# SBOMs and Software Vulnerabilities: Leveraging SCA for Software Supply Chain Security

# Abstract

- Software Bill of Materials (SBOM) and Software Composition Analysis (SCA) have become common terminology in the software industry
- Understanding both is essential for managing the growing risk of software vulnerabilities for all kinds of software and planning for compliance with rapidly evolving regulatory and business requirements
- This presentation will cover:
  - The several SBOM specifications
  - Using SCA to find and report software licenses and vulnerabilities
  - Overview of nexB's DejaCode, ScanCode and VulnerableCode to manage software supply chain risk

# Agenda

nexB

- ▷ Software Supply Chain and Bills of Materials
- ▷ Software Composition Analysis
- ▷ Using SCA to create and manage SBOM data
- ▷ Key industry and regulatory trends to watch

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 - often in the range of 80% (depending on how you count).

# Software Supply Chain

- SBOMs are a key part of the larger concept of a Software Supply Chain
- Most concepts borrowed from discrete manufacturing
- BOMs in the software context appeared in draft legislation in 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

# Software Bill of Materials (SBOM)

- An SBOM is a list of software components used in a product
  - 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 likely need to produce SBOMs

# Why SBOMs

- Providing an SBOM with your software is becoming a requirement for doing business with US government agencies
- Most modern software contains third-party software - FOSS or Proprietary - which means potential risks in the areas of licensing and vulnerabilities
- A better question might be: Why haven’t we been using SBOMs before?

# Why SBOMs [2]

- 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
- And the entry point for managing these risks is agreeing somehow on the identification of the software units across a supply chain

# SBOM Standards

There are currently two emerging standards for an SBOM:

- CycloneDX - https://cyclonedx.org/ - from OWASP
- SPDX - https://spdx.dev/ - from the Linux Foundation
- And one weaker candidate: SWID - https://csrc.nist.gov/projects/Software-Identification-SWID
- It is unlikely that there will be only one standard and possible that there will be more than two
- These are standards for data exchange, not design standards for any particular software system

# SBOM Standards [2]

- SBOMs are a top priority for improving software supply chain security
- CISA* currently has five weekly meetings on the topic!
- 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

* CISA: Cybersecurity and Infrastructure Security Agency within DHS

# Supply Chain Best Practices

- 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 BOMs and units

  which means standardarizing data and

  learning to translate among multiple standards

# Software Composition Analysis

Software Composition Analysis 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.

# Software Composition Analysis [2]

A more detailed list:

- Software Component Identification
- Dependency Management
- Software Bills of Materials
- License Identification
- Vulnerability Reporting
- Code Quality Reporting
- Community Health Reporting
- SCA Management

# Software Composition Analysis [3]

- Overall SCA needs to be a core competency for a software development organization
- Embedded 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 have been focused on either vulnerabilities OR licensing
- Vulnerabilities and licenses seem like oil & vinegar
  - The communities of interest are separate - security vs legal
  - License data may be complex but it is 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

