Automated SBoM generation

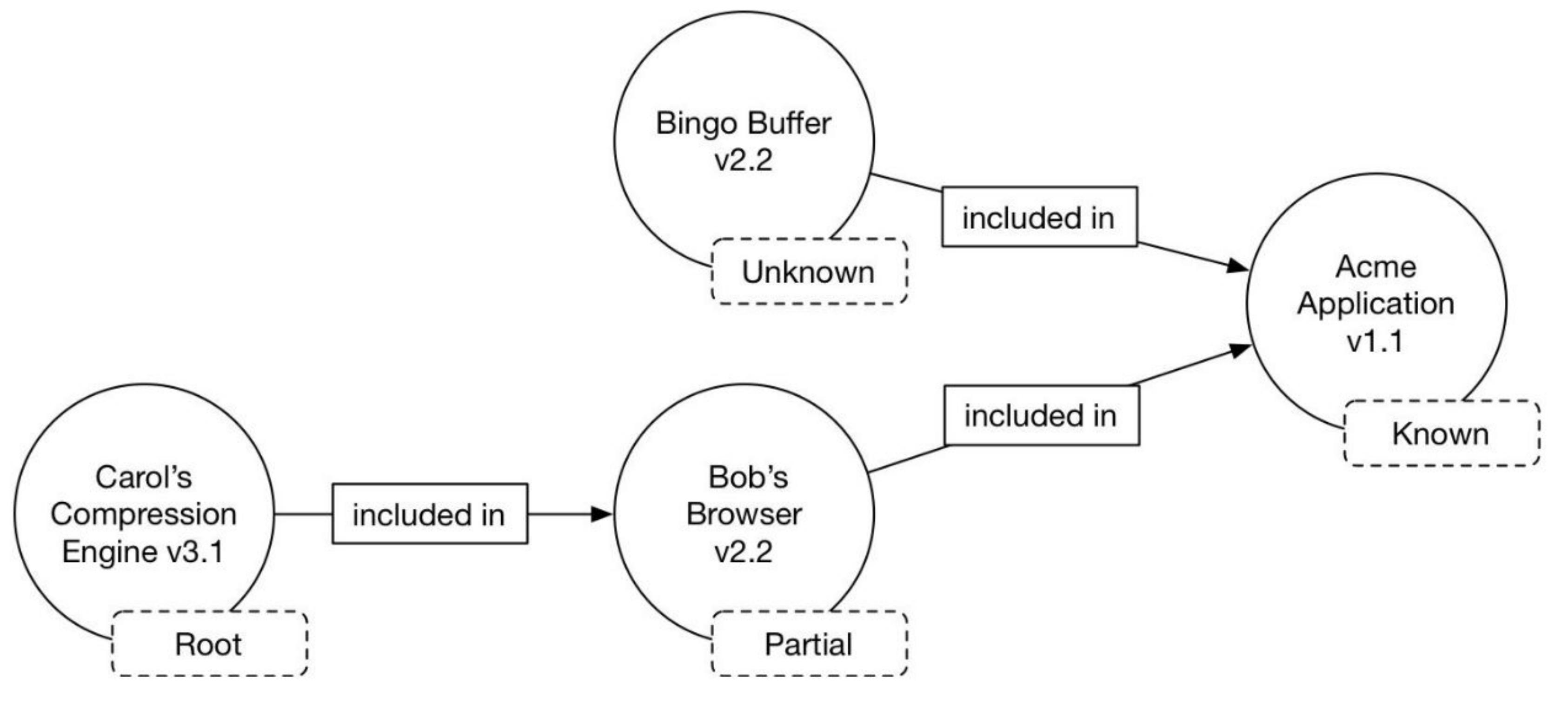
A case study of SBoM generation in meta build systems

Joshua Watt
FOSDEM 2023
February 5th, 2023

About Me

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

What is an SBoM?

