

LF Mentorship Series: Generating Software Bill of Materials

Kate Stewart, VP, Dependable Embedded Systems
The Linux Foundation

Thursday March 25, 2021

LinkedIn: <https://www.linkedin.com/in/katestewartastin/>

Twitter: @_kate_stewart

Email: kstewart@linuxfoundation.org

INGREDIENTS: ENRICHED BLEACHED WHEAT FLOUR [FLOUR, REDUCED IRON, "B" VITAMINS (NIAIN, THIAMINE MONONITRATE (B1), RIBOFLAVIN (B2), FOLIC ACID)], WATER, SUGAR, CORN SYRUP, HIGH FRUCTOSE CORN SYRUP, PARTIALLY HYDROGENATED VEGETABLE AND/OR ANIMAL SHORTENING (SOYBEAN, COTTONSEED AND/OR CANOLA OIL, BEEF FAT), WHOLE EGGS, DEXTROSE. CONTAINS 2% OR LESS OF: SOY LECITHIN, LEAVENINGS (SODIUM ACID PYROPHOSPHATE, BAKING SODA, CORNSTARCH, AND MONOCALCIUM PHOSPHATE) WHEY, MODIFIED CORN STARCH, GLUCOSE, SOY FLOUR, SALT, MONO AND DIGLYCERIDES, CELLULOSE GUM, CORNSTARCH, SODIUM STEAROYL LACTYLATE, NATURAL AND ARTIFICIAL FLAVOR, SORBIC ACID (TO RETAIN FRESHNESS), POLYSORBATE 60, SOY PROTEIN ISOLATE, CALCIUM AND SODIUM CASEINATE, YELLOW 5, RED 40. 518701

CONTAINS WHEAT, EGG, MILK AND SOY
212016_MP



CONTAINS WHEAT, EGG, MILK AND SOY
212016_MP

, REDUCED IRON, "B"
IN (B2), FOLIC ACID)],
SYRUP, PARTIALLY
TENING (SOYBEAN,
E EGGS, DEXTROSE.
NGS (SODIUM ACID
ND MONOCALCIUM
, SOY FLOUR, SALT,
I, SODIUM STEAROYL
C ACID (TO RETAIN
LCIUM AND SODIUM
518701







INGREDIENTS: ENRICHED BLEACHED WHEAT FLOUR [FLOUR, REDUCED IRON, "B" VITAMINS (NIACIN, THIAMINE MONONITRATE (B1), RIBOFLAVIN (B2), FOLIC ACID)], WATER, SUGAR, CORN SYRUP, HIGH FRUCTOSE CORN SYRUP, PARTIALLY HYDROGENATED VEGETABLE AND/OR ANIMAL SHORTENING (SOYBEAN, COTTONSEED AND/OR CANOLA OIL, **BEEF FAT**) WHOLE EGGS, DEXTROSE. CONTAINS 2% OR LESS OF: SOY LECITHIN, LEAVENINGS (SODIUM ACID PYROPHOSPHATE, BAKING SODA, CORNSTARCH, AND MONOCALCIUM PHOSPHATE) WHEY, MODIFIED CORN STARCH, GLUCOSE, SOY FLOUR, SALT, MONO AND DIGLYCERIDES, CELLULOSE GUM, CORNSTARCH, SODIUM STEAROYL LACTYLATE, NATURAL AND ARTIFICIAL FLAVOR, SORBIC ACID (TO RETAIN FRESHNESS), POLYSORBATE 60, SOY PROTEIN ISOLATE, CALCIUM AND SODIUM CASEINATE, YELLOW 5, RED 40. 518701

CONTAINS WHEAT, EGG, MILK AND SOY
212016_MP




Overview of Supplied Installation Kits

Anchoring Kit

Part	Used in	Number of units
Rear anchoring bracket	<i>Mount the Rear Anchoring Brackets for the Classic Battery Cabinet, page 22</i>	1 
Rod for fastening to rear anchoring bracket	<i>Position the Classic Battery Cabinets, page 28</i>	4 
Front anchoring bracket	<i>Mount the Front Anchoring Brackets on the Classic Battery Cabinets, page 33</i>	1 
Front anchor cover		1 

Installation Kit 0M-816909

Part	Used in	Number of units
Spacer between classic battery cabinet and UPS	<i>Mount the Rear Anchoring Brackets for the Classic Battery Cabinet, page 22</i>	1 

Three perspectives across the supply chain

- Produce Software (Supplier/Upstream)
- Choose Software (Consumer/Downstream)
- Operate Software (Use/In Organization)

How many organizations can answer:

Am I potentially affected by \$**vulnerability**\$?



Or know how to detect and remediate software supply chain attacks?

SUPPLY CHAIN ATTACK

Attackers insert malicious code into a DLL component of legitimate software. The compromised DLL is distributed to organizations that use the related software.

EXECUTION, PERSISTENCE

When the software starts, the compromised DLL loads, and the inserted malicious code calls the function that contains the backdoor capabilities.

DEFENSE EVASION

The backdoor has a lengthy list of checks to make sure it's running in an actual compromised network.

RECON

The backdoor gathers system info

INITIAL C2

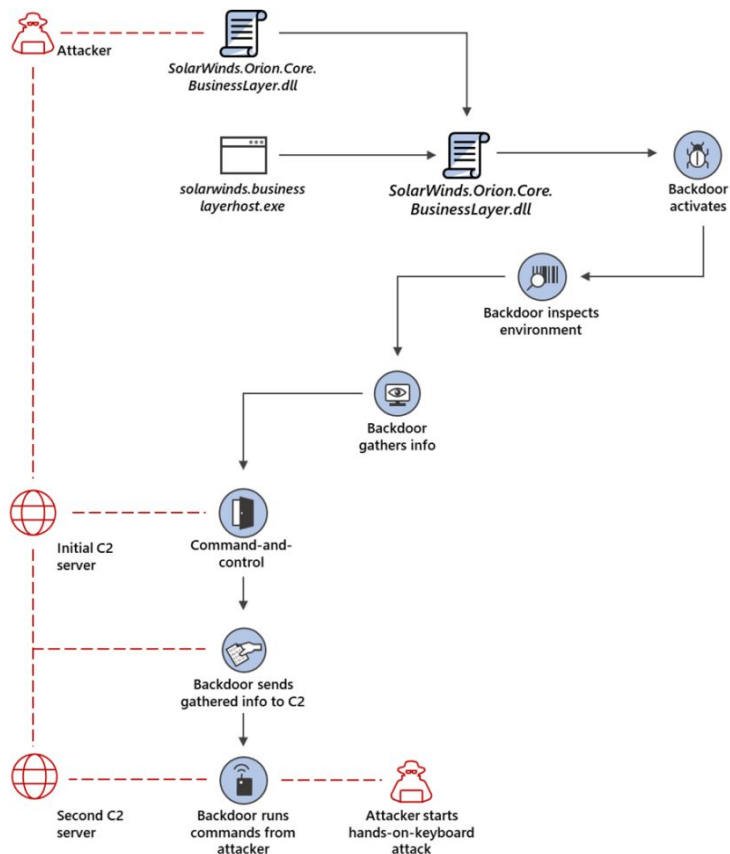
The backdoor connects to a command-and-control server. The domain it connects to is partly based on info gathered from system, making each subdomain unique. The backdoor may receive an additional C2 address to connect to.

