What is SBOM （<https://www.ntia.gov/sites/default/files/publications/sbom_at_a_glance_apr2021_0.pdf>）

A Software Bill of Materials (SBOM) is a structured list that provides detailed information about the various software components used in a particular software application, system, or product. Think of it as a list of ingredients in a recipe, but for software. An SBOM includes information about both proprietary and open-source components, their versions, licenses, and dependencies. SBOMs should also include baseline attributes with the ability to uniquely identify individual components in a standard data format. The most efficient generation of SBOMs is as a byproduct of a modern development process. The primary purpose of an SBOM is to uniquely and unambiguously identify components and their relationships to one another.

A diagram of a tree

Description automatically generated

Benefits: reduce cost, security risk, license risk, and compliance risk.

Use cases: improved software development, supply chain management, vulnerability management, asset management, procurement, and high assurance processes.

Types of SBOM: design, source, build, analyzed, deployed, runtime

Software supply chain security

The supply chain is made up of everything and everyone that touches the code in the software development lifecycle (SDLC), from application development to the CI/CD pipeline and deployment. It includes networks of information about the software, like the components (infrastructure, hardware, operating systems, cloud services, etc.), the people who wrote them, and the sources they come from, like registries, GitHub repositories, codebases, or other open-source projects. It also includes any vulnerabilities that may negatively impact software security and that’s where software supply chain security comes in.

Security risks: risk to any component of the software supply chain presents a potential risk to every software artifact relying on that supply chain component.

4 types of risk: vulnerabilities, licensing (legal risk), third party dependencies, processes and policies

Vulnerability Exploitability eXchange (VEX) is used to allow a software supplier or other parties to assert the status of specific vulnerabilities in a particular product.

Automated security testing tools: Software composition analysis (SCA), static application security testing (SAST), and dynamic application security testing (DAST)

Tools:

FOSSA

Spectral

Jit

Codenotary

JFrog

Anchore

Cybeats

Endor Labs

Rezilion

SPDX SBOM Generator

Types of SBOM Documents: <https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf>

SBOM Vulnerability Assessment & Corresponding Requirements (<https://www.ntia.doc.gov/files/ntia/publications/camp_andalibi_-_2021.06.16.pdf>)

SBOM Tool Classification Taxonomy: <https://www.ntia.gov/sites/default/files/publications/ntia_sbom_tooling_taxonomy-2021mar30_0.pdf>

Blockchain-enabled SBOM <https://arxiv.org/abs/2307.02088>

Detection system: <https://www.scientific.net/AMM.336-338.2419>

1. Supply chain, sbom, c++ (javascript, python, .net) (any existing tools or study)
2. C++ project to look into (open source, big project)
3. How many tools above works with C++
4. What (how) is third party dependence in C++?
5. To find out what can I do for this problem.
6. An empirical study on TPLs in C/C++ <https://arxiv.org/pdf/2209.02575.pdf> : introduce a tool called CCScanner to detect detect dependencies introduced through 21 management tools and detect code clone

TPL detection: <https://ieeexplore-ieee-org.eres.library.manoa.hawaii.edu/stamp/stamp.jsp?tp=&arnumber=10172788> : introduce a Software Composition analysis SCA framework called OSSFP to detect TPL

Tools for TPL management: NPM for Node.js, Pip for python, Maven for Java. No such tool for c/c++ yet.

OSSPolice detects C/C++ TPL dependencies in Android applications but it collects GitHub repo as TPLs leads to false positives

Code clone detector: limitation is source code often not get put in the repo

SBOM tools can be used but only provide the description of dependencies

CMake

1. Till July 2023, there are around 24k C/C++ repo that can be studied
2. In C/C++, methods to handle dependencies are system package manager, header-only libraries, git submodules.

Application level package manager for C/C++: Conan

Code clone detection for C++: Centris, DejaVu, FOSSID

TPL in binaries detection: OSSPolice, Modx

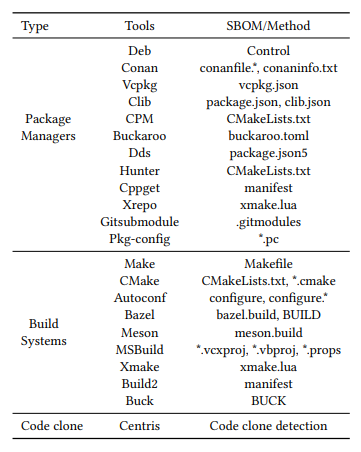
SBOM: OWASP, Sonatype

1. Dependencies are added unintentionally in build scripts not explicitly managing dependency installation, TPL inconsistent between databases, system libraries on OS are the most important libraries for C/C++

No central SBOM to store all package information

TPL in C/C++ are TPL data and methods. TPL data includes default OS libraries, mirrors of system level package managers, and source repo. TPL methods include package management tools and code clone.

1. Large scale studies for unified C/C++ package manager or build such tool (maybe too big project)
2. Try to use CCScanner
3. Try to use conan and other tools to get familiar with



Direction:

1. Create tool/
2. Literature review/
3. Empirical study (tools simility and difference) requires more projects

Package history (different release for

Tasks: this month get familiar with all tools to decide the direction, compare them