SBOMs and their Role in Security

Alexios Zavras

About myself

- Greek, living in Munich
- Intel's Chief Open Source Compliance Officer
- Free and Open Source Software since 1983
- Long-term, big-picture view
- Working on SBOM issues for 15+ years (SPDX, OpenChain, …)
- Organizer of FOSDEM SBOM devroom
- PhD in Computer Science, having studied in Greece and USA

Software is complex

- Nowadays almost always a combination of components
- 80 – 20 rule

SBOM

Software Bill of Materials

Software Bill of Materials (SBOM)

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

- Components include libraries and modules
- Components can be open source or proprietary
- Components can be freely available or paid
- Data can be widely available or access-restricted

Who should use an SBOM?

- Any organization concerned about better supporting their software products internally and better supporting their customers
- Different views
  - Produce / Consume (Use / Integrate)
- 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

Why have an SBOM?

- Legal compliance
  - License obligations, Open Source or not
  - Comply with *all* obligations of *all* licenses of *all* components
    - Straightforward
    - But not trivial or easy
- Export
- Security

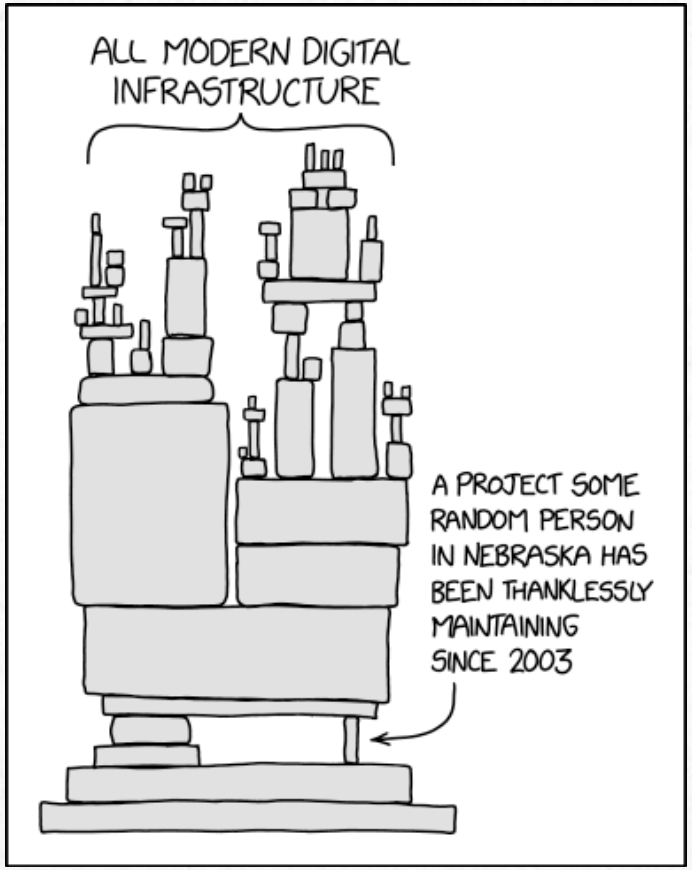
Why have an SBOM?

- Legal compliance
- Export
- **Security**

Do YOU know...

... whether you are affected by $VULNERABILITY?

Most do not know what software is running



Dependency, by xkcd, CC-BY-NC-2.5

Regulation

Regulation is ~~coming~~ here!

- EU
  - Cyber Resilience Act (CRA)
- US
  - EO 14028 on Improving the Nation’s Cybersecurity; May 2021
  - National Cybersecurity Strategy Implementation Plan; July 2023
- Germany
- Japan
- …

Need a Bill of Materials

A comprehensive list of software components with information on:

• Name	zlib	gcc
• License	Zlib license	GPLv3
• Version	1.3.2	16.1
• Origin	`https://zlib.net`	`https://gcc.gnu.org`
• …		“not modified”

