The State of SBoMs in Embedded

Joshua Watt

Embedded Linux Conference

June 23, 2025

yocto PROJECT

Copyright 2025 Joshua Watt, Creative Commons BY-SA 4.0 International License

About Me

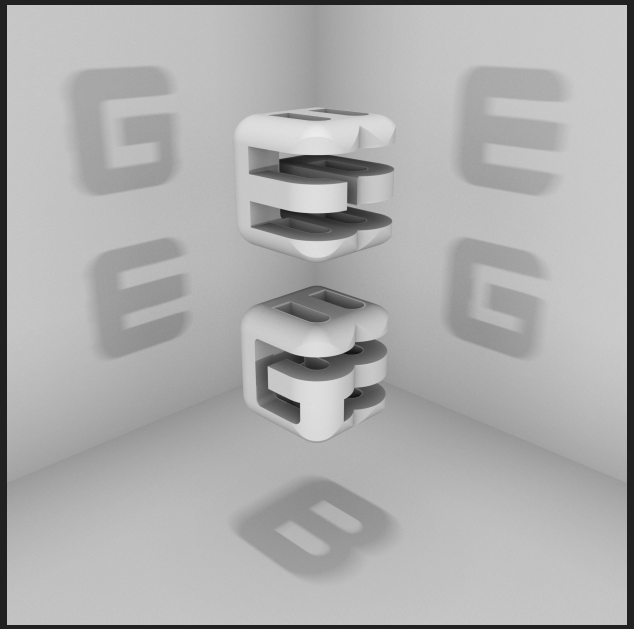
- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Member of the OpenEmbedded Technical Steering Committee (TSC)
- Member of Yocto TSC
- Member of the SPDX Technical Team
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- X/Twitter: @JPEW_dev
- LinkedIn: joshua-watt-dev

Disclaimer

- Opinions are my own
- My area of expertise is Yocto & SPDX
  - I'm not here to sell that today
  - Many of my examples will be from these
- My background is engineering Consumer Products

What is an SBoM, exactly?

And why do I want one?



Max Braun, CC BY-SA 2.0 via Wikimedia Commons

"Nutrition Information" for Software

Ingredients: bash, Linux, u-boot, sshd, openssl, busybox

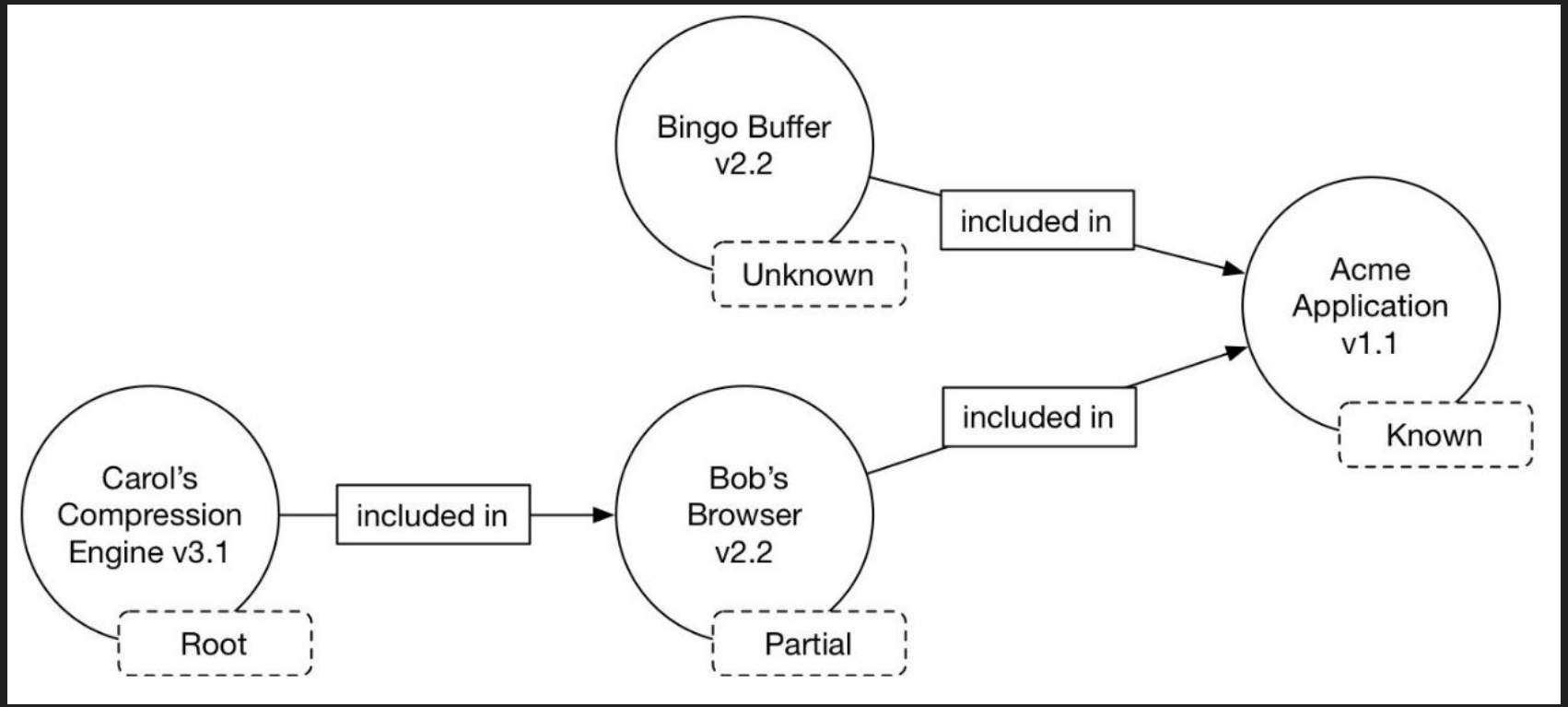
SBoM Facts

1 Serving per Device

Serving Size	**1**
CVEs Patched	**2**
CVE-2019-18276	
CVE-2014-0160	
Patches Applied	**30**
Calories from Bits	**10**

- Easily "digestible" and intuitive if you're unfamiliar with SBoMs
- Standardized format for information
- Misses out on the Supply Chain

A description of relationships between components



Source: NTIA's [Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM)]

Standardized Format for the Exchange of Information

Regulatory Agencies

Internal Compliance

Downstream 3rd Parties

Standardized Format for the Exchange of Inform

Do you have any vulnerabilities?

Regulatory

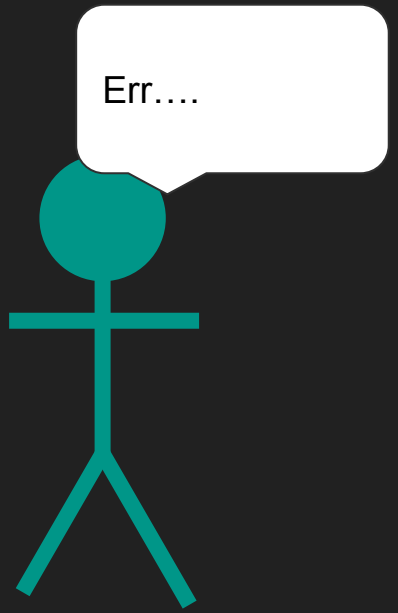
What Licenses are we using?

Internal

What's in the software you gave us?

Downstream 3rd Parties

Standardized Format for the Exchange of Inform



Do you have any vulnerabilities?

