Securing Open Source Supply Chains: FOSS for FOSS

Abstract

- Understanding both Software Bill of Materials (SBOM) and Software Composition Analysis (SCA) is essential for Software Supply Chain Security
- FOSS tools avoid vendor lock-in and also enable upstream projects to apply SCA for better supply chain security
- This presentation will cover:
  - Using SCA to find and report software licenses and vulnerabilities
  - Generating and consuming SBOMs with rapidly evolving regulatory and business requirements
  - Overview of FOSS tools like nexB's DejaCode, ScanCode and VulnerableCode to manage software supply chain risk

Agenda

- Background on Software Bills of Materials
- Software Composition Analysis Process
- Using FOSS SCA tools to create and manage SBOM data
- Securing the Open Source Software Supply Chain

NB: The primary focus of this discussion is Free and Open Source Software (FOSS) but most points also apply to to Proprietary Software. And most modern Proprietary Software contains FOSS - usually in the range of 80% or more depending on how you count.

Why trust nexB?

- Recognized by major companies as:
  - Trusted experts in Software Composition Analysis
  - Developers of best-in-class SCA tools
- FOSS-first mission: FOSS for FOSS
  - Our tools for FOSS SCA are open source
  - Focused on supporting the FOSS ecosystem
- nexB team members are thought leaders
  - Creators of ScanCode: https://www.aboutcode.org/projects/scancode.html
  - Creators of package-url: https://github.com/package-url
  - Co-founders of SPDX: https://spdx.org
  - Co-founders of ClearlyDefined: https://clearlydefined.io

Software Bill of Materials (SBOM)

- An SBOM is a list of software components used in a product
  - Concepts borrowed from discrete manufacturing
  - The list is typically a hierarchy (“graph”)
  - What is a software component? *There is no standard terminology!*
  - A component may be a file (source or binary) or a package of files
  - A package may be an archive with or without metadata
- Many possible SBOM use cases
  - Packaged software
  - Software deployed on a device
  - Software deployed on the Cloud
  - The Customer/recipient of an SBOM may be anywhere in the supply chain
  - Anyone who distributes software in any way will need to produce SBOMs

Why SBOMs

- An SBOM is a prerequisite for managing license and vulnerability risks from third-party software
- And for sharing that information across your supply chain
- Automation is essential to cope with the rapid and continuing increase in the volume of FOSS packages
- The entry point for managing these risks is agreeing somehow on the identification of the software units across a supply chain

Why SBOMs [2]

- Providing an SBOM with your software is now a requirement for doing business with US government agencies
  - The Cyber Supply Chain Management and Transparency Act of 2014 focused on vulnerabilities
  - The May 2021 Executive Order on Improving the Nation's Cybersecurity added the broader concept of software supply chain
  - CISA* currently has five weekly meetings on the topic!
- Modern software contains third-party software - FOSS or Proprietary - with potential licensing and vulnerability risks
- A better question: Why haven't we been using SBOMs before?

* CISA: Cybersecurity and Infrastructure Security Agency within DHS

SBOM Standards

Two emerging standards for an SBOM:

1. CycloneDX - https://cyclonedx.org/ - from OWASP
2. SPDX - https://spdx.dev/ - from the Linux Foundation
   - One weaker candidate: SWID - https://csrc.nist.gov/projects/Software-Identification-SWID

- Unlikely that there will be only one standard…
- And possible that there will be more than two.
- Remember: These are standards for data exchange, not design standards for any particular software system

SBOM Standards [2]

- Other standards will be required like Package URL to reliably identify a unit of software: https://github.com/package-url/purl-spec
- Waiting for a complete and final specification is not a realistic option
  - Best approach is to get started now
  - With an expectation that standards and tools will change
  - Just like the rest of the software domain

SBOMs for Software Supply Chain Security

- Software organizations can learn a lot from manufacturing best practices
- Each organization in a supply chain is responsible for knowing the origin and quality of the materials included in a product at their stage of production
- This requires knowing and sharing information in the format of SBOMs which means standardizing data and learning to translate among multiple standards

Software Composition Analysis

SCA is a set of processes and tools that cover:

- Identification – Identify distinct “units” of third-party software used in a product or project and their provenance
- Licensing – Determine the licensing for each software unit
- Security – Identify known security vulnerabilities for each software unit
- Quality – Evaluate the quality of a software unit based on software development data, such as number of bugs, fixes, etc.