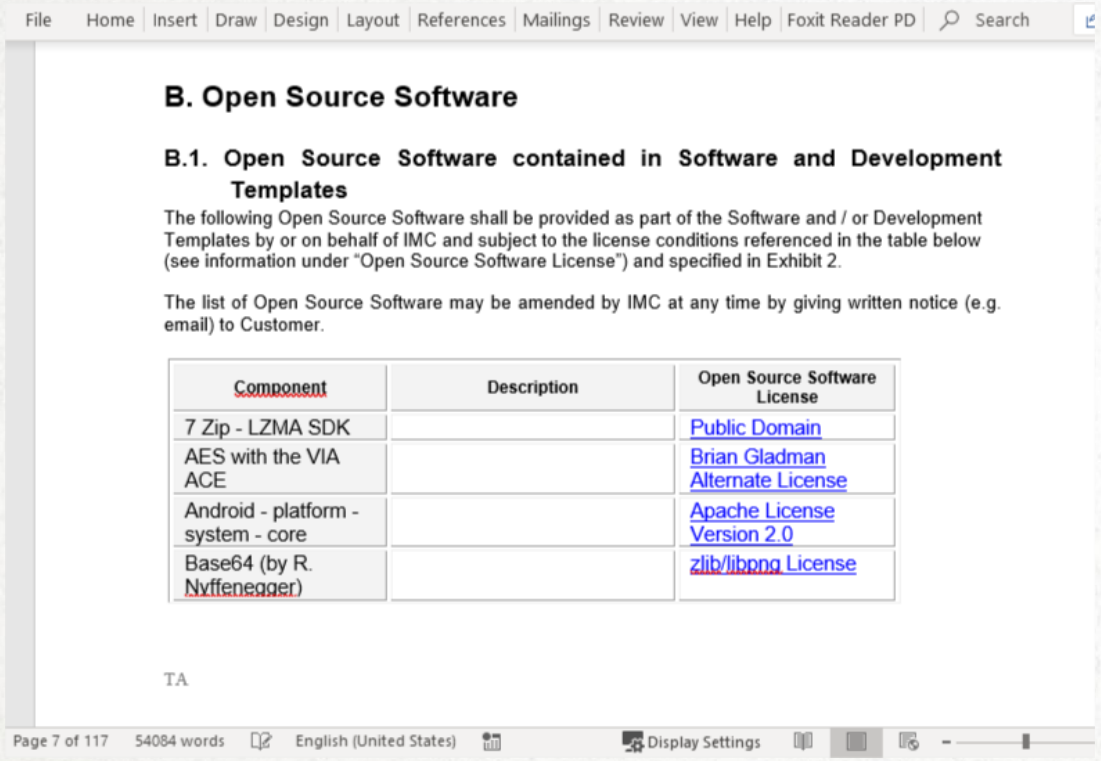
Contents of a minimum viable SBOM

Data Field	Description
Supplier Name	The name of an entity that creates, defines, and identifies components.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y.
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
Timestamp	Record of the date and time of the SBOM data assembly.

The Minimum Elements for an SBOM, by US Department of Commerce

How to deliver this information?

How to deliver this information?



File Home Insert Draw Design Layout References Mailings Review View Help Foxit Reader PD Search

B. Open Source Software

B.1. Open Source Software contained in Software and Development Templates

The following Open Source Software shall be provided as part of the Software and / or Development Templates by or on behalf of IMC and subject to the license conditions referenced in the table below (see information under "Open Source Software License") and specified in Exhibit 2.

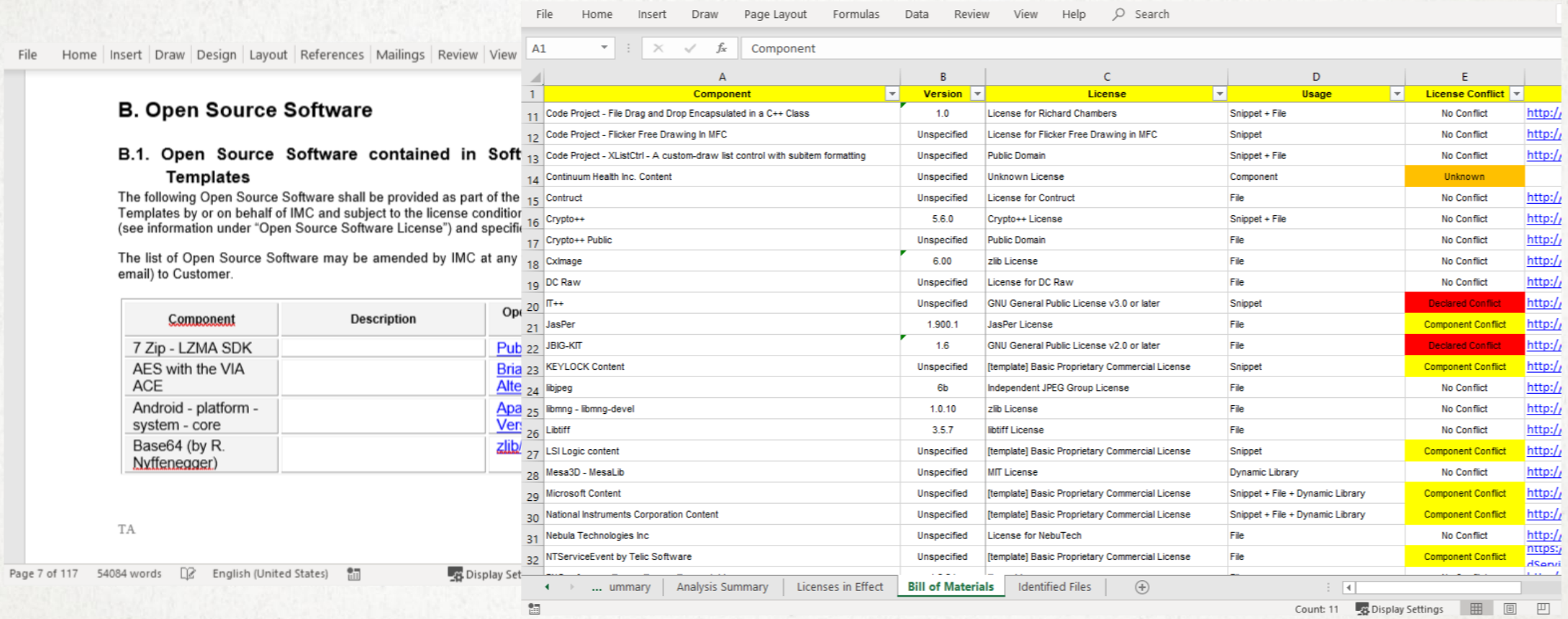
The list of Open Source Software may be amended by IMC at any time by giving written notice (e.g. email) to Customer.