Regulatory

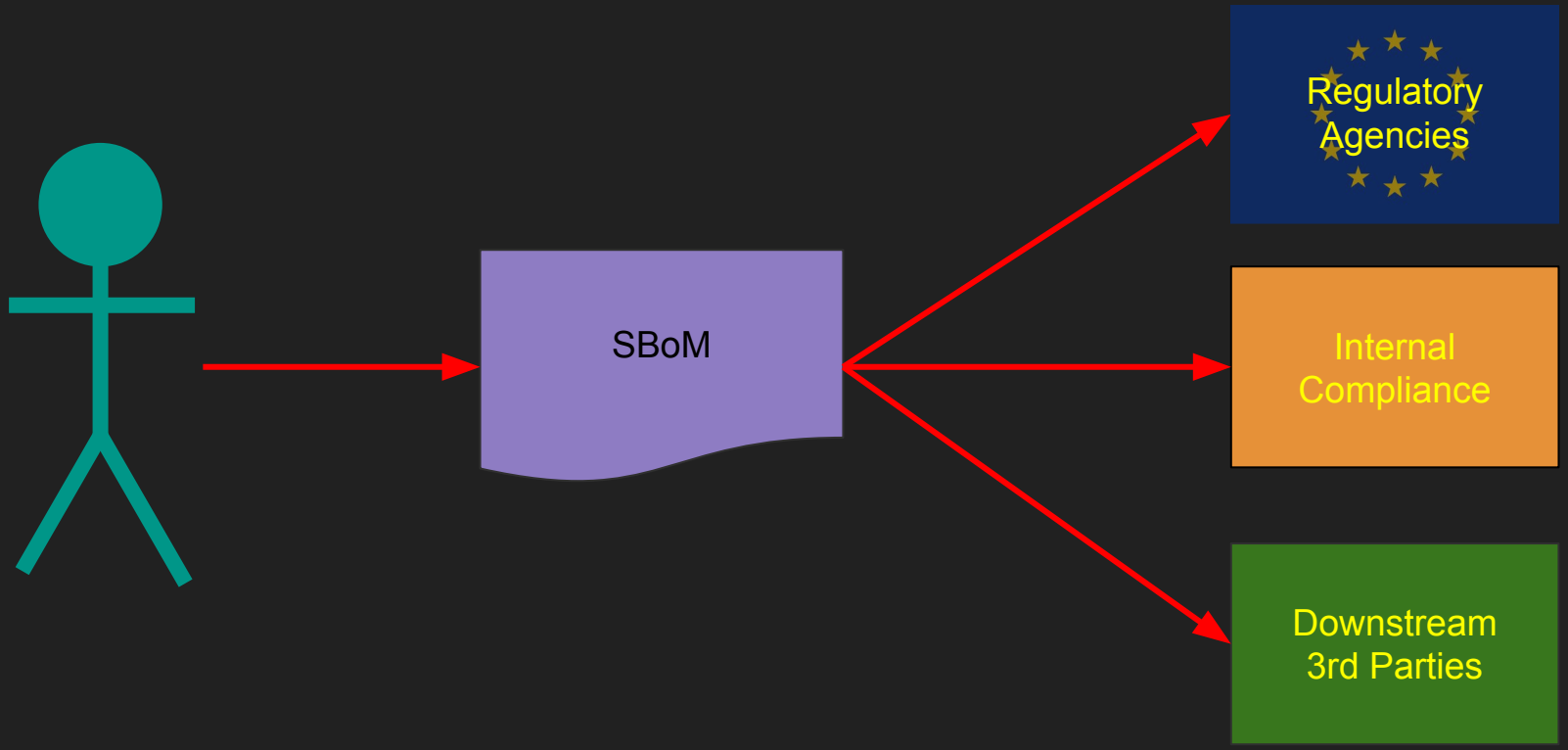
What Licenses are we using?

Internal

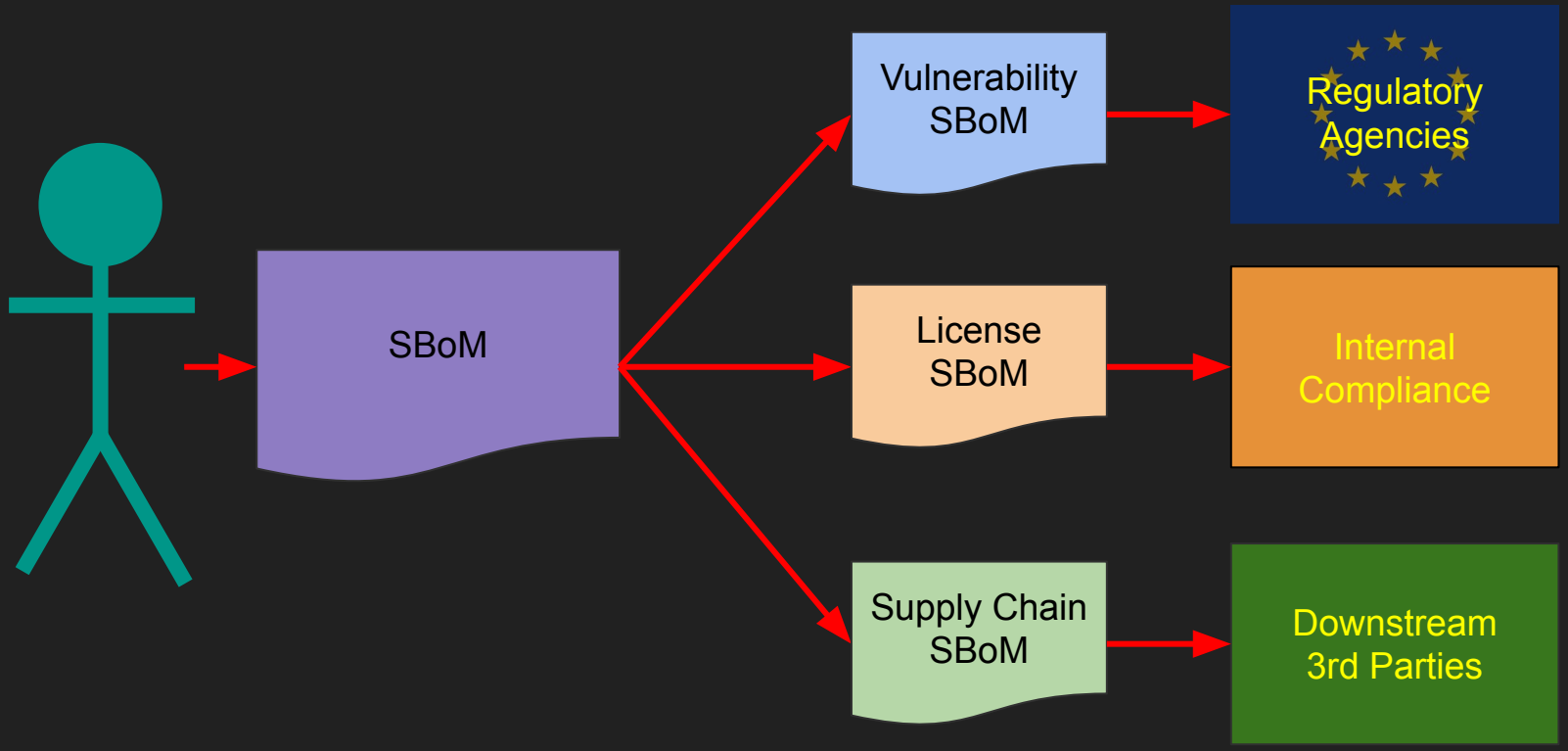
What's in the software you gave us?

Downstream 3rd Parties

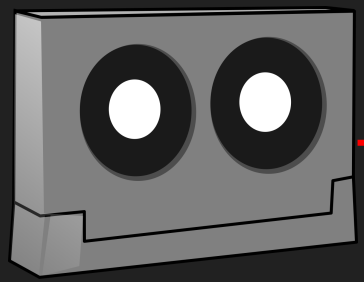
Standardized Format for the Exchange of Information



Standardized Format for the Exchange of Information



Standardized Format for the Exchange of Information



KSEltar, CC BY-SA 3.0 via Wikimedia Commons

SBoM

Vulnerability SBoM

License SBoM

Supply Chain SBoM

Regulatory Agencies

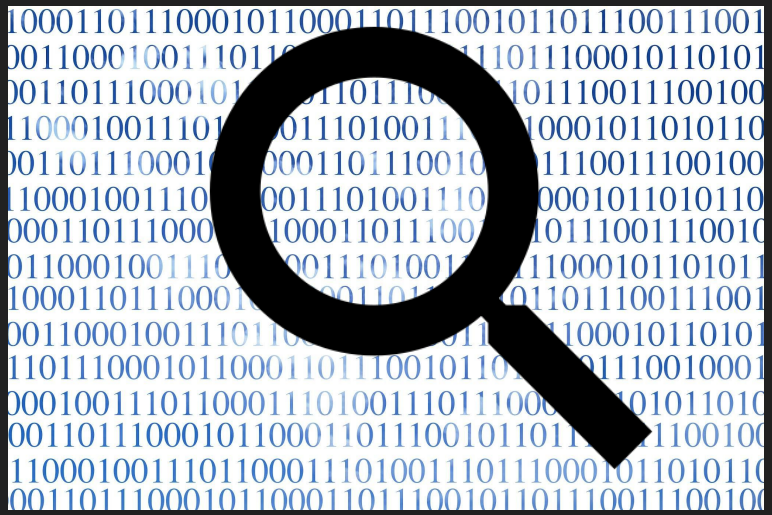
Internal Compliance

Downstream 3rd Parties

What's in my Software?