EXFILTRATION

The backdoor sends gathered information to the attacker.

HANDS-ON-KEYBOARD ATTACK

The backdoor runs commands it receives from attackers. The wide range of backdoor capabilities allow attackers to perform additional activities, such as credential theft, progressive privilege escalation, and lateral movement.



Cost to Remediate?

Cost to developers to fix is small, cost to users

POLICY

Cleaning up SolarWinds hack may cost as much as \$100 billion

Government agencies, private corporations will spend months and billions of dollars to root out the Russian malicious code

Source: <https://www.rolcall.com/2021/01/11/cleaning-up-solarwinds-hack-may-cost-as-much-as-100-billion/>

ITPro.

Business Cloud Hardware Infrastructure Security Software Technology Resources .co.uk



NEWS Home > Security > Cyber Security

Flaws in open source protocols expose millions of embedded devices

Amnesia:33 vulnerabilities could impact numerous industries, from health care to retail and beyond

by: **Rene Millman** 9 Dec 2020

Source: <https://www.itpro.co.uk/security/cyber-security/358064/flaws-in-open-source-protocols-expose-millions-of-embedded-devices>

The system **outage lasted for more than 40 days** and the health system reassigned or furloughed around 300 workers who were unable to do their jobs as a result of the computer outage. UVM Health brought in the National Guard's cybersecurity unit to help restore the computers. During the outage, the health system **postponed some services**. On Dec. 8, UVM Medical Center President and COO Stephen Leffler, MD, said the health system is losing \$1.5 million per day in revenue and extra expenses; the health system expects the entire incident will cost more than **\$63 million** by the time it resolves next year.

Source:

<https://www.beckershospitalreview.com/cybersecurity/the-5-most-significant-cyberattacks-in-healthcare-for-2020.html>

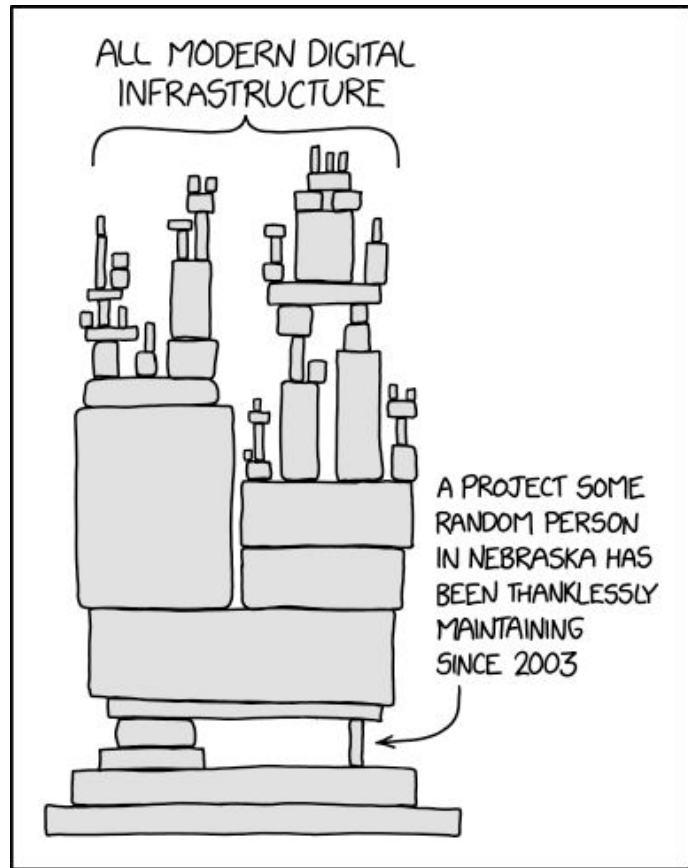
**"99% of codebases audited in 2019 contained
open source components.**

Open source made up **70%** of the audited
codebases."

Source: [2020 Black Duck Report](#)

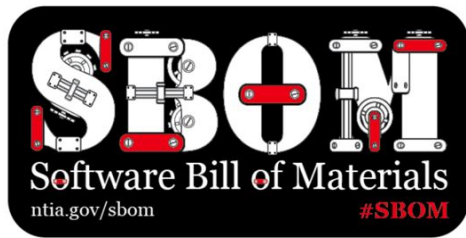
“ We’ve observed **double and triple digit growth** in open source component ecosystems for a decade, and there is **no slowdown in sight.**”

Most companies
are **not** able to
accurately
summarize the
software that is
running on their
systems.



Source: <https://xkcd.com/2347/> This work is licensed under a [Creative Commons Attribution-NonCommercial 2.5 License](https://creativecommons.org/licenses/by-nc/2.5/).