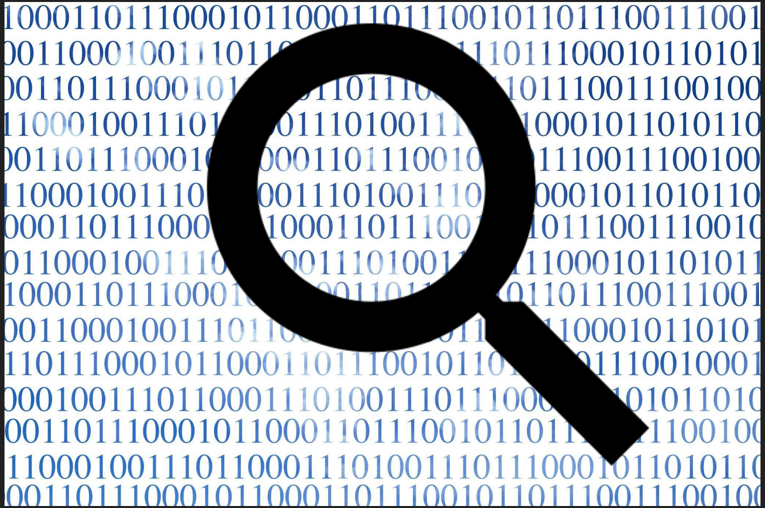


Source: NTIA's [Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM)]

Why are SBoMs important?

- 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

"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**

An SBoM is a method of describing the information about a Software Supply Chain using a standardized encoding that allows for easy exchange of data

Multiple different SBoM formats may describe the same Software Supply Chain

"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

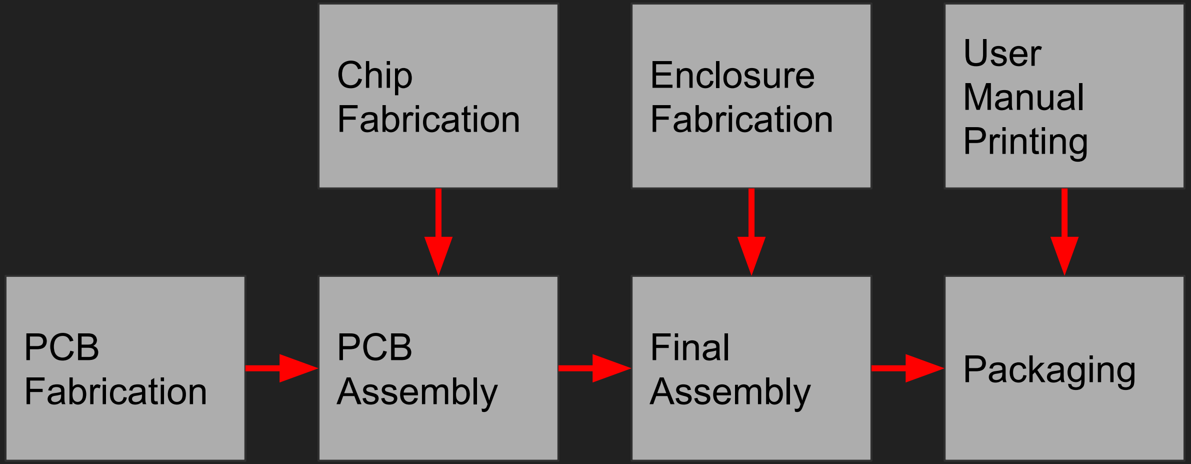
Patches Applied **30**

Good Analogy, but is missing a few key points:

- Where did the software come from?
- How did it get in here?