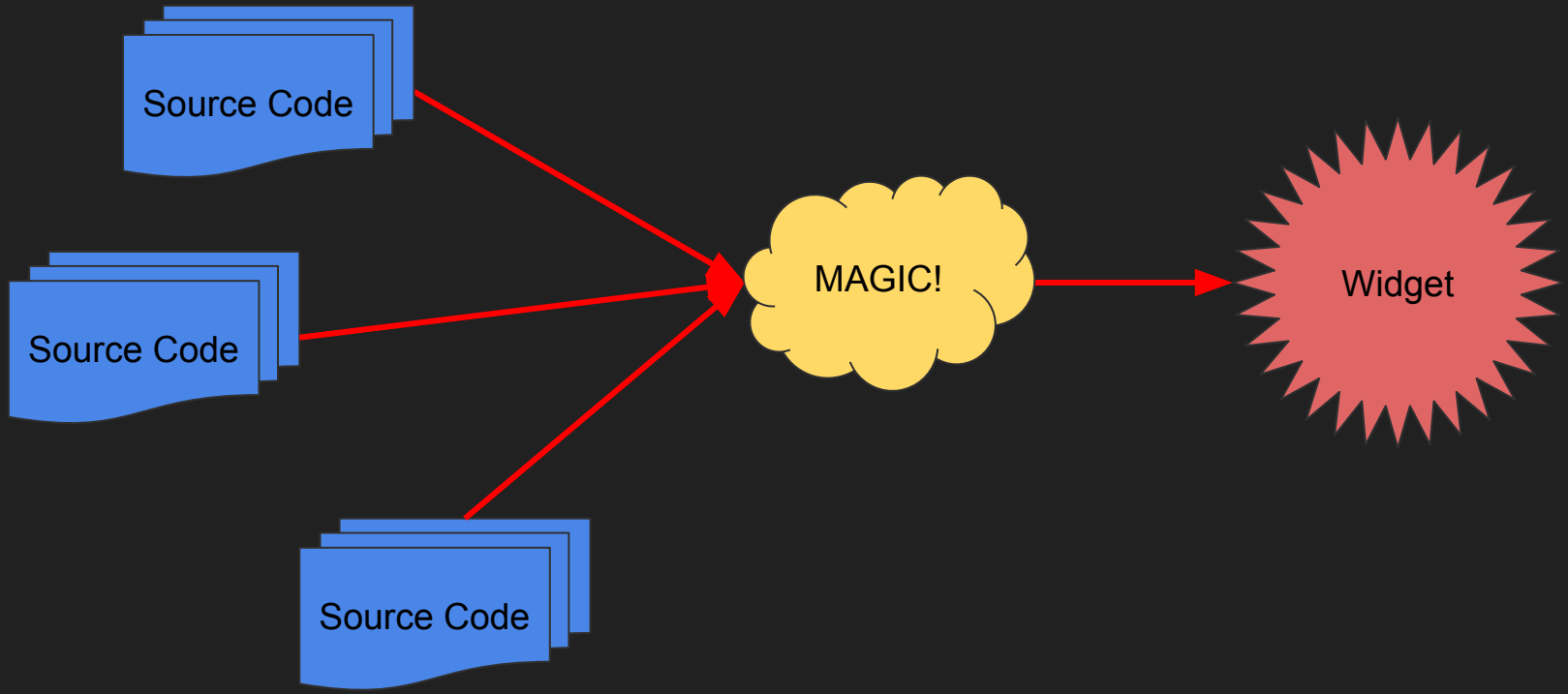
- Software is often not easily observable in its final form
- What's in my Software?
  - Where did it come from?
  - What version is it?
- Am I complying with Software Licenses?
- Has it been tampered with?
- Is it vulnerable to exploits?
- **Can deliverables be traced back to their code?**

What's really in here?

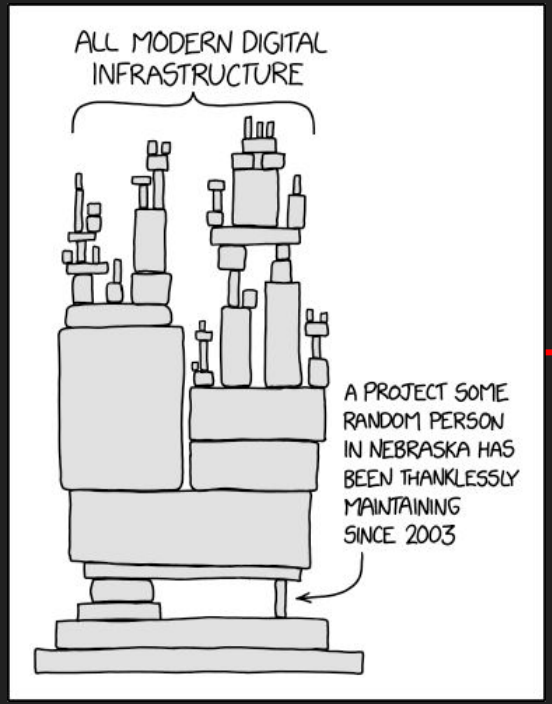


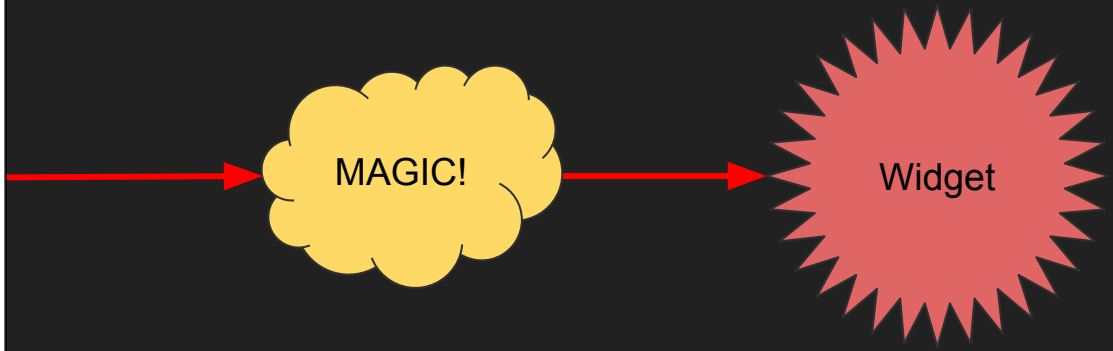
Sérgio Valle Duarte, CC BY 3.0, via Wikimedia Commons