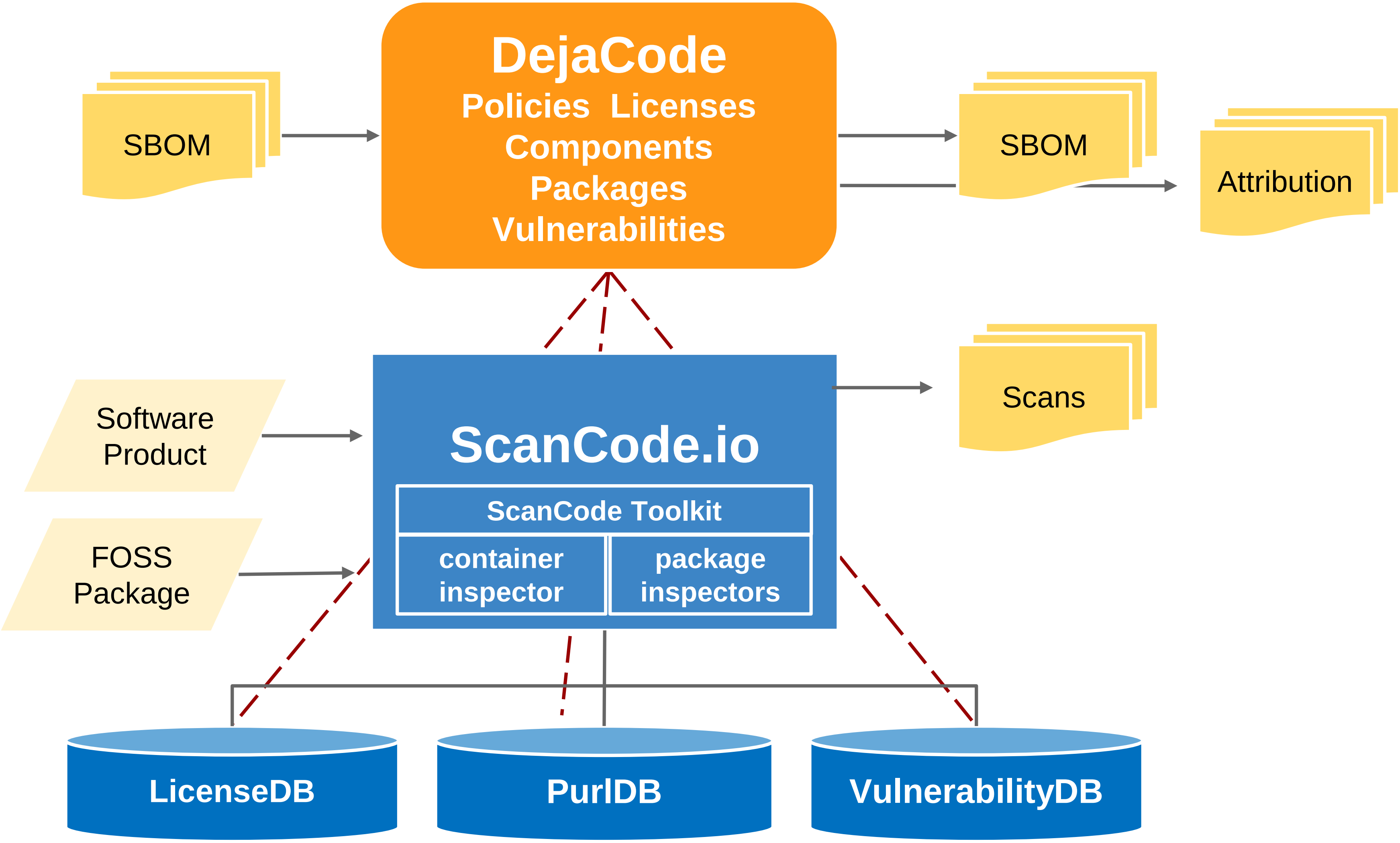
# SCA Tools [2]

- Most current tools are Proprietary and increasingly expensive with the surge of interest in SBOMs
  - Trend seems to be charging based on the number of developers
  - Good for the vendor not so much 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
- Current hot topic is SBOMs

# nexB SCA Tools

- Modular tools for developer with:
  - 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
- Many other FOSS projects
- DejaCode - enterprise SCA management application

# nexB SCA Solutions Overview



# DejaCode

- Enterprise 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 detected by ScanCode
  - **ScanCode Workbench**: Desktop app to review Toolkit Scans
  - scancode-analyzer: Analyze and improve license detection accuracy

- See [https://nexb.com/scancode/](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

# Why nexB

- nexB has been recognized by marquee 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
  - Creator of package-url: https://github.com/package-url
  - Co-founders of SPDX: https://spdx.org
  - Co-founders of ClearlyDefined: https://clearlydefined.io

# Contact us

- Contacts
  - Michael Herzog
    mjherzog@nexb.com
  - Philippe Ombredanne
    pombredanne@nexb.com
  - Dennis Clark
    dmclark@nexb.com
- More information - https://www.nexb.com/