Component	Description	Open Source Software License
7 Zip - LZMA SDK		Public Domain
AES with the VIA ACE		Brian Gladman Alternate License
Android - platform - system - core		Apache License Version 2.0
Base64 (by R. Nyffenegger)		zlib/libpng License

TA

Page 7 of 117 54084 words English (United States) Display Settings

How to deliver this information?



File Home Insert Draw Design Layout References Mailings Review View

B. Open Source Software

B.1. Open Source Software contained in Soft
Templates

The following Open Source Software shall be provided as part of the
Templates by or on behalf of IMC and subject to the license conditio
(see information under "Open Source Software License") and specifi

The list of Open Source Software may be amended by IMC at any
email) to Customer.

Component	Description	Op
7 Zip - LZMA SDK		Pub
AES with the VIA ACE		Bria Alte
Android - platform - system - core		Apa Ver
Base64 (by R. Nyffenegger)		zlib

TA

Page 7 of 117 54084 words English (United States) Display Set

File Home Insert Draw Page Layout Formulas Data Review View Help Search

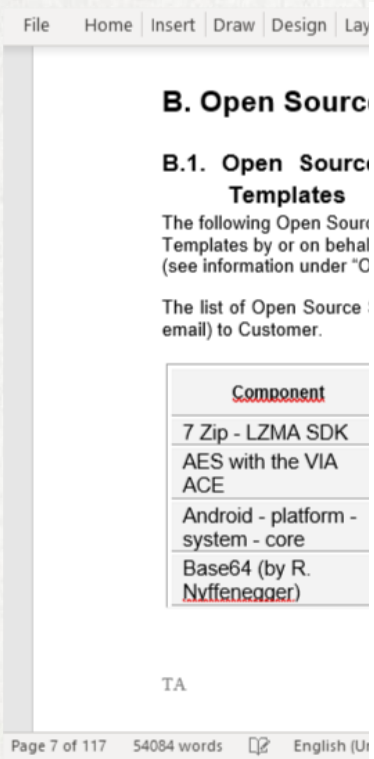
A1 Component

	A	B	C	D	E	
1	Component	Version	License	Usage	License Conflict	
11	Code Project - File Drag and Drop Encapsulated in a C++ Class	1.0	License for Richard Chambers	Snippet + File	No Conflict	http:/
12	Code Project - Flicker Free Drawing In MFC	Unspecified	License for Flicker Free Drawing in MFC	Snippet	No Conflict	http:/
13	Code Project - XListCtrl - A custom-draw list control with subitem formatting	Unspecified	Public Domain	Snippet + File	No Conflict	http:/
14	Continuum Health Inc. Content	Unspecified	Unknown License	Component	Unknown	
15	Contruct	Unspecified	License for Contruct	File	No Conflict	http:/
16	Crypto++	5.6.0	Crypto++ License	Snippet + File	No Conflict	http:/
17	Crypto++ Public	Unspecified	Public Domain	File	No Conflict	http:/
18	CxImage	6.00	zlib License	File	No Conflict	http:/
19	DC Raw	Unspecified	License for DC Raw	File	No Conflict	http:/
20	IT++	Unspecified	GNU General Public License v3.0 or later	Snippet	Declared Conflict	http:/
21	JasPer	1.900.1	JasPer License	File	Component Conflict	http:/
22	JBIG-KIT	1.6	GNU General Public License v2.0 or later	File	Declared Conflict	http:/
23	KEYLOCK Content	Unspecified	[template] Basic Proprietary Commercial License	Snippet	Component Conflict	http:/
24	libjpeg	6b	Independent JPEG Group License	File	No Conflict	http:/
25	libmng - libmng-devel	1.0.10	zlib License	File	No Conflict	http:/
26	Libtiff	3.5.7	libtiff License	File	No Conflict	http:/
27	LSI Logic content	Unspecified	[template] Basic Proprietary Commercial License	Snippet	Component Conflict	http:/
28	Mesa3D - MesaLib	Unspecified	MIT License	Dynamic Library	No Conflict	http:/
29	Microsoft Content	Unspecified	[template] Basic Proprietary Commercial License	Snippet + File + Dynamic Library	Component Conflict	http:/
30	National Instruments Corporation Content	Unspecified	[template] Basic Proprietary Commercial License	Snippet + File + Dynamic Library	Component Conflict	http:/
31	Nebula Technologies Inc	Unspecified	License for NebuTech	File	No Conflict	http:/
32	NTServiceEvent by Telic Software	Unspecified	[template] Basic Proprietary Commercial License	File	Component Conflict	https:

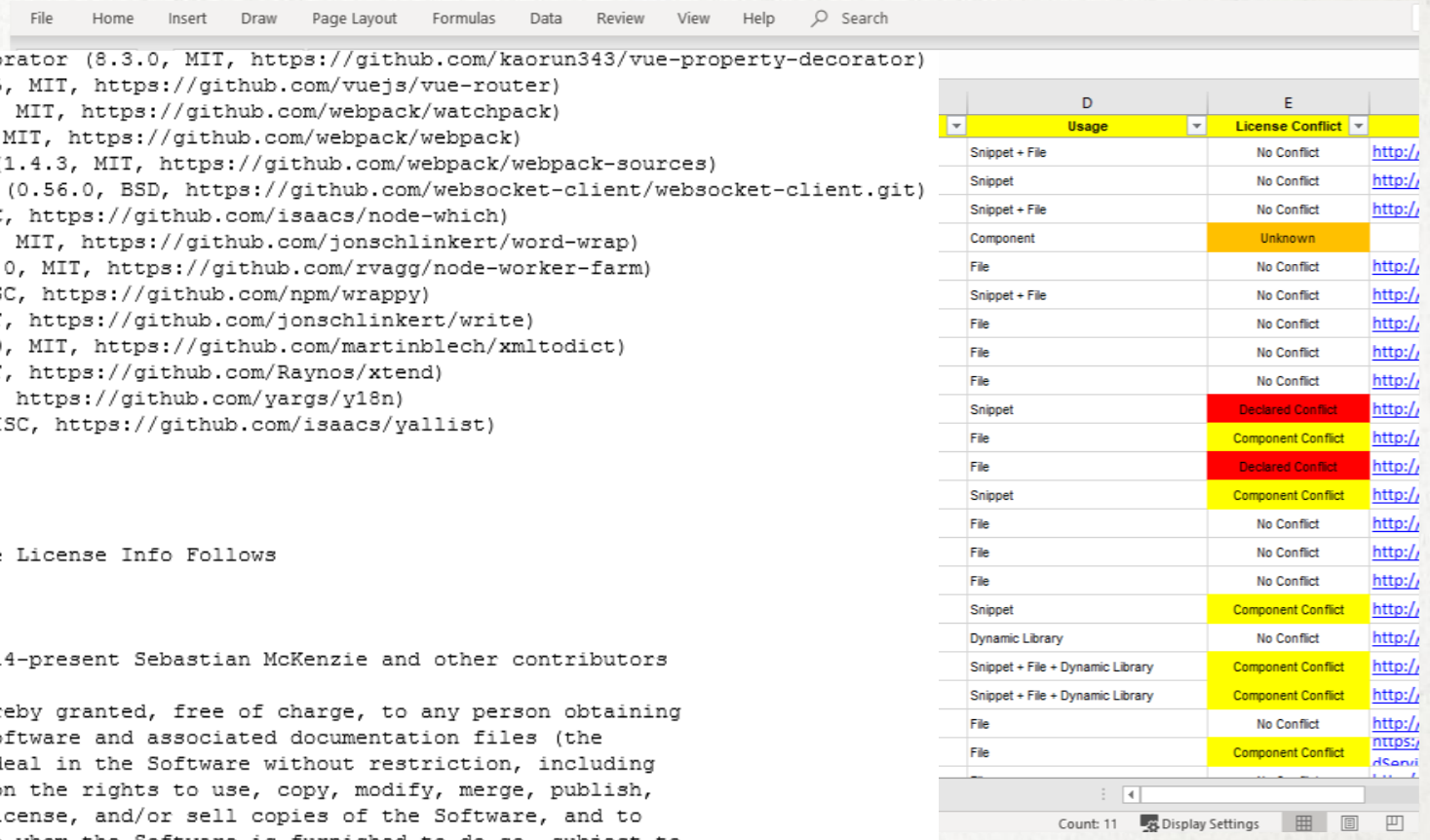
... ummary | Analysis Summary | Licenses in Effect | Bill of Materials | Identified Files