Describes the Software Supply Chain



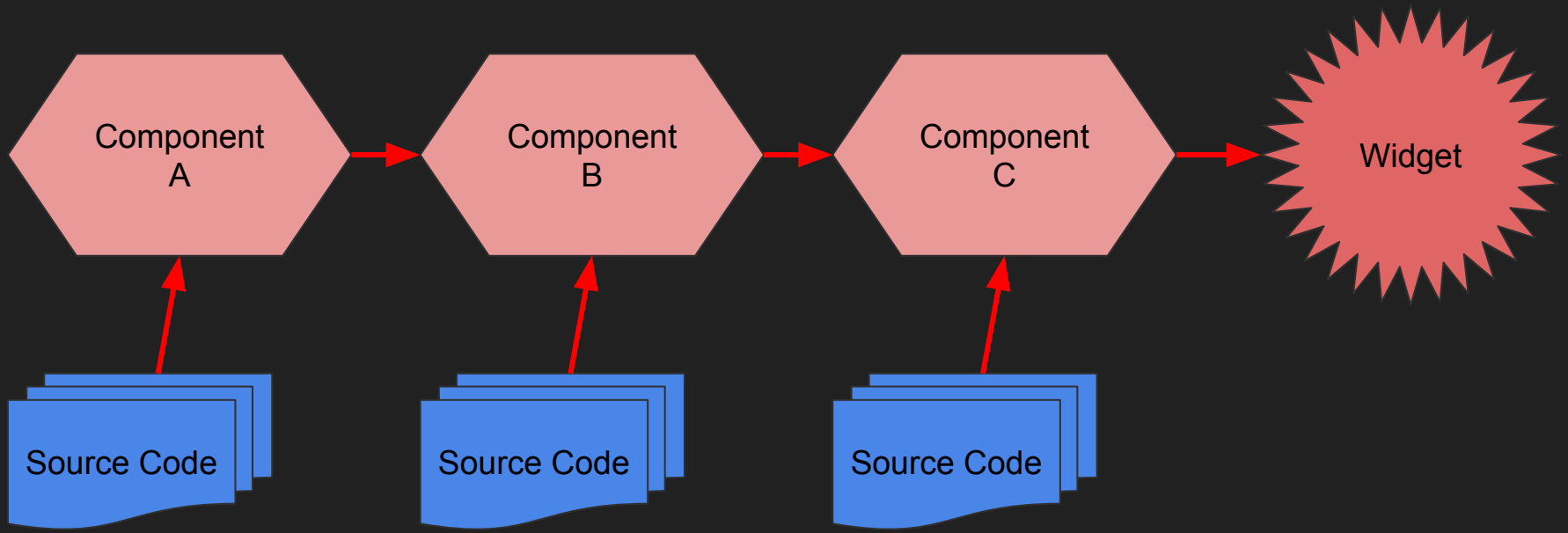
Describes the Software Supply Chain



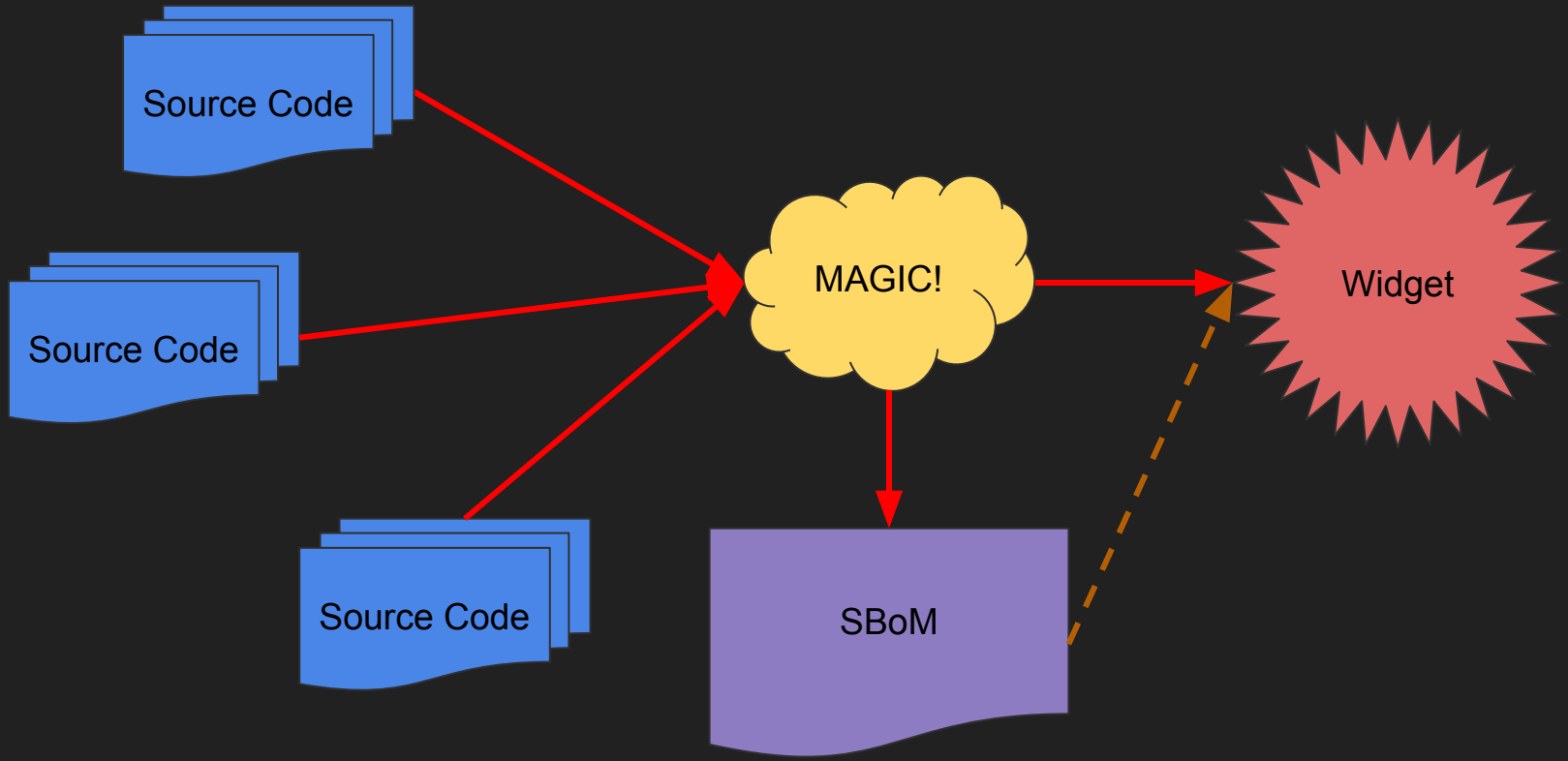


XKCD #2347 CC BY-NC 2.5

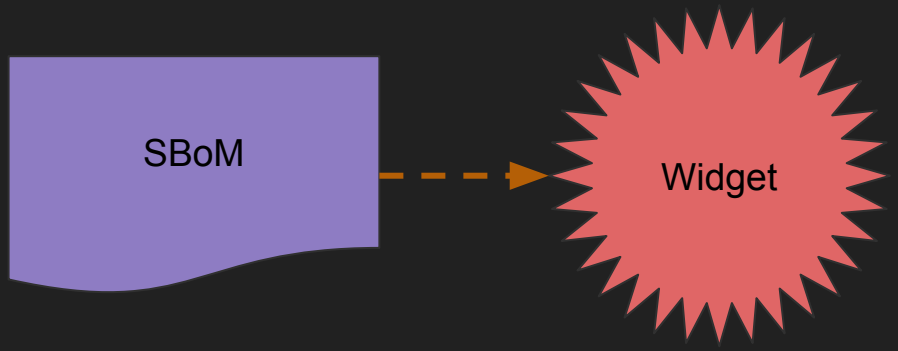
Describes the Software Supply Chain



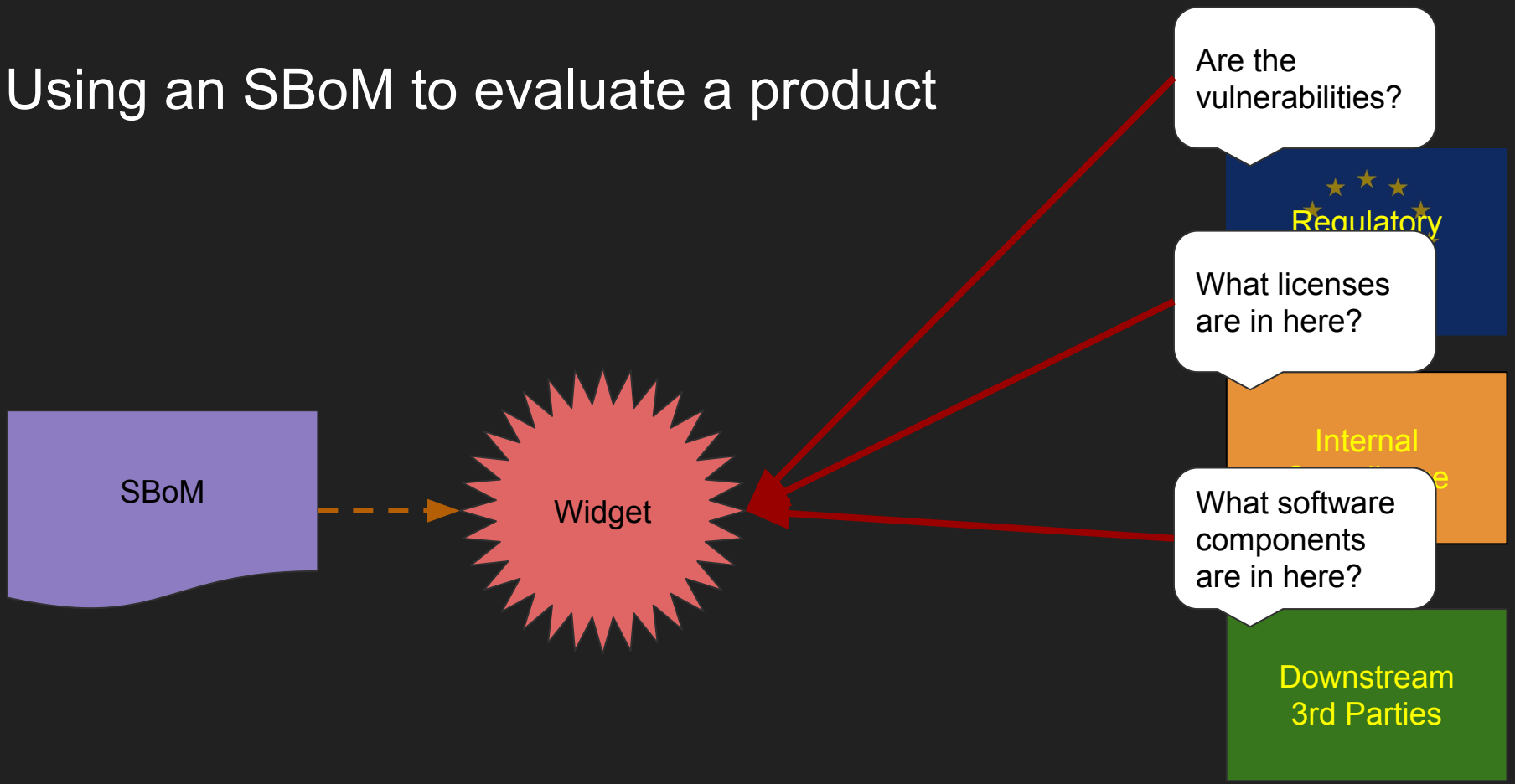
Describes the Software Supply Chain



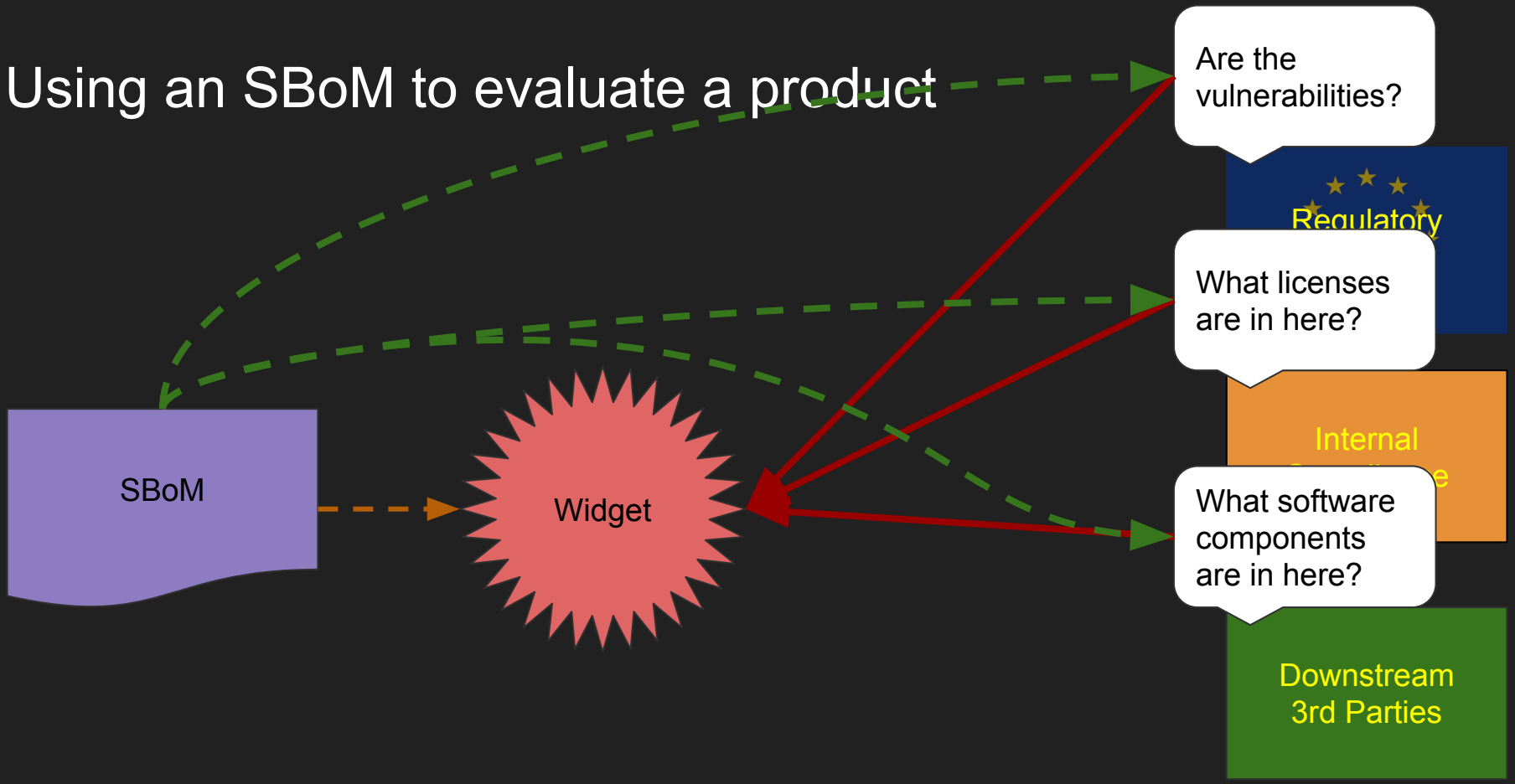
Using an SBoM to evaluate a product



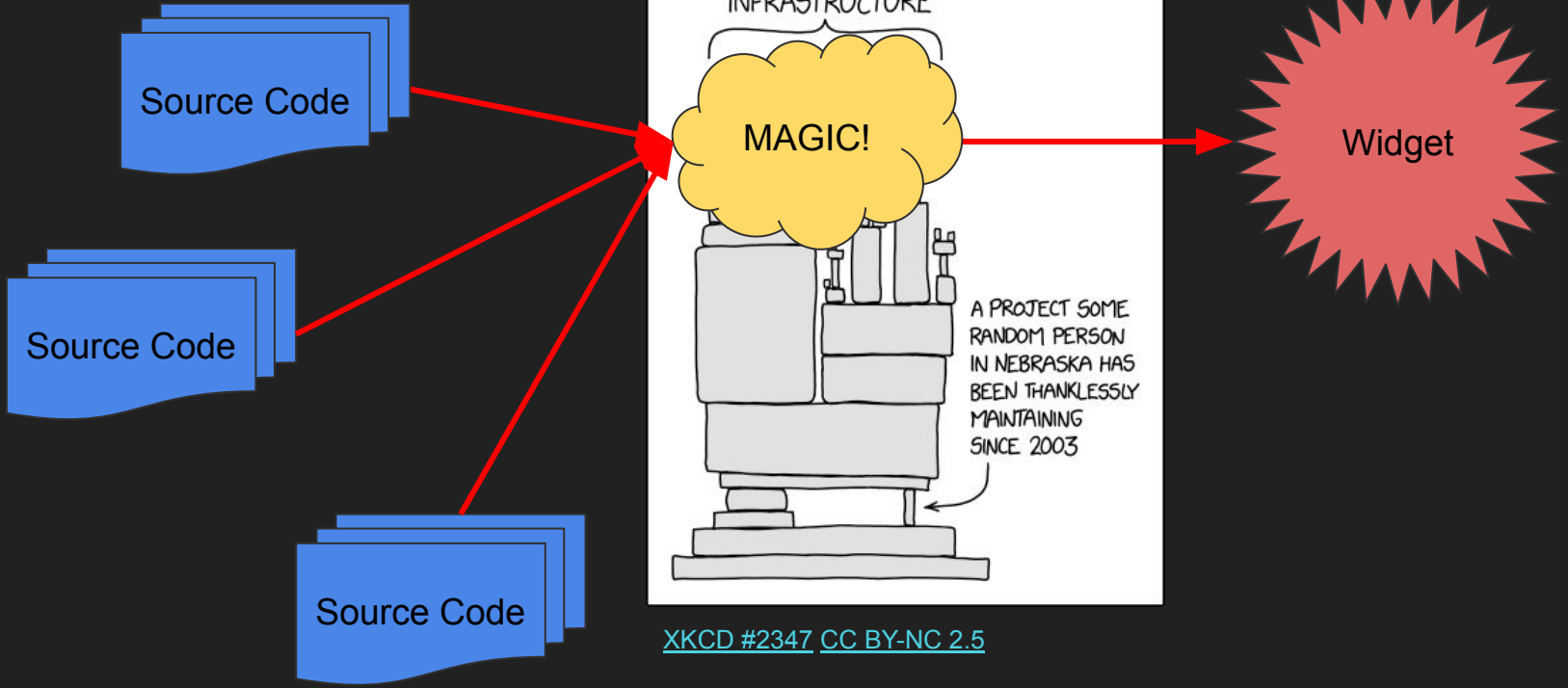
Using an SBoM to evaluate a product



Using an SBoM to evaluate a product

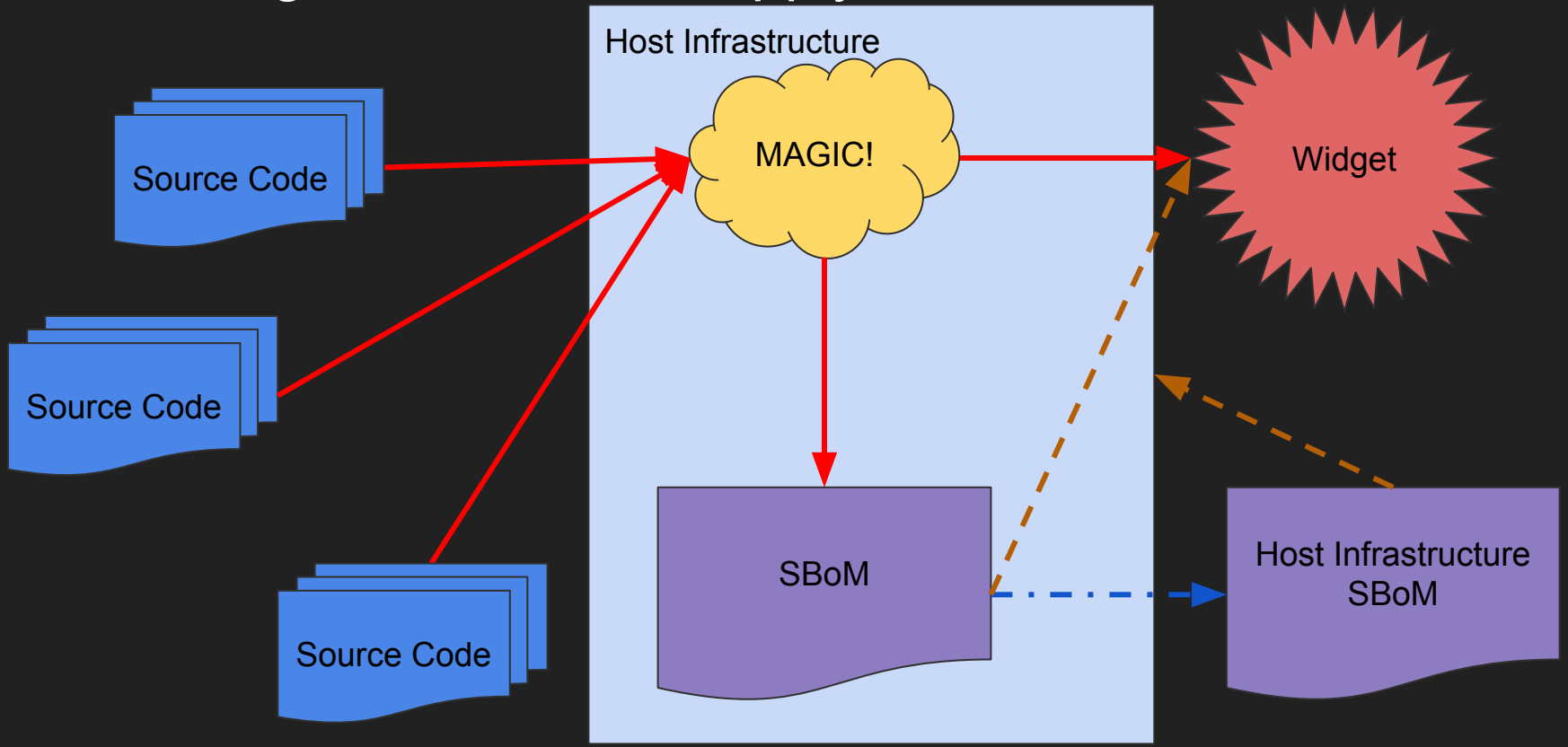


Extending the Software Supply Chain

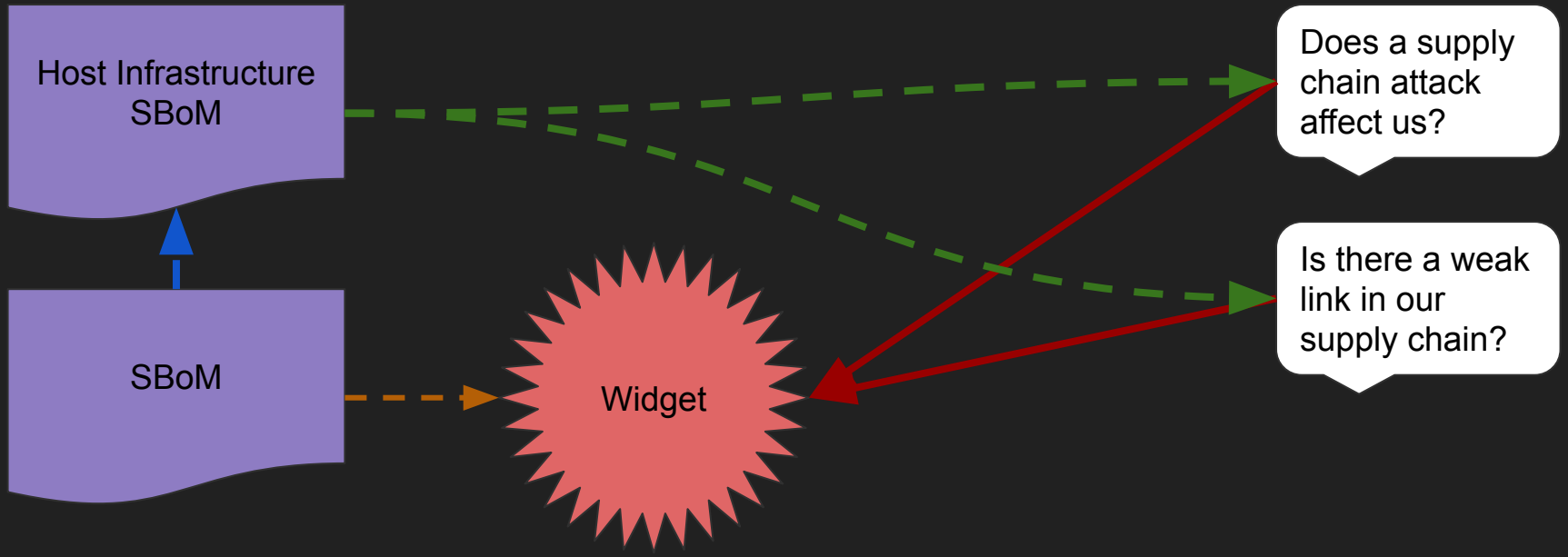