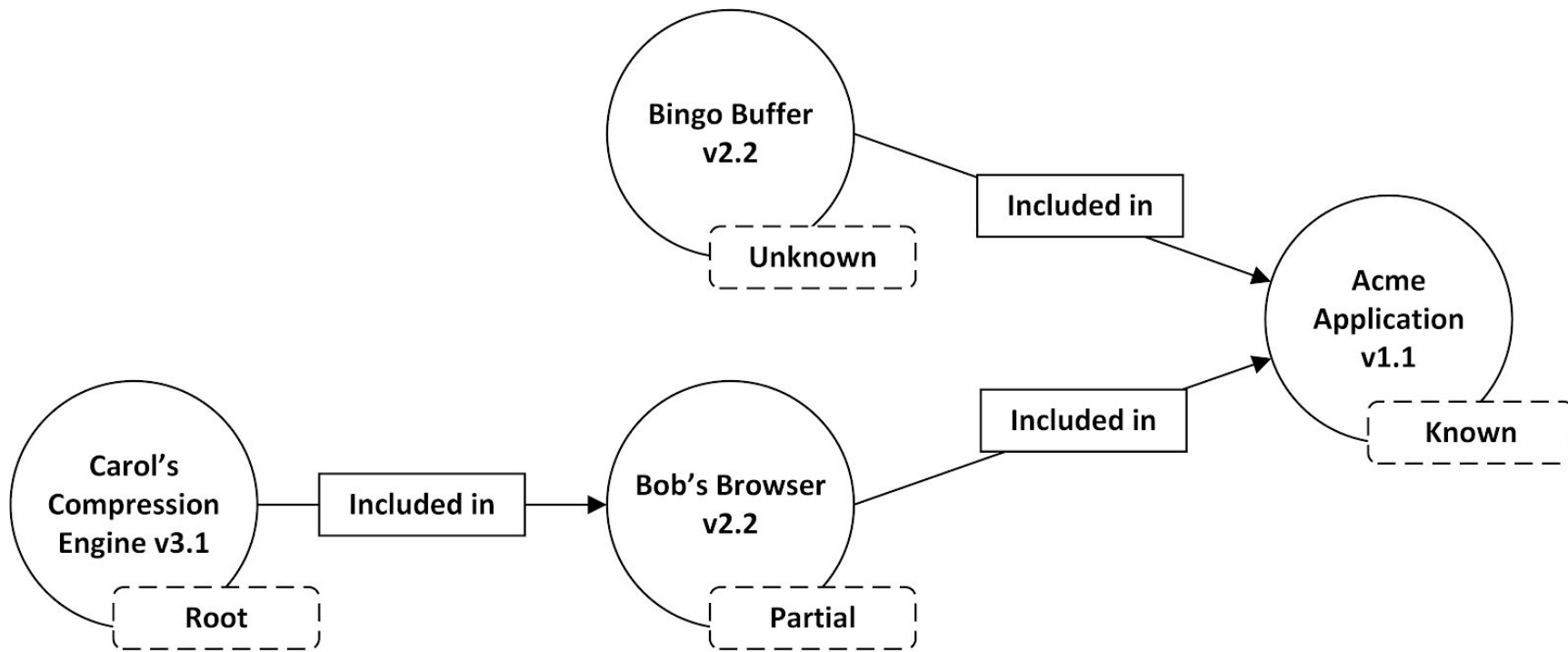
Software Bill of Materials (**SBOM**)



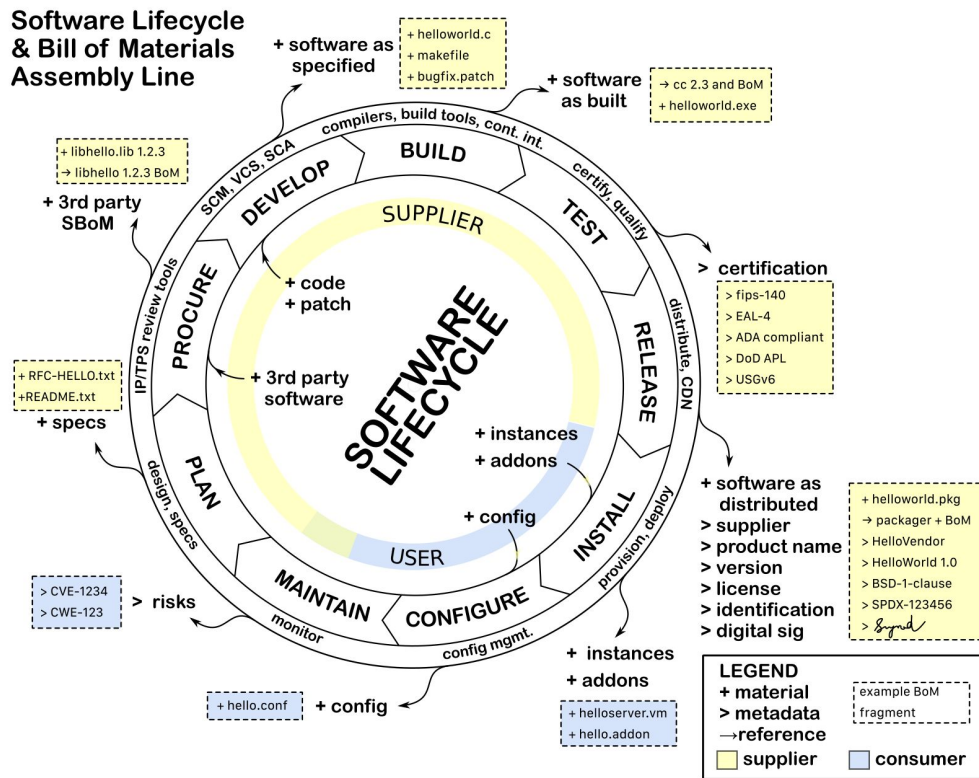
An SBOM is a formal record containing the details and supply chain relationships of various components used in building software.

These components, including libraries and modules, can be open source or proprietary, free or paid, and the data can be widely available or access-restricted.

What does an SBOM contain?



When should an SBOM be used?



Why Should I care about this now?

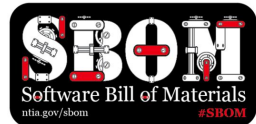
Detection and remediation of vulnerabilities is costing \$\$\$\$ and is motivating interest in improving **Cybersecurity Supply Chain Management**.

Factors that have contributed to the problems:

- › **Reuse:** Fast time to market by reusing existing components
- › **Containers:** Components being executed on system are not obvious
- › Software Transparency is an **assumed as “pre-requisite” for analysis**, but may not be machine readable ⇒ manual effort, binary and container analysis, trace tools, etc.

Regulatory authorities growing awareness of cybersecurity supply chain threats - FDA, FERC...

In US, **NTIA** is coordinating to create an industry friendly solution.



Why are you hearing more about them now?

- Supply chain security issues increasingly visible - Solarwinds, etc.
- Seeing as expectation from government & regulatory agencies:
 - in US: FDA, NERC
 - in Europe: ENISA Cybersecurity for Cloud Services

Ref	Description	Ass. Level
DEV-02.1	The CSP shall maintain a list of dependencies to hardware and software products used in the development of its cloud service	Basic
DEV-02.2	The CSP shall document and implement policies for the use of third-party and open source software	Substantial
DEV-02.3	The CSP makes its list of dependencies available to customers upon request	Substantial

Guidance elements

DEV-02.1	For its software components, the list of dependencies is often called Software Board of Materials (SBOM) . In the context of [EUCSA], Article 51(d) requires the identification and documentation of known dependencies. Dependencies should include all software modules, libraries or APIs used, as well as development tools.
----------	---



Download

PDF document, 3.31 MB

CIP-013-1 – Cyber Security - Supply Chain Risk Management

A. Introduction

1. **Title:** Cyber Security - Supply Chain Risk Management
2. **Number:** CIP-013-1
3. **Purpose:** To mitigate cyber security risks to the reliable operation of the Bulk Electric System (BES) by implementing security controls for supply chain risk management of BES Cyber Systems.