Count: 11 Display Settings

How to deliver this information?




```
439.    vue-property-decorator (8.3.0, MIT, https://github.com/kaorun343/vue-property-decorator)
440.    vue-router (3.1.5, MIT, https://github.com/vuejs/vue-router)
441.    watchpack (1.6.0, MIT, https://github.com/webpack/watchpack)
442.    webpack (4.41.5, MIT, https://github.com/webpack/webpack)
443.    webpack-sources (1.4.3, MIT, https://github.com/webpack/webpack-sources)
444.    websocket-client (0.56.0, BSD, https://github.com/websocket-client/websocket-client.git)
445.    which (1.3.1, ISC, https://github.com/isaacs/node-which)
446.    word-wrap (1.2.3, MIT, https://github.com/jonschlinkert/word-wrap)
447.    worker-farm (1.7.0, MIT, https://github.com/rvagg/node-worker-farm)
448.    wrappy (1.0.2, ISC, https://github.com/npm/wrappy)
449.    write (1.0.3, MIT, https://github.com/jonschlinkert/write)
450.    xmltodict (0.12.0, MIT, https://github.com/martinblech/xmltodict)
451.    xtend (4.0.2, MIT, https://github.com/Raynos/xtend)
452.    y18n (4.0.0, ISC, https://github.com/yargs/y18n)
453.    yallist (3.1.1, ISC, https://github.com/isaacs/yallist)



# 0.    @babel/code-frame License Info Follows

      MIT License

      Copyright (c) 2014-present Sebastian McKenzie and other contributors

      Permission is hereby granted, free of charge, to any person obtaining
      a copy of this software and associated documentation files (the
      "Software"), to deal in the Software without restriction, including
      without limitation the rights to use, copy, modify, merge, publish,
      distribute, sublicense, and/or sell copies of the Software, and to
      permit persons to whom the Software is furnished to do so, subject to
      the following conditions:

      The above copyright notice and this permission notice shall be
      included in all copies or substantial portions of the Software
```



Standardization

- Standard format
- Can be used by everyone
- Clear, unambiguous
- Automatically produced and processed

“The good thing about standards is that there are so many to choose from.”

— **Andrew S. Tanenbaum**

SBOM standards

SPDX

- Older
  - started in Feb 2010
  - first release Aug 2011
  - latest release 3.0
- Initially compliance-focused
- ISO/IEC 5962:2021 (and :2026)

Cyclone DX

- Newer
  - started in May 2017
  - first release Mar 2018
  - latest release 1.7
- Initially security-focused
- ECMA-424

Format wars are not interesting

- SPDX vs CycloneDX
- Largely inter-operable (for basic information)
- Constantly evolving
- I have seen (and participated)
  - UNIX wars
  - BSD wars
  - Programming language wars
  - Editor wars
  - SBOM wars

Naming

One of the hardest problem in CS

The problem

- How do we refer to a software component?
  - “OpenSSL”
  - “OpenSSL 3.4.5”
  - “OpenSSL 3.4.5 from the project releases on GitHub”
  - “OpenSSL 3.4.5 from Debian”
  - “OpenSSL 3.4.5 from Ubuntu”
- Different ecosystems
- Imprecise wording (ironic)
  - Names, Locators, Identifiers, …

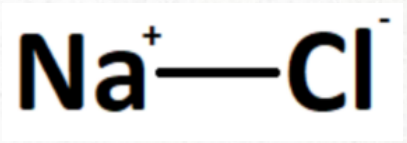
Major distinction

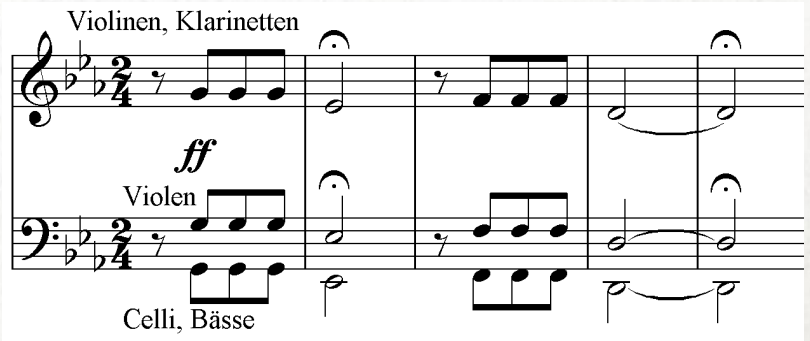
Extrinsic

- *Registry* keeps correspondence to object
- URL, DOI, PURL
- Physical examples:

Intrinsic

- Connection to object in the identifier
- SWHID
- Examples:





PURL

Package URL

What is PURL?

- An open standard for uniquely identifying software packages across different ecosystems
- Created in 2017 by Philippe Ombredanne
- Standardized as ECMA-427
- URL-based syntax

Anatomy of a PURL

`pkg:<type>/<namespace>/<name>@<version>?<qualifiers>#<subpath>`

- scheme “pkg”
- type
- namespace (optional)
- name
- version (optional)
- qualifiers (optional)
- subpath (optional)

PURL types and examples

Type	Example
PyPI	`pkg:pypi/django@4.2.5`
NPM	`pkg:npm/%40angular/core@16.2.0`
Maven	`pkg:maven/org.apache.logging.log4j/log4j-core@2.20.0`
Deb	`pkg:deb/debian/curl@7.50.3-1?arch=i386&distro=jessie`
...	and many other types

SWHID

Software Hash IDentifier

What is a SWHID?