Read SCA the FOSS Way for more information

Software Composition Analysis [2]

- Overall SCA needs to be a core competency for ay software development organization
- Embed in the software development workflow from design through release - as it is in manufacturing
- The choice of SCA tools will depend on your platform, stack and product

SCA Tools

- Primary focus of SCA tools has been on security vulnerabilities because of the perceived higher risk
- Most SCA tools focus on either vulnerabilities OR licensing
- Vulnerabilities and licenses seem like oil and water
  - The communities of interest are separate - security vs legal
  - License data may be complex, but generally stable over time
  - Vulnerability data is also complex, but extremely dynamic - if included directly in an SBOM, it may be wrong by the time you receive an SBOM
- But you need SCA coverage for both - plus quality

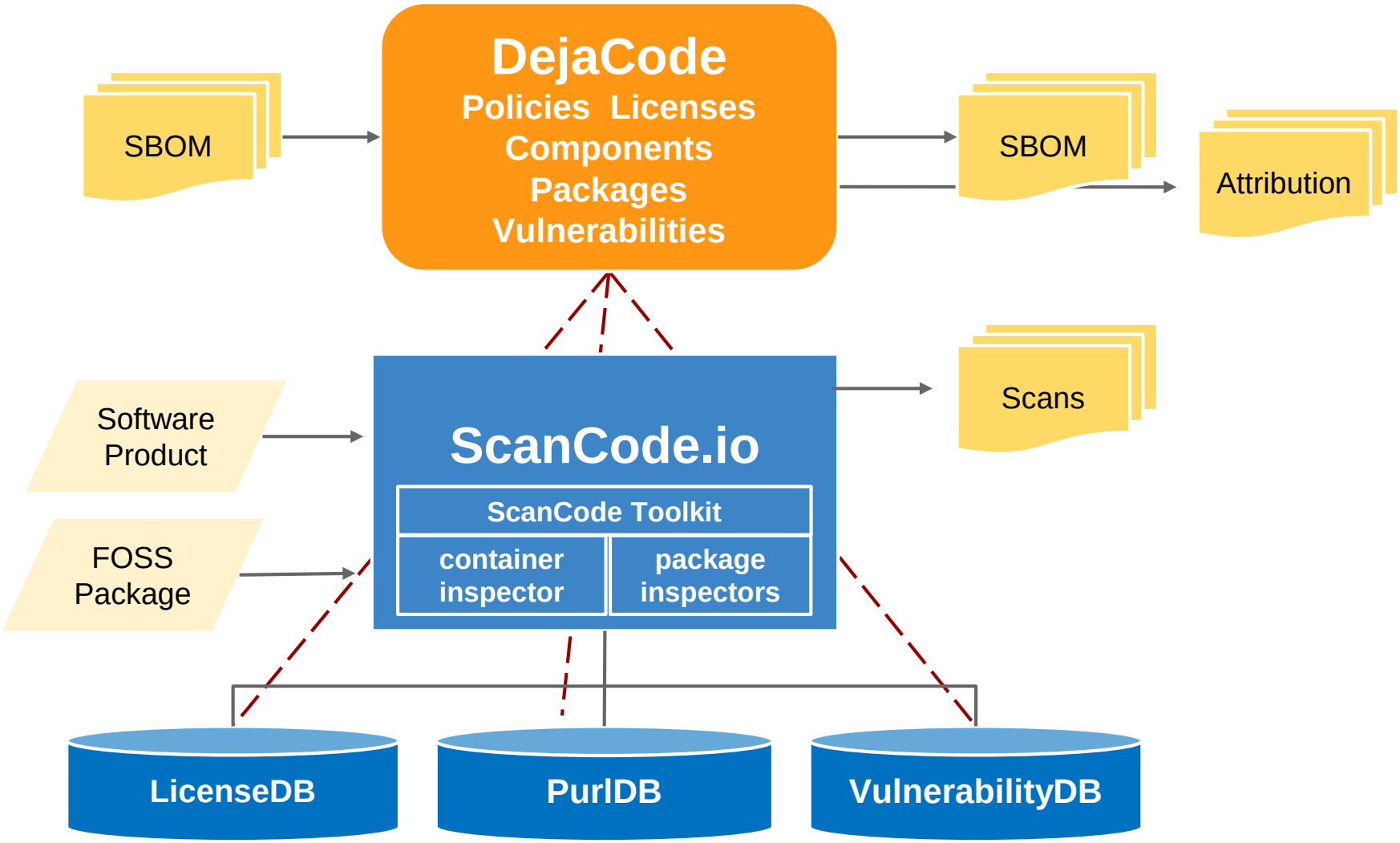
SCA Tools [2]