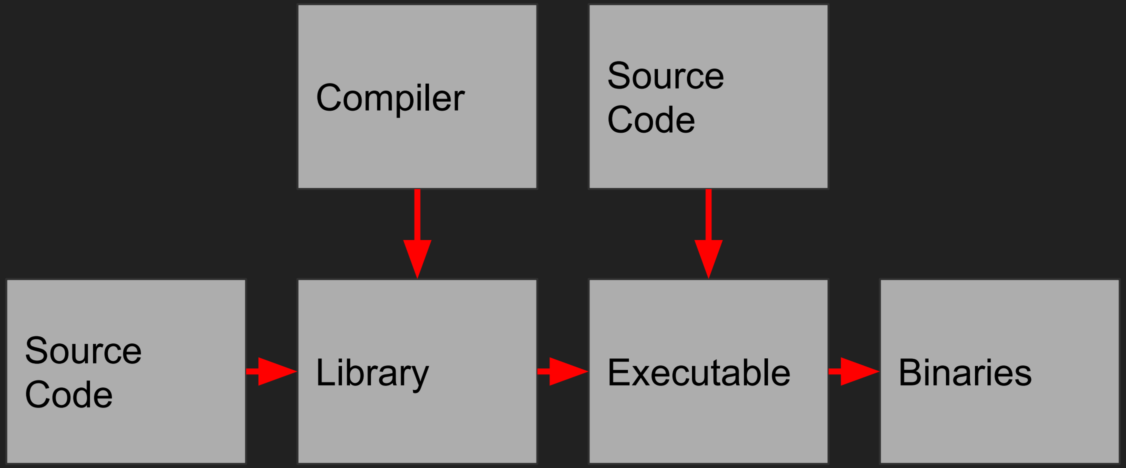
The "Supply Chain" part

Physical Supply Chains



- Where do components come from?
- What is being combined at each step?
- Where does combination take place and Who did it?
- When did the combination occur?

Software Supply Chains



- Where do components come from?
- What is being combined at each step?
- Where does combination take place and Who did it?
- When did combination occur?

SPDX Build Profile

Releasing with SPDX 3.0 within a few months

- *When* was a Build done?
- *Who* wanted the build done?
  - A person
- *Who* actually performed the build?
  - A person, or a service like "GitHub Actions"
- *How* was the build done?
  - tool-specific information about how the build was performed, like command line arguments, etc.
  - Build time and Run time dependencies already captured by core SPDX profile
- *Where* was the build done?
  - Build host (maybe another SBoM)
  - Tools used (e.g. compiler, etc.)
- *What* is covered by the core SPDX profile

Build SBoMs need to be generated at build time

SBoM Types

- **Source SBoM**
  - An SBoM that ships with source code, e.g. in the upstream repository
- **Build SBoM**
  - An SBoM generated when source code is built
- **Post Mortem SBoM**
  - An SBoM generated by a scanning tool after the code has been built

No one method of providing SBoMs can provide everything! Each has their strengths and weaknesses.

(When) Build Time

Source SBoM	Build SBoM	Post Mortem SBoM
No	Yes	Maybe

Videoplasty.com, CC BY-SA 4.0, via Wikimedia Commons

(How) Build Time Dependencies

Source SBoM

Yes (e.g. Cargo, NPM, etc.)

Yes but not concretely

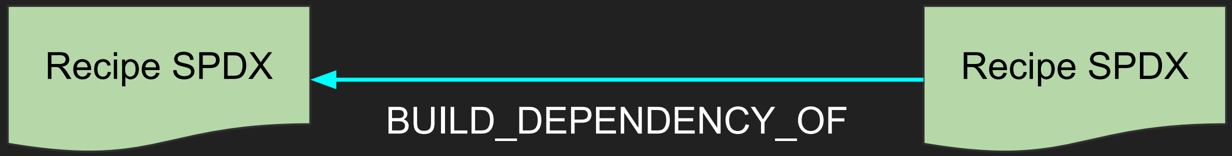
Build SBoM

Yes; build time dependencies have to be correct in order to successfully build

Post Mortem SBoM

Maybe; probably heuristically

Static libraries are problematic



(How) Runtime Time Dependencies

Source SBoM

Yes but not concretely

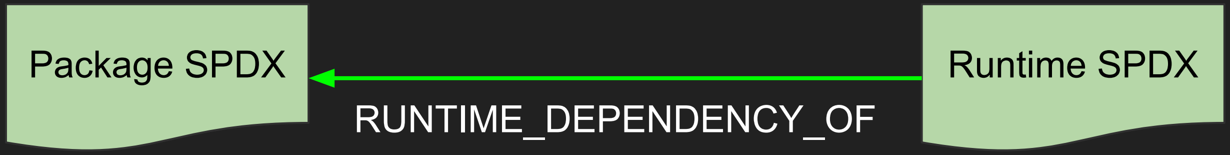
Build SBoM

Yes; runtime time dependencies have to be encoded in packages for successful installation and runtime behavior