- A persistent, intrinsic identifier for source code artifacts
- Created initially by Software Heritage
- Works like a digital fingerprint
  - computed directly from the code's content, not from metadata
- Officially standardized as ISO/IEC 18670:2025 (April 2025)

Anatomy of a SWHID

`swh:1:dir:df32c75242bf8d797ccd43af8ce8e294f35cd8fd`

- prefix `swh`
- version `1`
- object type `dir`
- content hash

- Optional qualifiers to add more info, like origin, lines, etc.

SWHID types

- Five object types
- Content `cnt` (file)
- Directory `dir` (tree)
- Revision `rev` (commit)
- Release `rel` (tag)
- Snapshot `snp` (repo)

Use cases (coverage)

Since it's an intrinsic identifier, it can be used for everything:

- `gcc-16.1`
- gcc snapshot of revision `3050eeee`
- `gcc-16.1` patched with `patches.tar.gz`
- `custom-sw.tar.gz`

SPDX

~~Software~~ System Package Data Exchange

The SPDX project

- Represent ~~Software~~ System Bill of Materials efficiently
  - Specification
  - License List
  - Code for low-level libraries
- Teams: technical, legal, outreach
- Working groups: per area of interest (implementers, SaaS, cryptography, …)
- Community-led project
  - Open collaboration
  - Complete transparency
  - Inclusive participation

Widespread industry support

All logos used with written permission

Long history of specification development

SPDX®

SPDX data

- Just facts, not interpretation
- Graph-based
  - In contrast to Document-based SPDXv2
- Model in RDF
- Serialized in variety of formats (JSON-LD, triples, XML, …)
- Allows incremental addition of information

Profiles


SPDX CORE


SPDX SOFTWARE


SPDX SECURITY


SPDX LICENSING


SPDX AI


SPDX DATA


SPDX BUILD


SPDX LITE

Upcoming profiles

Short RDF primer

Resource Description Framework

What is RDF?

- A W3C standard for describing resources on the web as structured data
- Models information as a directed graph of relationships between things
- Foundation of the Semantic Web and Linked Data movement
  - Machine-readable; Decentralized; Extensible; Interoperable
- First published in 1999

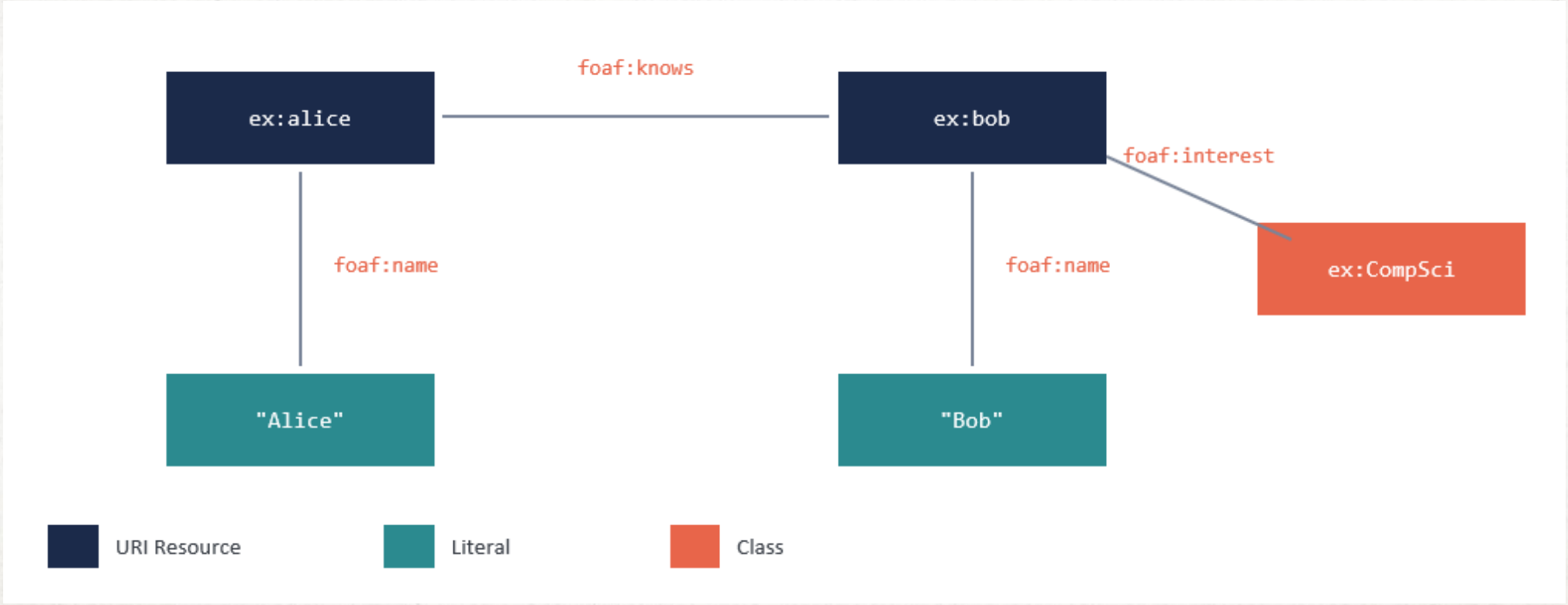
RDF triples

- Every piece of information is expressed as a triple:
  - Subject (URI or blank node)
  - Predicate (URI)
  - Object (literal, URI, or blank node)

```
<http://example.org/alice>  <http://xmlns.com/foaf/0.1/knows>  <http://example.org/bob> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex:   <http://example.org/> .

ex:alice  foaf:knows  ex:bob .
```