- Most current tools are proprietary and increasingly expensive with the surge of interest in SBOMs
  - Trend seems to be charging based on the total number of developers in your organization
  - Good for the vendor - not for the customer
- Proprietary solutions may work for large companies, but they will not work across the FOSS supply chain
  - Proprietary data about FOSS vulnerabilities is particularly problematic as a barrier to community access and analysis

FOSS tools for SCA

- Modular tools for developers:
  - Free and open source software (Apache 2.0)
  - Free and open data (CC-BY-SA)
- ScanCode: Leading code scanner
- VulnerableCode: New tools and database for aggregating vulnerability data from across the FOSS supply chain
- PurlDB: New tools and database for aggregating package data across the FOSS supply chain
- DejaCode: SCA management application

FOSS tools for Software Supply Chain Security



DejaCode

- Compliance application / system of record for:
  - Managing Inventory and BOM data
  - Defining and applying license policies
  - Identifying and addressing package vulnerabilities
  - Generating FOSS compliance documents such as Product Attribution Notices and SBOMs
- Built-in integration with ScanCode.io, VulnerableCode.io and PurlDB
- SaaS or on-premises
- See https://nexb.com/dejacode/

ScanCode

- Identify FOSS and other third-party components & packages
- Detect licenses, copyrights and dependencies
- ScanCode Projects include:
  - **ScanCode.io:** Server system with customizable pipelines and UI
  - **ScanCode Toolkit**: Scanning engine - use it in SCIO or as a separate CLI or library
  - **LicenseDB**: 2000+ licenses recognized by ScanCode
  - **ScanCode Workbench**: Desktop app to review Toolkit Scans
  - scancode-analyzer: Analyze and improve license detection accuracy

- See https://nexb.com/scancode/ for more information

VulnerableCode

- Collect and aggregate vulnerability data from many public sources
  - Projects, GitHub, Linux Distros, NVD, Package managers and more
  - Focus on upstream projects (source of the source)
- Apply confidence based system: not all data are equally trusted and of equivalent quality
- Discover relations (and inconsistencies) between vulnerabilities and packages from mining the graph
- Public VulnerableCode database is available at: https://public.vulnerablecode.io/
  - Also tools to build your own database
  - Working on data sharing and curation
- See https://nexb.com/vulnerablecode/ for more information

PurlDB

- Collect and aggregate package metadata from many public sources
  - Package manager repositories
  - GitHub, GitLab and other source repositories
  - Linux distros
  - Focus on upstream projects (source of the source)
- Will support package matching as a complement to scanning
- Also tools to build your own database
- See https://github.com/nexB/purldb/ for more information

Other AboutCode projects

- container-inspector: Analyze Docker and other images
- debian-inspector: Parse Debian copyright files
- nuget-inspector: Resolve C# dependencies
- python-inspector: Resolve Python dependencies
- aboutcode-toolkit: Generate Attribution Notices
- package-url (purl): URL string to identify and locate a software package across programing languages, package managers, packaging conventions, tools, APIs and databases.
  - Adopted by ORT, CycloneDX and many other major projects
  - See also https://github.com/package-url
- univers: parse and compare package versions and version ranges
- See https://github.com/nexB for the complete list of projects

Contact us

- Contacts
  - Michael Herzog
    mjherzog@nexb.com
  - Philippe Ombredanne
    pombredanne@nexb.com
  - Dennis Clark
    dmclark@nexb.com

- More information - https://www.nexb.com/