Post Mortem SBoM

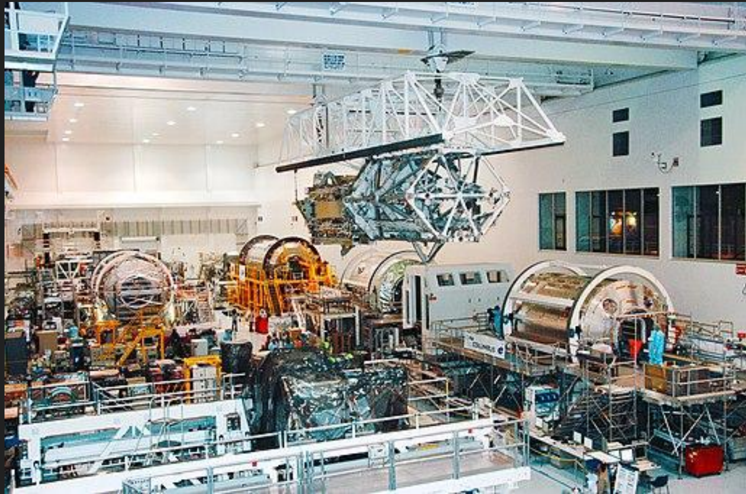
Shared libraries - yes

Runtime dynamically loaded libraries - Probably not



(Where) Build Environment

Source SBoM	Build SBoM	Post Mortem SBoM
No	Yes	Highly unlikely, probably heuristically



Advantages of generating Supply Chain from Build tools

- Authoritative
  - First hand knowledge; the tool doing the build is generating the SBoM
- Accurate
  - No guessing or heuristics are necessary for most information
- Comprehensive
  - Able to analyze most steps in assembly
  - Potentially able to report on things that may be difficult in any other context
    - E.g. static libraries, build-time & runtime dependencies for components

What can Generate Supply Chain SBoM information?

- Container Build systems
  - Docker build
  - Buildah
- Meta (distro) build systems
  - OpenEmbedded
  - Debian
  - Fedora
- Package Build systems
  - Autotools
  - cmake
  - Meson

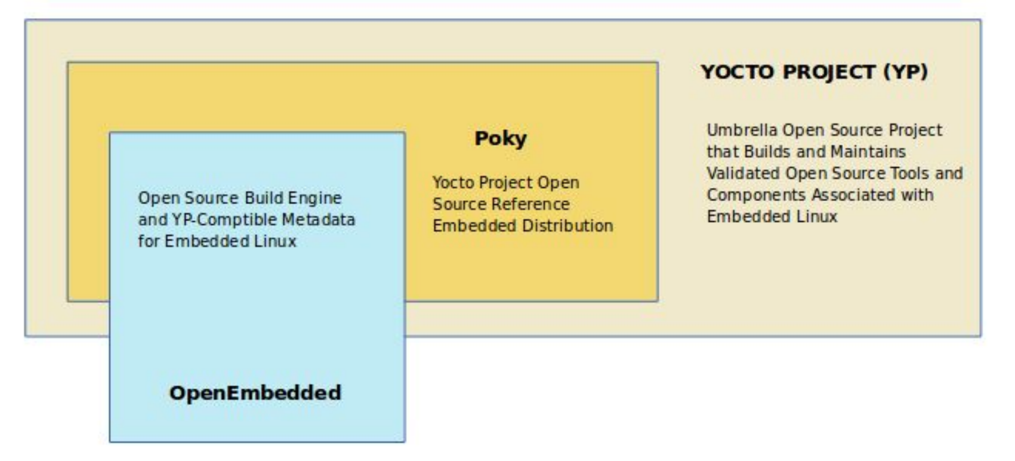
Hélène Rival, CC BY-SA 4.0, via Wikimedia Commons