Graph data


foaf:knows
ex:alice
ex:bob
foaf:interest
foaf:name
foaf:name
ex:CompSci
"Alice"
"Bob"
URI Resource
Literal
Class

Serializations

- RDF data can be serialized in various formats:
  - Turtle
  - JSON-LD
  - RDF/XML
  - N-triples
  - ...

Serialization examples

Turtle

```
@prefix foaf:
            <http://xmlns.com/foaf/0.1/> .
@prefix ex:   <http://example.org/> .


ex:alice
    a           foaf:Person ;
    foaf:name   "Alice" ;
    foaf:age    30 ;
    foaf:knows  ex:bob .

ex:bob
    a           foaf:Person ;
    foaf:name   "Bob" .
```

JSON-LD

```
{
  "@context": {
    "foaf": http://xmlns.com/.../0.1/
  },
  "@id": "http://example.org/alice",
  "@type": "foaf:Person",
  "foaf:name": "Alice",
  "foaf:knows": {
    "@id": http://example.org/bob
  }
}
```

Security content

Security-related data

- Vulnerability
- Score
- Exploitation reports
- Justifications
- References and identifiers

Vulnerabilities and severities

- CVE
- CWE
- Metrics (scores)
  - CVSS (Common Vulnerability Scoring System)
    - v2, v3, v4
  - EPSS (Exploit Prediction Scoring System)
  - SSVC (Stakeholder-Specific Vulnerability Categorization)

Exploitability

VEX (Vulnerability Exploitability eXchange)

- affected
- not affected
- fixed
- under investigation

Justifications

- componentNotPresent
  - The software is not affected because the vulnerable component is not in the product.
- inlineMitigationsAlreadyExist
  - Built-in inline controls or mitigations prevent an adversary from leveraging the vulnerability.
- vulnerableCodeCannotBeControlledByAdversary
  - The vulnerable component is present, and the component contains the vulnerable code. However, vulnerable code is used in such a way that an attacker cannot mount any anticipated attack.
- vulnerableCodeNotInExecutePath
  - The affected code is not reachable through the execution of the code, including non-anticipated states of the product.
- vulnerableCodeNotPresent
  - The product is not affected because the code underlying the vulnerability is not present in the product

References

Ways to refer to external identifiers

- cpe22 (Common Platform Enumeration v2.2)
- cpe23 (Common Platform Enumeration Naming v2.3)
- cve (Common Vulnerabilities and Exposures)
- ecli (European Case Law Identifier)
- eli (European Legislation Identifier)
- swhid (SoftWare Hash IDentifier)

Much more…

- riskAssessment
- secureSoftwareAttestation
- securityAdversaryModel
- securityAdvisory
- securityFix
- securityPenTestReport
- securityPolicy
- securityThreatModel
- vulnerabilityDisclosureReport (VDR)

More importantly

- Security information changes over time
- Static SBOM content not useful
- Dynamic updates required

Other data

Information that is useful to be in an SBOM

Licensing information

- Every software component is released under a license
- May be open source or proprietary
- Licenses are combined into License Expressions
  - `AND`, `OR`, `WITH`, `+`
- Different relationships
  - `DeclaredLicense`
  - `ConcludedLicense`

AI information

- Generative AI Adoption at Breakneck Speed
- Traditional view: Source code, binaries, libraries, dependencies
- New reality: Models, datasets, training pipelines, inference engines
- Software assets now include:
  - Pre-trained models and fine-tuned variants
  - Training and validation datasets
  - Model weights and parameters
  - AI-generated code and content