XKCD #2347 CC BY-NC 2.5

Extending the Software Supply Chain

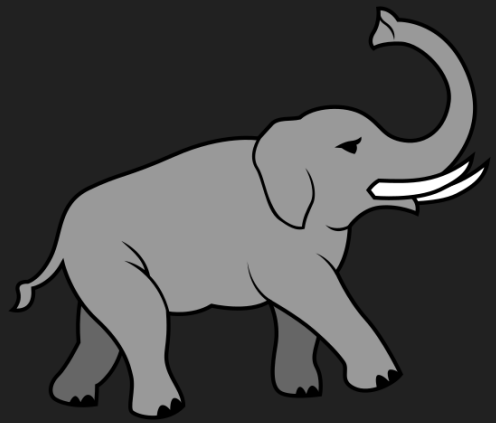


Using an SBoM to evaluate a supply chain



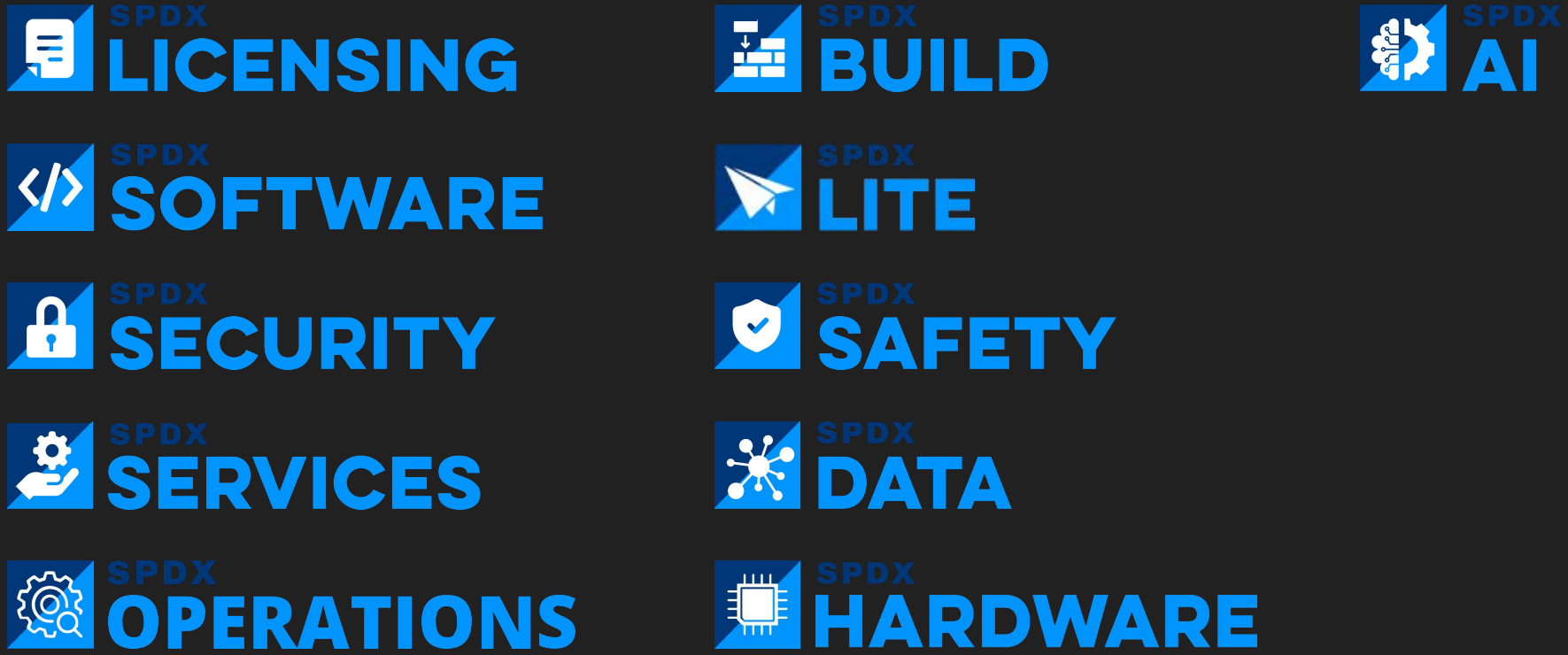
Regulatory Compliance

- SBoMs will likely become increasingly required by regulatory bodies
  - E.g. the CRA
- Building SBoMs into your workflows now can save some headache later



Fenn-O-maniC, CC BY-SA 4.0, via Wikimedia Commons

… And More


SPDX
LICENSING
SPDX
BUILD
SPDX
AI
SPDX
SOFTWARE
SPDX
LITE
SPDX
SECURITY
SPDX
SAFETY
SPDX
SERVICES
SPDX
DATA
SPDX
OPERATIONS
SPDX
HARDWARE

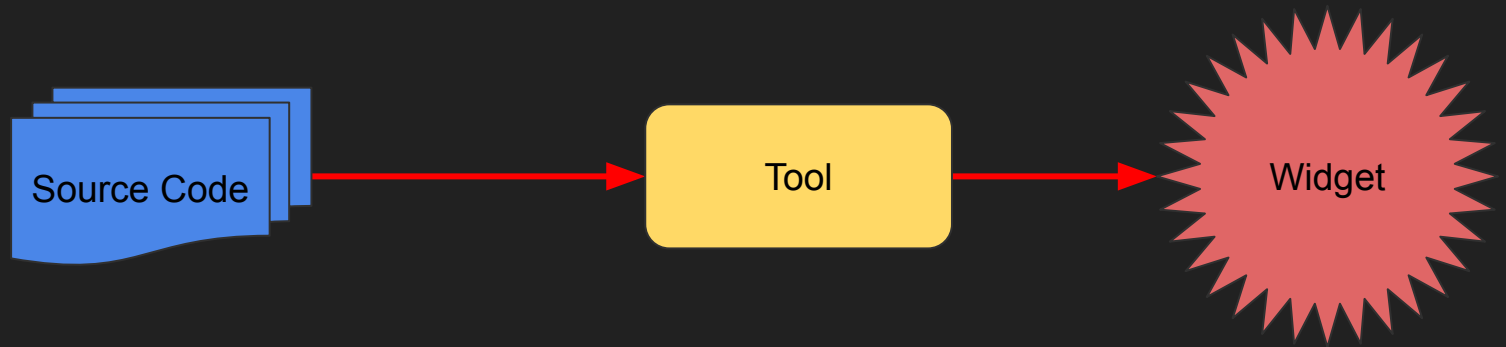
Strengths of Embedded Development for Supply Chains