OpenEmbedded Example

OpenEmbedded and Yocto Project

OpenEmbedded

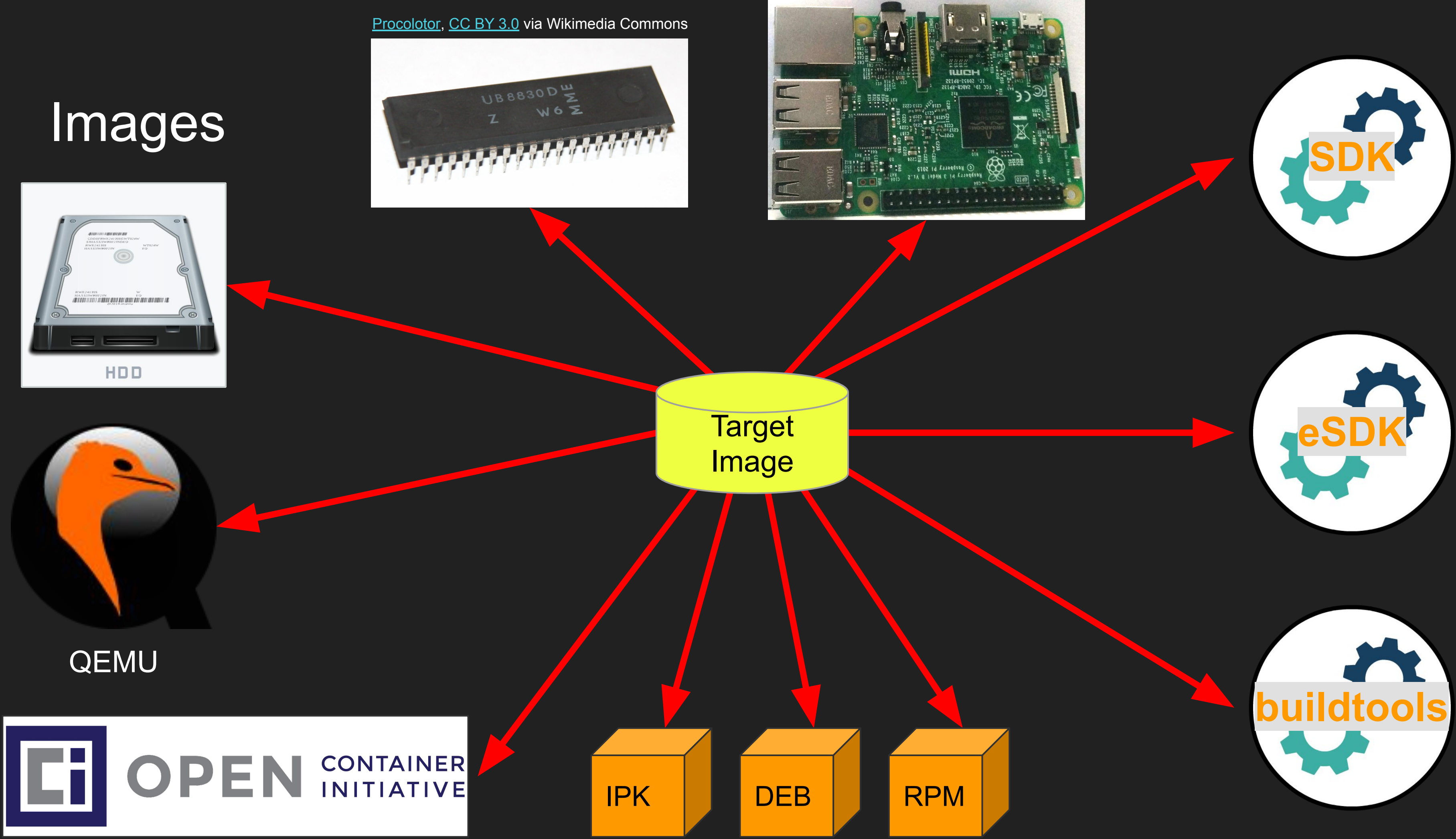
- Community project
- OpenEmbedded core layer
- Build system (bitbake)


YOCTO PROJECT (YP)
Umbrella Open Source Project that Builds and Maintains Validated Open Source Tools and Components Associated with Embedded Linux
Poky
Yocto Project Open Source Reference Embedded Distribution
Open Source Build Engine and YP-Comptible Metadata for Embedded Linux
OpenEmbedded

Yocto Project

- Linux Foundation project
- Poky reference distribution
- Runs QA tests
- Manages release schedule
- Provides funding for personnel
- Documentation