Source: <https://www.nerc.com/pa/Stand/Reliability%20Standards/CIP-013-1.pdf>

Who should use an SBOM?

Any organization concerned about better supporting their software products internally, supporting their customers, and positively differentiating themselves in the marketplace should consider creating SBOMs and providing them to support their customers.

An SBOM is commonly required as part of any product's BOM so necessary information is available:

- **Contractual** - negotiated terms, implementation strategies
- **Legal** - compliance with licensing and regulatory obligations
- **Technical** - identification of software or component dependencies and supply chain risk, vulnerability and asset management

NTIA Multistakeholder Process on Software Component Transparency | ntlia.gov/sbom

SBOM FAQ

Table of Contents

Table of Contents	1
OVERVIEW	2
Q: What is an SBOM?	2
Q: Who should have an SBOM?	2
Q: Who uses an SBOM and for what?	2
BENEFITS	3
Q: What are the benefits of an SBOM?	3
Q: How does an SBOM help in the event of a cyberattack?	3
Q: In addition to vulnerability management, how can SBOMs help me?	4
Q: How have bills of material and supply chain transparency been helpful elsewhere?	4
COMMON MISCONCEPTIONS & CONCERNS	4
Q: Won't SBOMs be a "roadmap to the attacker"?	4
Q: Does an SBOM require source code disclosure?	5
Q: Does a list of the software components I include expose my intellectual property?	5
Q: Does an SBOM increase my exposure to license violations?	5
Q: Does an SBOM enable patent or license "tricks"?	5
Q: Will SBOMs increase my licensing costs or licensing commitments?	6
CREATION	6
Q: Who creates and maintains an SBOM?	6
Q: What should be included in an SBOM?	6
Q: When is an SBOM created, changed, or maintained?	6
Q: Some software components are made up of other software components themselves. Can an SBOM show that hierarchy?	7
Q: How deep in the dependency graph should an SBOM enumerate?	7
DISTRIBUTION & SHARING	7
Q: If I make an SBOM, do I have to make it public?	7
Q: How will SBOM data be shared?	7
ROLE SPECIFIC	8
Q: How can SBOMs be leveraged as a Purchaser?	8
Q: How can SBOMs help an engineer provide surveillance for deployed technology in the field for emerging vulnerabilities?	8
GET INVOLVED	9
Q: Where can I find more information about the NTIA SBOM process? How do I get involved?	9
Last Revised: 2020-09-08	1

Use Cases: Producing software

- Monitor for vulnerabilities in components
- Better manage code base & minimize bloat
- Execute allow-list or deny-list practices
- Prepare and respond to end-of-life contingencies
- Know and comply with license obligations
- Provide an SBoM for customers



Use Cases: Choosing software

- Identify known vulnerabilities
- More targeted security analysis
- Compliance
- EOL awareness
- Verify sourcing & supplier claims
- Understand software integration
- Market signal of secure development process.



Use Cases: Operating Software

- Vulnerability management
- Better understanding of operational risks
- Real time data on components in assets
- Improved understanding of potential exploitability
- Enable potential non-SW mitigations



3.0 Open Source Content Review and Approval

3.1 Bill of Materials

A process exists for creating and managing a bill of materials that includes each Open Source component (and its Identified Licenses) from which the Supplied Software is comprised.

Verification Material(s):

- ☐ 3.1.1 A documented procedure for identifying, tracking, reviewing, approving, and archiving information about the collection of Open Source components from which the Supplied Software is comprised.
- ☐ 3.1.2 Open Source component records for the Supplied Software that demonstrates the documented procedure was properly followed.

Rationale:

To ensure a process exists for creating and managing an Open Source component bill of materials used to construct the Supplied Software. A bill of materials is needed to support the systematic review and approval of each component's license terms to understand the obligations and restrictions as it applies to the distribution of the Supplied Software.

SBOMs make simple financial sense

"Having visibility into all our product families through an SBOM could easily save us **100,000 man hours** in time and effort to identify where potentially risky components are, and efficiently manage them across our development teams and customers."

– Product Security Director, Fortune Global 500 company

What do SBOM's look like today?

Many different forms...

Need to standardize to automate.



What third-party software is used by SimpleRisk and how is it licensed?

Modified on: Sat, Mar 14, 2020 at 2:22 PM



Print

As with just about any software product these days, SimpleRisk did not write 100% of the code included in the product. Over the years, we have used a variety of third-party software to provide features and functionality for our user base. In 2019, we performed a full audit of all third-party source code that has been included in the product and verified that we are in compliance with all known licenses for the included software. Below is the Bill of Materials (BOM), produced from that effort, which outlines each of these software packages and the licensing that was found for them. If you are the developer, maintainer, or licensor for any of these packages and believe that this information is in error, we'd ask that you please submit a support ticket to address your concerns.

SimpleRisk Bill of Materials (BOM)

SimpleRisk Core: This is the free and open source offering from SimpleRisk that also forms the basis for both our on-prem and hosted offerings. It is licensed under the Mozilla Public License (MPL) 2.0.

PHP Libraries Included in the SimpleRisk Core

- HighchartsPHP: Licensed under the GNU General Public License (Version 3, 29 June 2007).
- PHPMailer: Licensed under the GNU Lesser General Public License (Version 2.1, February 1999)
- CSRF-Magic (<http://csrf.htmlpurifier.org/>): Licensed under the BSD 2-Clause "Simplified" License.
- Epiphany: Custom copyright notice and license located under `simplerisk/includes/epiphany/LICENSE`.
- Zend Escaper: Custom copyright notice and license located under `simplerisk/includes/Component_ZendEscaper/LICENSE.md`.

Javascript Libraries Included in the SimpleRisk Core

- HighCharts: SimpleRisk has purchased a perpetual Highcharts OEM License for unlimited installations. This license applies for all customers using SimpleRisk, regardless of whether they are utilizing it in a hosted or on-premise installation.
- JQuery (<https://jquery.org/>): Licensed under the MIT license.
- JQuery Tree Widget (<https://github.com/daredevel/jquery-tree>): Licensed under the MIT license.
- EasyUI for JQuery (www.jeasyui.com): Licensed under the [freeware](#) license.
- Sorttable (<https://kryogenix.org/code/browser/sorttable/>): Licensed under the [X11](#) license.