Defined Scope

- Embedded focused project usually have a fairly defined scope
- Even if they are allowing arbitrary 3rd party software, it has to be well specified (e.g. recipe, package, etc.)
  - Because Cross Compiling is *hard*
- This specification is pretty close to an SBoM
  - Sometimes there's even support to dump this out, just in a bespoke format

"Forward" Engineered Supply Chain

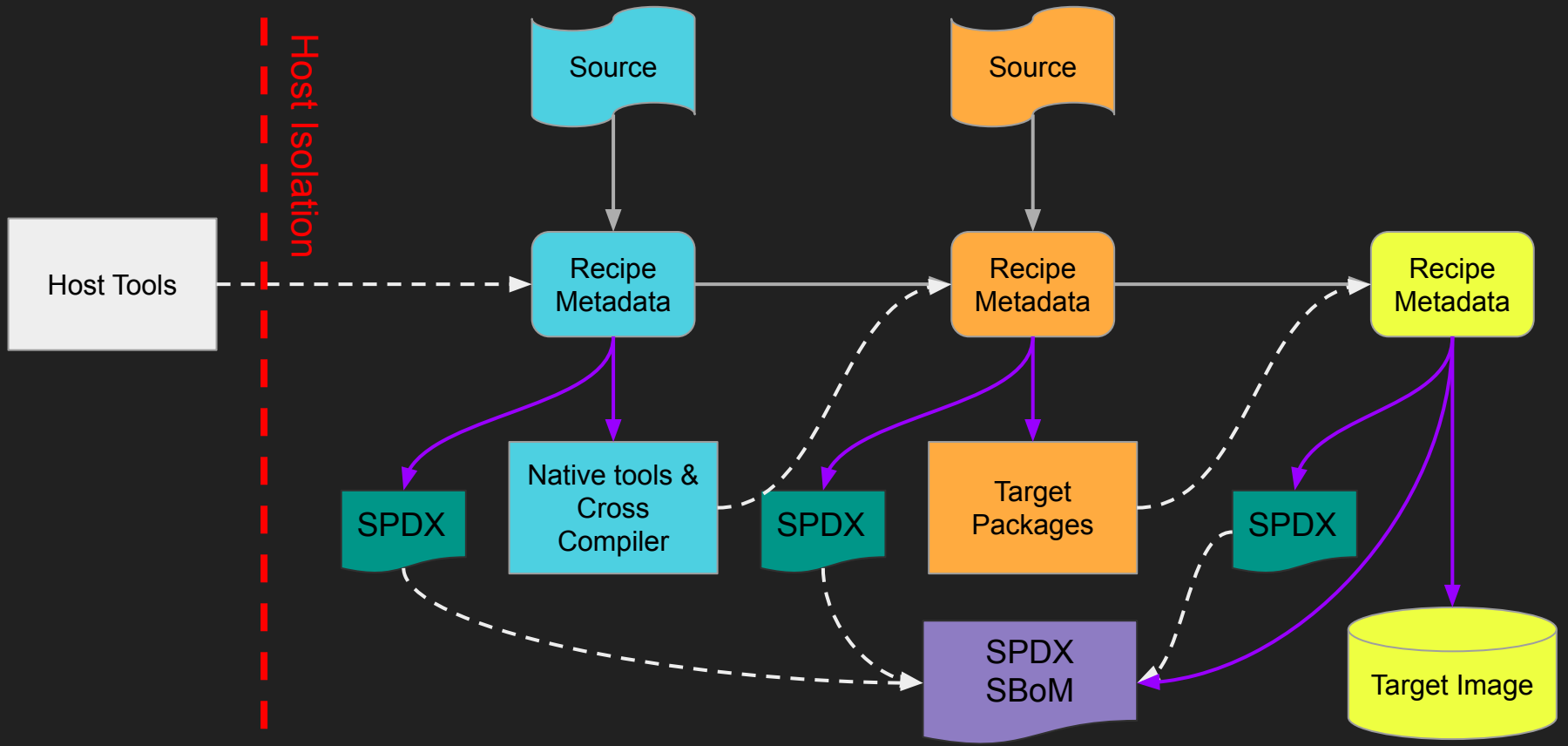
- Usually built from source
  - Extensive (and holistic) knowledge about what is built
  - Accurate and comprehensive dependencies
  - Usually some form of license tracking, etc.
- Supply chain is intrinsically "Forward" engineered
  - As opposed to "Reverse" engineered from resulting deliverables
  - Embedded Build systems typically state precisely *What* to build and *How* to build it to create the final deliverables



Host Isolation

- Constraints of Embedded Cross Compiling generally tend toward some form of host isolation
  - Many different forms:
    - Containers
    - Pre-build toolchains
    - Building required tools from source
    - Some combination of these
- This is also helpful from an SBoM perspective as it minimizes host "leakage" into the supply chain
- *However:* Expressing this in an SBoM can be tricky

SPDX Generation in Yocto (example)



What do we want in our SBoMs?

