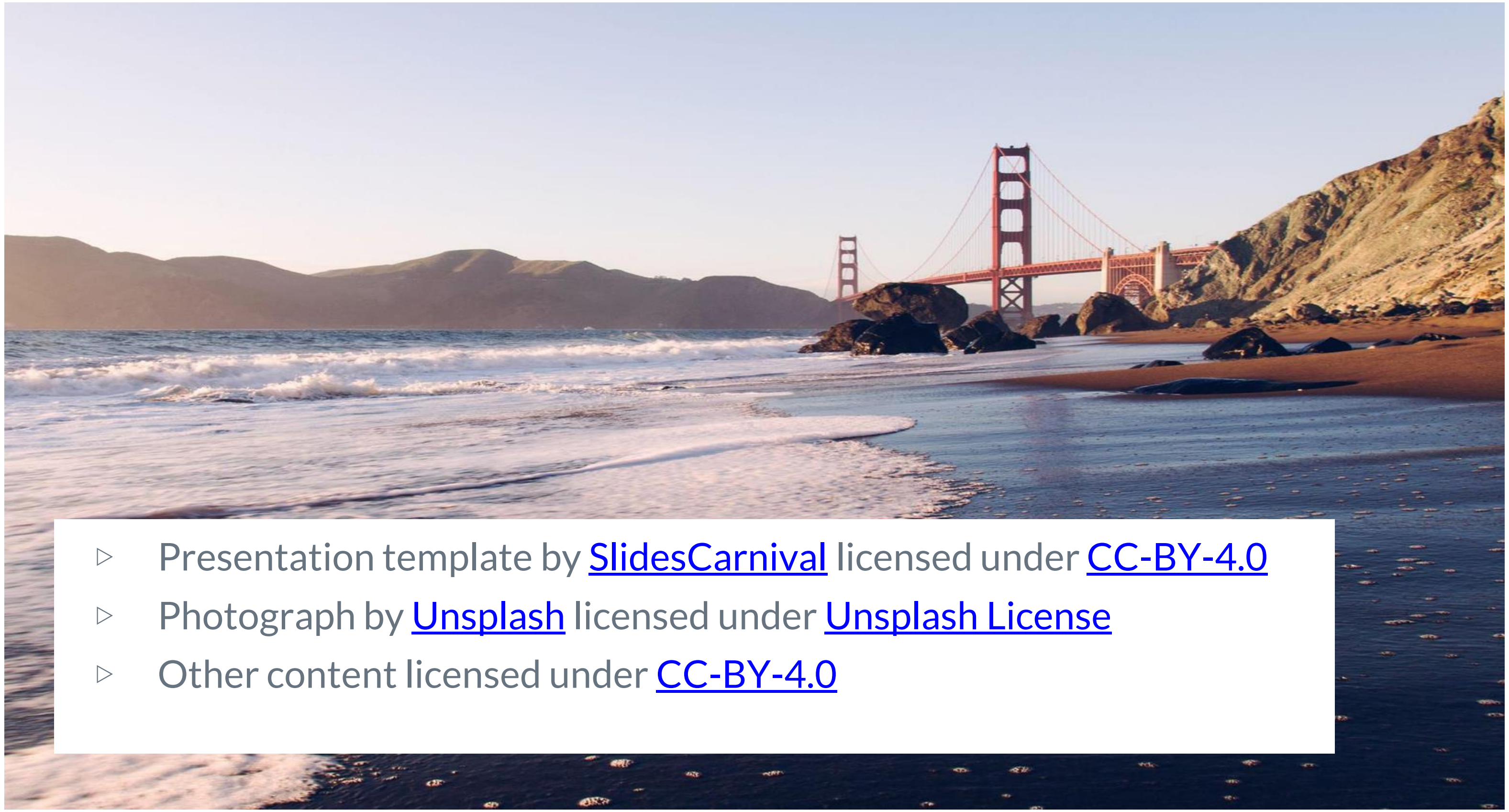
- ▷ Try out VulnerableCode with a free DejaCode account https://enterprise.dejacode.com/account/register/
- ▷ Register for our upcoming webinars https://nexb.com/webinars
- ▷ Read our latest blog post at https://nexb.com/vulnerablecode-public-release
- ▷ Download VulnerableCode at https://github.com/nexB/vulnerablecode/releases/latest
- ▷ Visit https://nexb.com/vulnerablecode for more information

If you want to help

You can contribute code, time, docs or funds

- Use these fine FOSS tools and specs
  - https://github.com/nexb/vulnerablecode
  - https://www.aboutcode.org/
  - https://github.com/nexB/
  - https://github.com/package-url
- Join the conversation at
  - https://gitter.im/aboutcode-org
- Donate at
  - https://opencollective.com/aboutcode

Credits



- Presentation template by SlidesCarnival licensed under CC-BY-4.0
- Photograph by Unsplash licensed under Unsplash License
- Other content licensed under CC-BY-4.0