Source: <https://simplerisk.freshdesk.com/support/solutions/articles/6000230367-what-third-party-software-is-used-by-simplerisk-and-how-is-it-licensed->

What should a minimum viable SBOM contain?

NTIA SBOM Baseline	Definition
Supplier Name	Name or identity of the supplier of the component in the SBOM entry, including some capability to note multiple names or aliases.
Component Name	Component (and supplier) names can be conveyed using a generic namespace:name construct.
Unique Identifier	A unique identifier can be generated and used to help identify components. This identifier could be a version 4 or 5 UUID.
Version String	Version information helps to identify a component.
Component Hash	Adding a cryptographic hash of the component is the most precise way to identify a binary, as-built component in an SBOM. The hash is effectively the unique identifier of a component, however other baseline identification information will be useful and even necessary.
Relationship	Relationship is inherent in the design of the SBOM. The default relationship type is includes.
Author Name	Author of the SBOM entry (this may not always be the supplier).

How to represent minimum viable SBOM Info?

NTIA SBOM Baseline	SPDX	SWID	CycloneDX
Supplier Name	(3.5) PackageSupplier:	<Entity> @role (softwareCreator/publisher), @name	publisher
Component Name	(3.1) PackageName:	<softwareIdentity> @name	name
Unique Identifier	(3.2) SPDXID:	<softwareIdentity> @tagID	bom/serialNumber and component/bom-ref
Version String	(3.3) PackageVersion:	<softwareIdentity> @version	version
Component Hash	(3.10) PackageChecksum:	<Payload>/../<File> @[hash-algorithm]:hash	hash
Relationship	(7.1) Relationship: CONTAINS	<Link> @rel, @href	(Nested assembly/subassembly and/or dependency graphs)
Author Name	(2.8) Creator:	<Entity> @role (tagCreator), @name	bom-descriptor:metadata/manufacture/contact

Taxonomy used for Classifying SBOM Tools

Category	Type	Description
Produce	Build	Document is automatically created as part of building an artifact and contains information about the build.
	Analyze	Analysis of source or binary files will generate the SBOM by inspection of the artifacts and any associated sources
	Edit	A tool to assist a person manually entering or editing SBOM data
Consume	View	Be able to understand the contents in human readable form (picture, figures, tables, text.). Use to support decision making & business processes.
	Diff	Be able to compare multiple SBOMs and clearly see the differences (e.g. comparing two versions of a piece of software)
	Import	Be able to discover, retrieve, and import an SBOM into your system for further processing and analysis
Transform	Translate	Change from one file type to another file type while preserving the same information.
	Merge	Multiple sources of SBOM and other data can be merged together for analysis and audit purposes
	Tool support	Support use in other tools by APIs, object models, libraries, or other reference sources

Tool Support for Different SBOM Formats

SPDX

Format Overview	2
Format Publishing History	2
Tool Classification Taxonomy	2
Open Source Tools	4
Augur	4
FOSSology	4
in-toio	5
kernel-spdx-ids	5
npm-spdx	6
Open Source Software Review Toolkit (ORT)	6
OWASP Dependency-Track	6
Quartemaster (QMSTR)	7
REUSE	8
ScanCode Toolkit	8
SPDX Java Libraries and Tools	9
SPDX Python Libraries	10
SPDX Golang Libraries	10
SPDX JavaScript Libraries	11
SPDX Online Tools	11
SPDX Maven Plugin	12
SPDX Build Tool	12
SPARTS	12
SW360	13
TERN	13
Yocto Project / OpenEmbedded	14
Proprietary Products	15
CyberProtek	15
FOSSID	15
Hub-SPDX (Black Duck Hub Report Utility)	16
MedScan	16
Protecode	17
Protex	17
SourceAuditor	17
TrustSource	18
Vigilant-ops	18

SWID

Format Overview	2
Format Publishing History	2
Tool Classification Taxonomy	2
Open Source Tools	3
Swidgen	3
StrongSwan SWID Generator	3
Labels4 SWID Generator	3
Labels4 SWID Maven Plugin	4
libswid	4
SwidTag	4
TagVault SWID Tag Creator	5
RPM 2 SWID Tag	5
NIST SWID for GNU Autotools	6
NIST SWID Tag Validator	6
NIST SWID Builder	6
NIST SWID Maven Plugin	7
NIST SWID Repo Client	7
WIX Toolset	8
swidq	8
Proprietary Products	9
IT Operations Management	9
Jamf Pro	9
CyberProtek	10
MedScan	10
BigFix Inventory	11
Vigilant-ops	12
Microsoft Endpoint Configuration Manager	12

<http://tiny.cc/SWID>

CycloneDX

Format Overview	2
Format Publishing History	2
Tool Classification Taxonomy	2
Open Source Tools	3
CycloneDX Core for Java	3
CycloneDX for .NET	3
CycloneDX for NPM	3
CycloneDX for Maven	4
CycloneDX for Gradle	4
CycloneDX for PHP Composer	4
CycloneDX for Python	5
CycloneDX for Ruby Gems	5
CycloneDX for Rust Cargo	5
CycloneDX for SBT	6
CycloneDX for Elixir Mix	6
CycloneDX for Erlang Rebar3	6
CycloneDX for Go	7
Eclipse SW360 Antenna	7
HERE Open Source Review Toolkit	7
Retire.js	8
OWASP Dependency-Track	8
OWASP Dependency-Track Jenkins Plugin	8
dtrack-audit	9
Proprietary Products	11
Sonatype Nexus IQ	11
Sonatype Nexus Lifecycle Jenkins Plugin	11
CyberProtek	12
MedScan	12
Reliza Hub	13

<http://tiny.cc/CycloneDX>

<http://tiny.cc/SPDX>

Information to Collect per Tool

Tool Template

Support	Produce, Consume, Transform
Functionality	
Location	Website: Source:
Installation instructions	
How to use	
Versions Supported	

Example: FOSSology

Support	Produce (Analyze, Edit!), Consume(View,Diff,Import), Transform(Translate, Merge, Tool Support)
Functionality	<p>FOSSology is an open source license compliance software system and toolkit allowing users to run license, copyright and export control scans from a REST API.</p> <p>As a system, a database and web UI are provided to provide a compliance workflow.</p> <p>As part of the toolkit multiple license scanners, copyright and export scanners are tools available to help with compliance activities.</p>
Location	Website: https://www.fossology.org/ Source: https://github.com/fossology
Installation instructions	https://www.fossology.org/get-started/
How to use	https://www.fossology.org/get-started/basic-workflow/
Versions Supported:	SPDX 2.1, SPDX 2.2

How?