Software Bill of Materials

- SBoM output files in a standardized format
- Ideally, conformance with some standard for minimum elements (e.g. NITA minimum elements, TR-03183 (CRA))
- Build time dependencies
- Licensing information
- Vulnerability Data (CVEs, CPEs, etc.)

Reproducible Builds

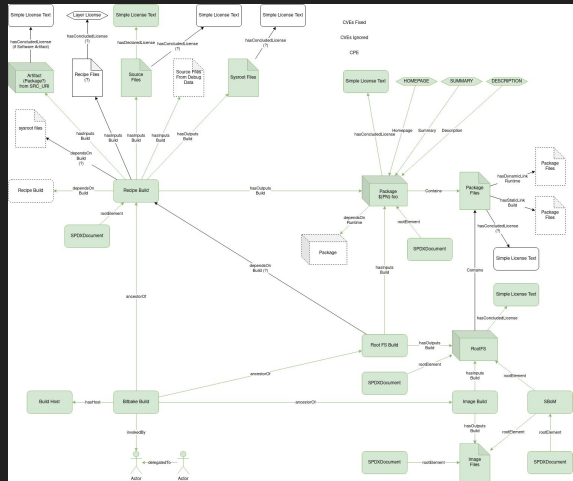
Reproducible Builds

- The ability to produce byte-for-byte output on subsequent builds
- Generally, this doesn't guarantee *your* build is reproducible
  - You need to test *your* build on your own
- Bonus points for being listed on: https://reproducible-builds.org/who/projects/
- Provides strength to supply chain

Project Support

Yocto

- SPDX 2.2 support since 3.4 Release (Honister)
- SPDX 3.0 support since 5.1 Release (Styhead)
  - Please use SPDX 3 if possible!
- Yocto project tests and tracks reproducibility for core recipes:
  - https://www.yoctoproject.org/reproducible-build-results/



Buildroot

- Reproducible builds enabled with BR2_REPRODUCIBLE=y
  - https://elinux.org/Buildroot:ReproducibleBuilds
- CycloneDX SBoM output
  - https://github.com/CycloneDX/cyclonedx-buildroot
  - WIP to integrate into main project

Zephyr SBoM

- SPDX 2.3 SBoM output (JSON format)
- Generated using `west`:
  - `west spdx -d BUILD_DIR`

https://docs.zephyrproject.org/latest/develop/west/zephyr-cmds.html#software-bill-of-materials-west-spdx

Apache NuttX RTOS

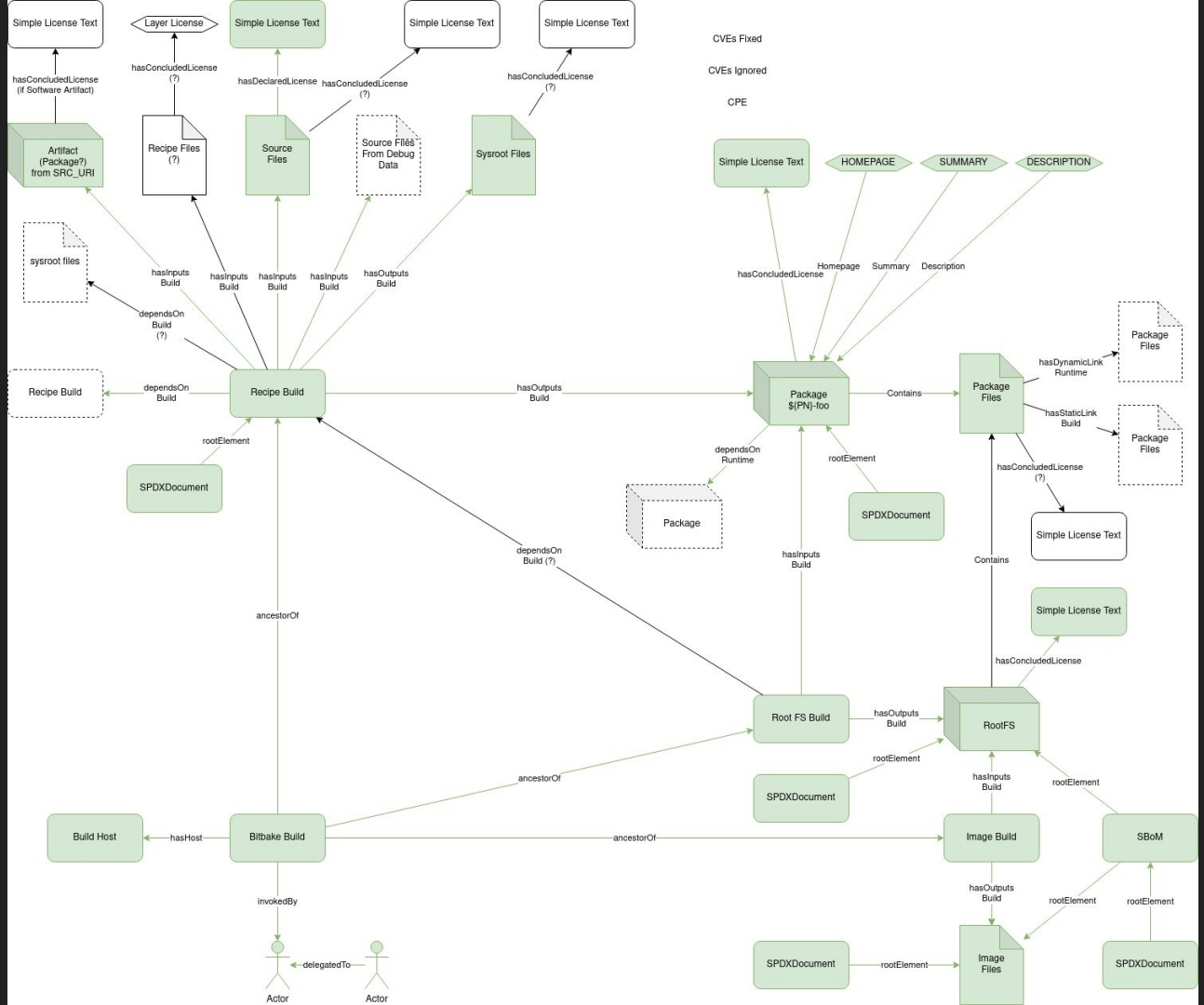
- SPDX 2.3 output (JSON format)

https://fosdem.org/2025/schedule/event/fosdem-2025-4528-sbom-journey-for-an-open-source-project-apache-nuttx-rtos/

Didn't See Your Project?

- Maybe I just didn't find it (Sorry)
  - I'm not trying to shame anyone!
- Ask around in the community
  - This is pretty important
  - If you already have the data, it's generally not too hard to get it into an accepted format
  - It's good PR for your project :)
- If you have questions about implementation, I'm happy to provide general guidance

Discussion/Questions


Simple License Text
Layer License
Simple License Text
Simple License Text
Simple License Text
CVEs Fixed
CVEs Ignored
CPE
hasConcludedLicense (if Software Artifact)
hasConcludedLicense (?)
hasDeclaredLicense
hasConcludedLicense (?)
hasConcludedLicense (?)
Artifact (Package?) from SRC_URI
Recipe Files (?)
Source Files
Source Files From Debug Data
Sysroot Files
Simple License Text
HOMEPAGE
SUMMARY
DESCRIPTION
sysroot files
hasInputs Build
hasInputs Build
hasInputs Build
hasInputs Build
hasOutputs Build
hasConcludedLicense
Homepage
Summary
Description
dependsOn Build (?)
Package Files
hasDynamicLink Runtime
Recipe Build
dependsOn Build
Recipe Build
hasOutputs Build
Package ${PN}-foo
Contains
Package Files
hasStaticLink Build
Package Files
rootElement
dependsOn Runtime
rootElement
hasConcludedLicense (?)
SPDXDocument
Package
SPDXDocument
Simple License Text
dependsOn Build (?)
hasInputs Build
Contains
Simple License Text
ancestorOf
hasConcludedLicense
Root FS Build
hasOutputs Build
RootFS
rootElement
ancestorOf
SPDXDocument
hasInputs Build
rootElement
Build Host
hasHost
Bitbake Build
ancestorOf
Image Build
SBoM
hasOutputs Build
rootElement
rootElement
invokedBy
SPDXDocument
rootElement
Image Files
SPDXDocument
delegatedTo
Actor
Actor