SPDX is comprehensive

- Different “profiles” about various areas of interest
- Ever-expanding coverage

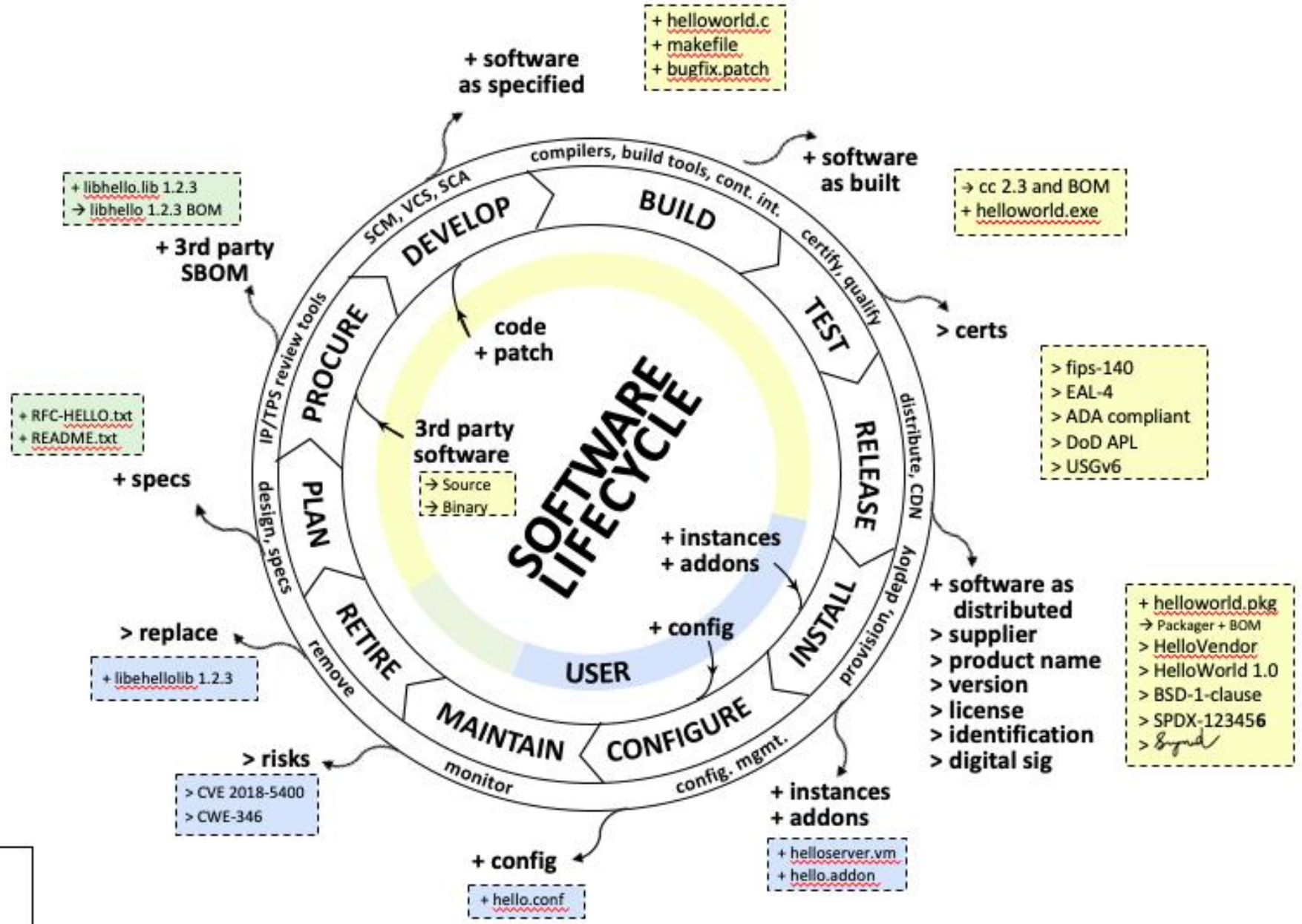
However, ...

Real life is not so simple

“Software” Bill of Materials

- What is “software”
- Many (conflicting) views

Software Lifecycle & Bill of Materials Generation



LEGEND

+ material
> metadata
→ reference
example BOM fragment
supplier
consumer
other

Types of SBOMs

Design	SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.
Source	SBOM created directly from the development environment, source files, and included dependencies used to build a product artifact.
Build	SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.
Analyzed	SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a "third-party" SBOM.
Deployed	SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.
Runtime	SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an "Instrumented" or "Dynamic" SBOM.

Tools

Tool functionality taxonomy

Category	Type	Description
Produce	Build	SBOM is automatically created as part of building a software 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 (e.g., picture, figures, tables, text, etc.). 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 combined together for analysis and audit purposes
	Tool support	Support use in other tools by APIs, object models, libraries, transport, or other reference sources

Tools classifications

- Licensed under:
  - Open Source
  - Proprietary
- SBOM Type
- List keeps expanding…
- Level:
  - Libraries
  - Purpose-specific
  - Complete applications
  - Integrated environments
- Ecosystem

Discussion

- Questions
- Comments
- Clarifications

Open issues

Not there yet

Scope of SBOMs

- For which products do we produce?
  - Proprietary?
  - Open Source?
  - Contributions?
- For which customers?
  - See “Distribution”, later
- When is it generated?
  - Upon release?
  - Daily?
  - CD?

SBOM delivery

- Still unspecified
- There is no one-way-fits-all
- How are SBOMs conveyed externally?
  - Delivered alongside software?
  - Stored on web site?
    - Publicly available?
  - Visibility and Access control
    - Are all SBOMs visible to everyone?
    - Are all legal texts (e.g., licenses) visible to everyone?

Legal implications of SBOMs

- What legal ramifications do SBOM statements have?

- Inaccurate information
  - Missing information
  - Erroneous information
  - Extraneous information

Thank you!