- Common **Processes and Norms** for sharing SBOM Information
- Common **Data Formats** for exchanging SBOM Information
- Commonly available, simple to use **Tooling** for effective & efficient communication




OpenChain Specification

Core of the OpenChain Project:

- Current version ISO/IEC 5230:2020.
- Identifies a minimum level of processes that organizations of any size can use to address open source compliance issues effectively
- Developed by a broad base of corporate and community participants
- **Translations** in Chinese (Simplified, Traditional), Japanese, Korean, Portuguese, Spanish, Italian, Polish & German available.



		OpenChain Specification 2.0
<hr/>		
Contents		
1) Introduction		3
2) Definitions		4
3) Requirements		5
1.0 Program Foundation		5
2.0 Relevant Tasks Defined and Supported		7
3.0 Open Source Content Review and Approval		8
4.0 Compliance Artifact Creation and Delivery		9
5.0 Understanding Open Source Community Engagements		10
6.0 Adherence to the Specification Requirements		11
Appendix I: Language Translations		12
<small>Copyright © 2016-2019 Linux Foundation. This document is licensed under the Creative Commons Attribution 4.0 International [CC-BY 4.0] license. A copy of the license can be found at https://creativecommons.org/licenses/by/4.0/</small>		
<hr/>		<small>2019 C</small>
<hr/>		<small>Page 2 of 12</small>

Software Package Data Exchange® (SPDX®) specification is a standard for communicating the component and metadata information associated with software

Charter: To create a set of **data exchange** standards that enable companies and organizations to share human-readable and machine-processable **software package metadata** to facilitate **software supply chain processes**.

SPDX v2.2.1 Document may contain:

Document Creation Information

Package Information

File Information

Snippet Information

Other Licensing Information

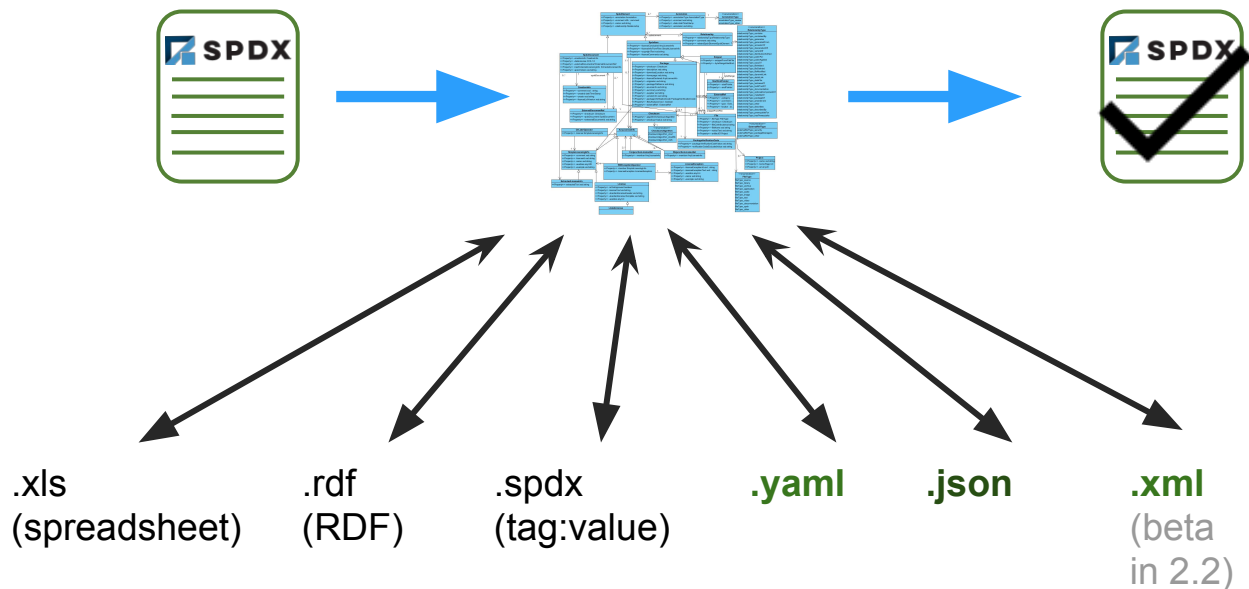
Relationships

Annotations

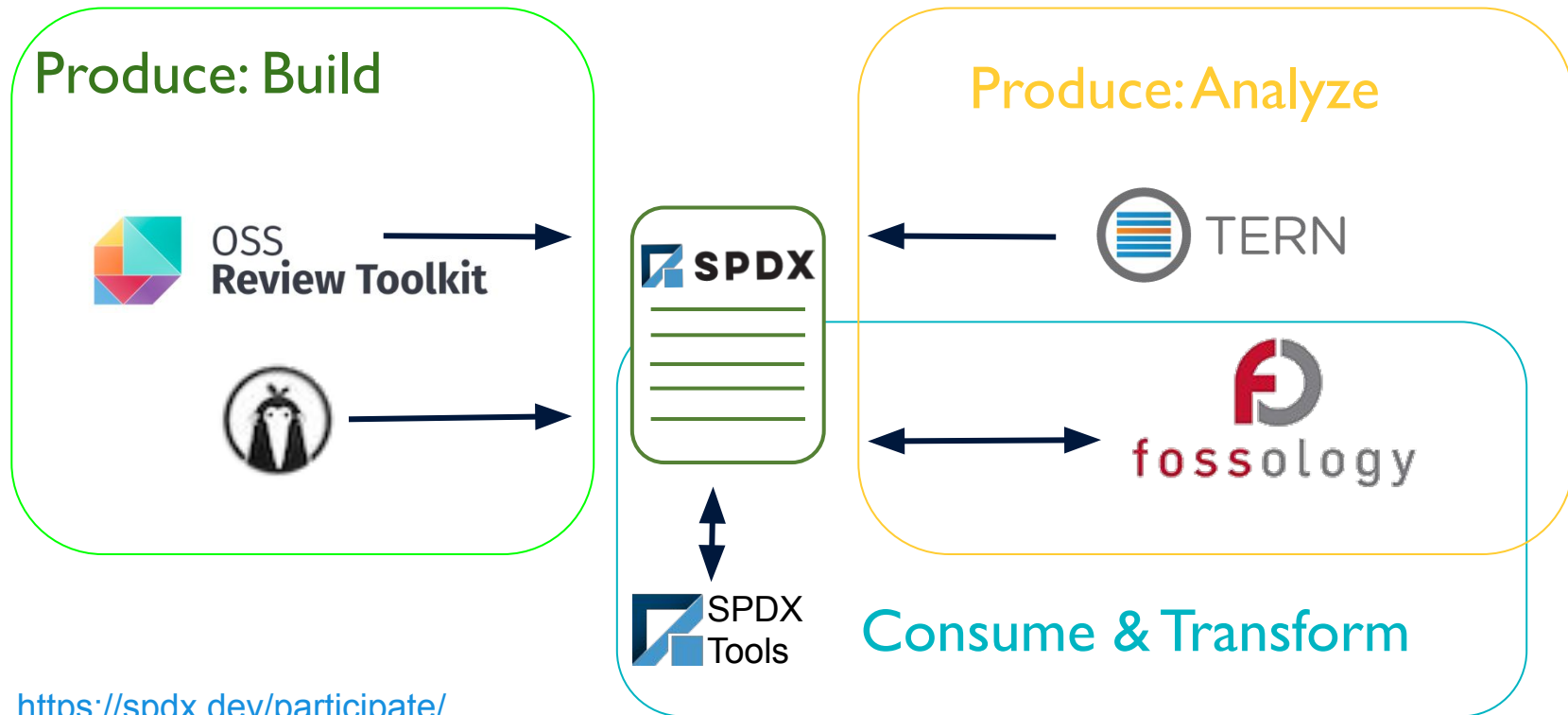
SPDX Specification History

- **2010/02** - specification drafting began in a work-group of FOSSBazaar under Linux Foundation that came to be called "SPDX".
- **2010/08** - "SPDX" announced as one of the pillars of the Linux Foundation's Open Compliance Program.
- **2011/08** - SPDX 1.0 specification - handles packages.
- **2012/08** - SPDX 1.1 specification - fixed flaw in verification algorithm
- **2013/10** - SPDX 1.2 specification - improved interaction with license list, additional fields for documenting project info.
- **2015/05** - SPDX 2.0 specification - added ability to handle multiple packages, relationships between packages and files, annotations.
- **2016/08** - SPDX 2.1 specification - added snippets, support for associating packages with external reference sources of information about packages, using SPDX License identifiers in files
- **2019/06** - SPDX 2.1.1 - conversion of specification from google docs to github as repository
- **2020/05** - SPDX 2.2 - include SPDX-Lite, more relationships, etc.
- **2020/08** - SPDX 2.2.1 - reformatting and prepared for submission to ISO.
- **2021/03** - ISO DIS balloting completed - specification Approved.

Underlying Model Supports File Formats




SPDX is being used as a common exchange standard



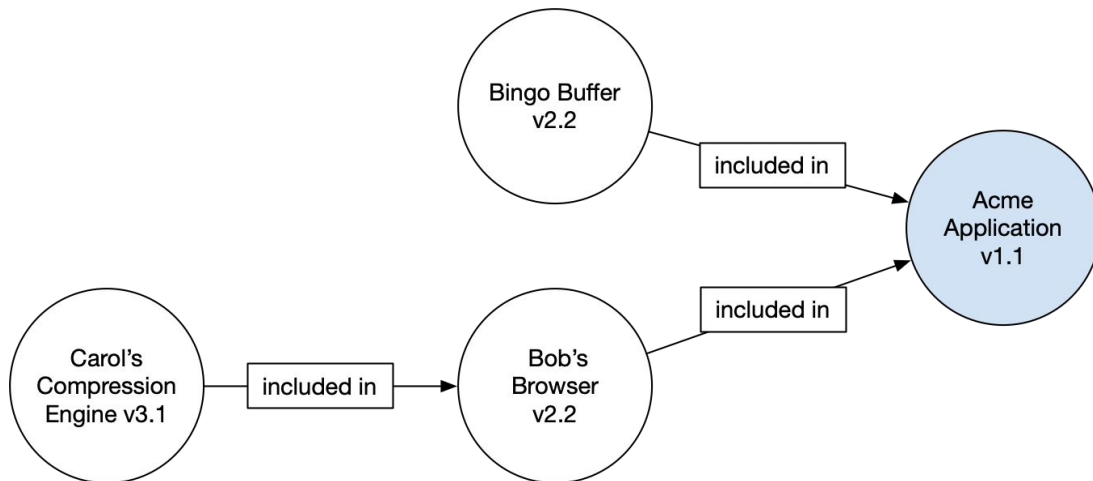
<https://spdx.dev/participate/>

SPDX Tools

Know a tool not
listed, feel free to
add
<http://tiny.cc/SPDX>

	
Format Overview	2
Format Publishing History	2
Tool Classification Taxonomy	2
Open Source Tools	4
Augur	4
FOSSology	4
in-toto	5
kernel-spdx-ids	5
npm-spdx	6
Open Source Software Review Toolkit (ORT)	6
OWASP Dependency-Track	6
Quartermaster (QMSTR)	7
REUSE	8
ScanCode Toolkit	8
SPDX Java Libraries and Tools	9
SPDX Python Libraries	10
SPDX Golang Libraries	10
SPDX JavaScript Libraries	11
SPDX Online Tools	11
SPDX Maven Plugin	12
SPDX Build Tool	12
SPARTS	12
SW360	13
TERN	13
Yocto Project / OpenEmbedded	14
Proprietary Products	15
CyberProtek	15
FOSSID	15
Hub-SPDX (Black Duck Hub Report Utility)	16
MedScan	16
Protecode	17
Protex	17
SourceAuditor	17
TrustSource	18
Vigilant-ops	18

How to start?



Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship
Application	Acme	1.1	Acme	0x123	234	Self
--- Browser	Bob	2.1	Bob	0x223	334	Included in
--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in
--- Buffer	Bingo	2.2	Acme	0x423	534	Included in

SwiftBOM- SBOM generator for demo and PoCs

sbom.democert.org/sbom/

SwiftBOM - SBOM Generator for PoC and Demos [Load Example](#) | [Import SPDX/Excel](#) Clear i 🗣

Document Name

Document Namespace

Creator Organization ▾

Created 📅

Creator Comment

Draft ACME INFUSION PoC II SBOM document in SPDX format. Unofficial content for demonstration purposes only

Primary Component + SPDX Lite ?

Component Name ■ Use CPE

Version

Supplier Name Organization ▾

Relationship Primary ▾

+ Add Component Reset Generate SBOM



SPDX ↓

SWID ↓

CycloneDX ↓

Graph ↓

Z ↓

Vulnerabilities

Close



Component1 (Browser)

Parent Component

PrimaryComponent (Application)

Browser

Compression

Child BOM exists

Component Name

Browser

Application

Version

2.2

Supplier Name

Buffer

Organization

Bob

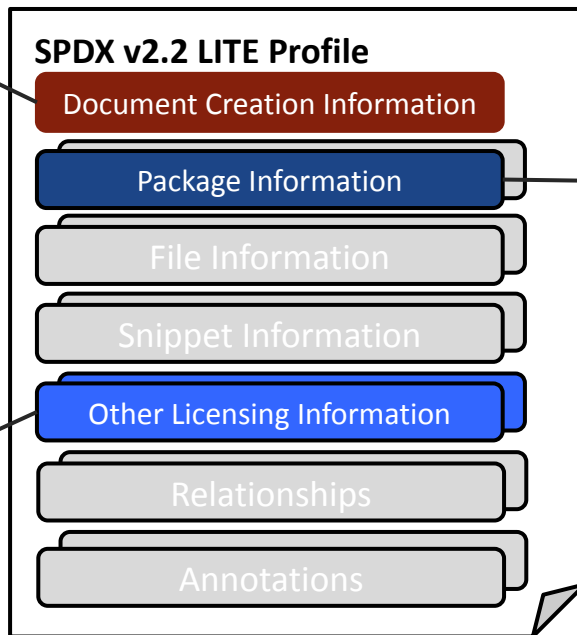
Relationship

Included

SPDX 2.2 Appendix VIII: **SPDX Lite Profile**

SPDX #	Field Name
2.1	SPDX Version
2.2	Data License
2.3	SPDX Identifier
2.4	Document Name
2.5	SPDX Document Namespace
2.8	Creator
2.9	Created

SPDX #	Field Name
6.1	License Identifier
6.2	Extracted Text
6.3	License Name
6.5	License Comment



SPDX #	Field Name
3.1	Package Name
3.2	Package SPDX Identifier
3.3	Package Version
3.4	Package File Name
3.7	Package Download Location
3.8	Files Analyzed
3.11	Package Home Page
3.13	Concluded License
3.15	Declared License
3.16	Comments on License
3.17	Copyright Text
3.20	Package Comment

SwiftBOM- SBOM generator - SPDX-Lite option

sbom.democert.org/sbom/

SwiftBOM - SBOM Generator for PoC and Demos [Load Example](#) [Import SPDX License](#) [Clear](#)

Document Name:

Document Namespace:

Creator:

Created:

Creator Comment: [Add optional SPDX info](#)

[Primary Component](#) [SPDX Lite](#)

Component Name:

Version:

Supplier Name:

Relationship:

Package File Name:

Package Download Location:

Files Analyzed:

Package Home Page:

Concluded License:

Declared License:

Comments on License:

Copyright Text:

License Identifier:

Extracted Text:

License Name:

License Comment:

[+ Add Component](#) [Cancel](#) [Generate SBOM](#)

SPDX Online Tools - Validate & Convert



Check it out at: <https://tools.spdx.org/>

More SPDX examples...

<https://github.com/swinslow/spdx-examples>

SPDX Examples

This repository includes demonstrations of [SPDX documents](#) for various examples of software combinations.

The examples include source code and built / packaged binaries for a variety of scenarios. The software in the repository is (for the most part) not taken from real projects. However, the examples are intended to be demonstrations of how SPDX can convey software bill of materials (SBOM) information for a variety of real-world scenarios.

Format of examples

Each example directory is structured as follows:

- `content/src/` : contains the example's source code
- `content/build/` : contains the example's built artifacts
- `spdx/` : contains one or more tag-value SPDX documents for the sources and the build artifacts
- `README.md` : more details about the particular example

Each directory contains a Makefile which is used to create the build artifacts. It assumes that the necessary tools (make, gcc, etc.) are present on your system, and doesn't do any autoconfiguration or the like. If somebody else wants to add that for greater build flexibility, they are welcome to do so, but that isn't really my goal here :)

Examples

#	Sources	Binaries	SPDX	Comments
1	1 C file	compiled with gcc	1 document	source and binary treated as one package
2	1 C file	compiled with gcc	2 documents	source and binary in separate packages
3	2 C files	compiled with gcc	2 documents	shared library, dynamically linked at runtime
4	2 C files	compiled with gcc	2 documents	shared library, dynamically linked at runtime, including system libs
5	2 Go files	compiled with go	2 documents	source and binary in separate packages
6	2 Go files	compiled with go	3 documents	source and binary in separate packages, separate doc for standard libs

<https://github.com/lfscanning>



lfscanning

Repositories 14 Packages People Projects

Find a repository...

Type

Sort

spdx-onap

SPDX files from LF license scans for ONAP repos

CC0-1.0 0 0 1 0 Updated 13 days ago



spdx-lfenergy

SPDX files from LF license scans for LF Energy repos

CC0-1.0 0 0 0 0 Updated 17 days ago



spdx-lfai

SPDX files from LF license scans for LFAI repos

CC0-1.0 0 0 0 0 Updated 29 days ago



spdx-acumos

SPDX files from LF license scans for Acumos repos

CC0-1.0 0 0 0 0 Updated on Feb 2



spdx-omp

SPDX files from LF license scans for OMP repos

CC0-1.0 0 0 1 0 Updated on Jan 15



Benefits from Adopting SBOMs

- › Identifying and avoiding known **vulnerabilities**
- › Quantifying and managing **licenses**
- › Identifying both security and license **compliance requirements**
- › Enabling **quantification of the risks** inherent in a software package
- › **Managing mitigations** for vulnerabilities (including patching and compensating controls for new vulnerabilities)
- › **Lower operating costs** due to improved efficiencies and reduced unplanned and unscheduled work.

These benefits can be seen by those who develop software, those who select or purchase software, and those who operate software, across every sector.



Thank you for joining us today!

We hope it will be helpful in your journey to learning more about effective and productive participation in open source projects. We will leave you with a few additional resources for your continued learning:

- The [LF Mentoring Program](#) is designed to help new developers with necessary skills and resources to experiment, learn and contribute effectively to open source communities.
- [Outreachy remote internships program](#) supports diversity in open source and free software
- [Linux Foundation Training](#) offers a wide range of [free courses](#), webinars, tutorials and publications to help you explore the open source technology landscape.
- [Linux Foundation Events](#) also provide educational content across a range of skill levels and topics, as well as the chance to meet others in the community, to collaborate, exchange ideas, expand job opportunities and more. You can find all events at events.linuxfoundation.org.