Images



Procolotor, CC BY 3.0 via Wikimedia Commons

Target Image

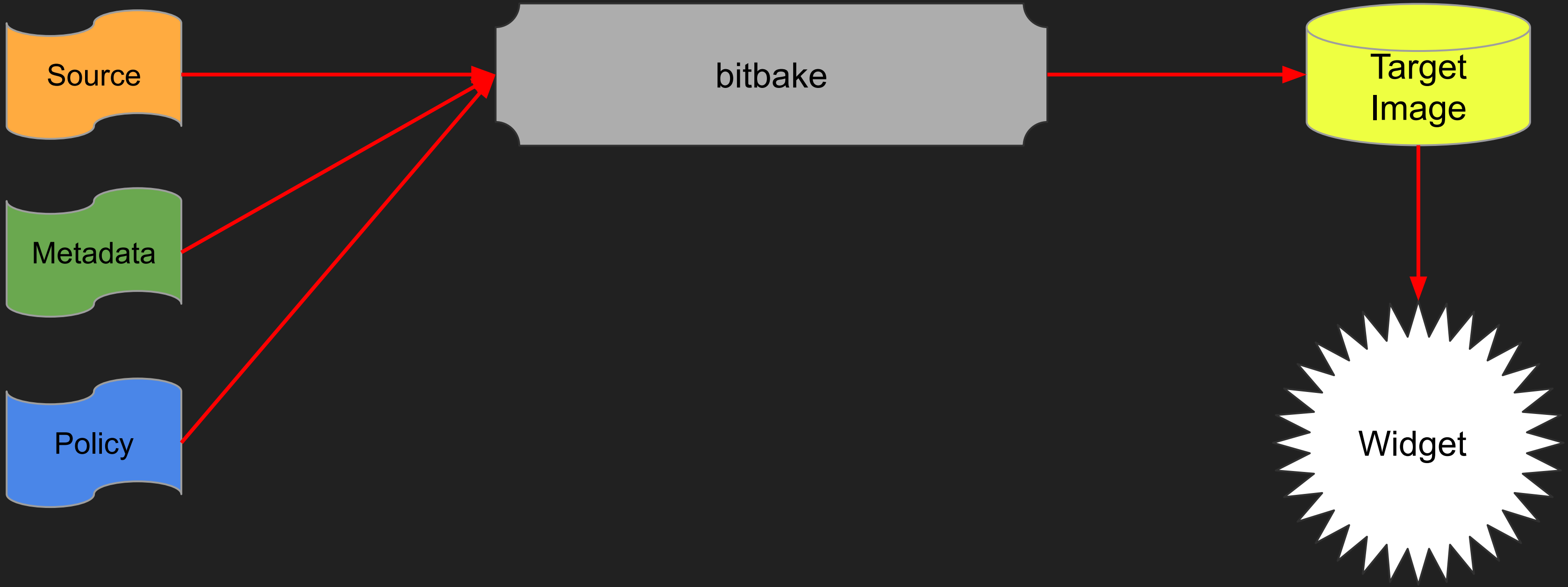
QEMU

IPK

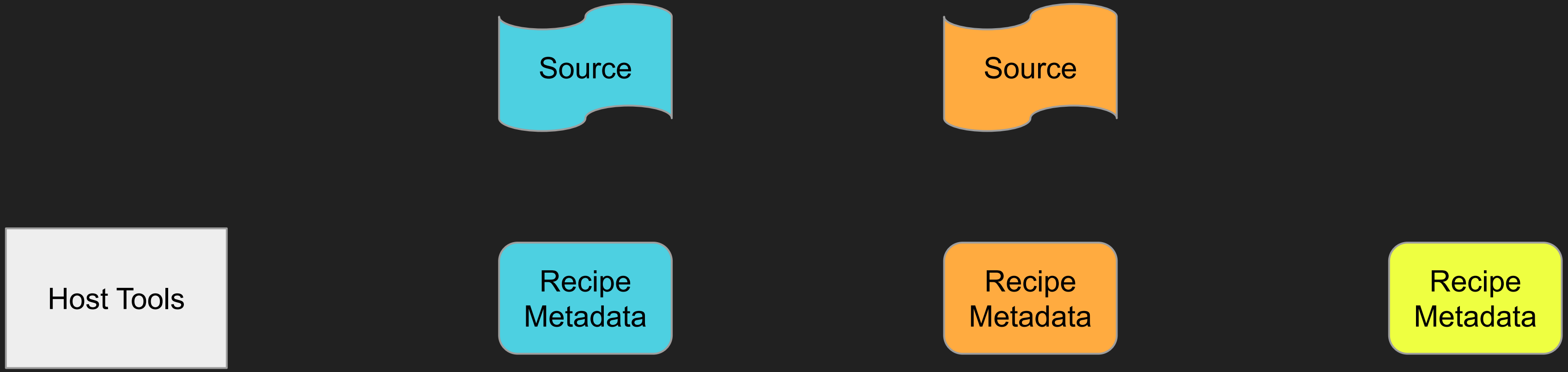
DEB

RPM

Build Images from Source Code



Simplified Build Flow



Source

Source

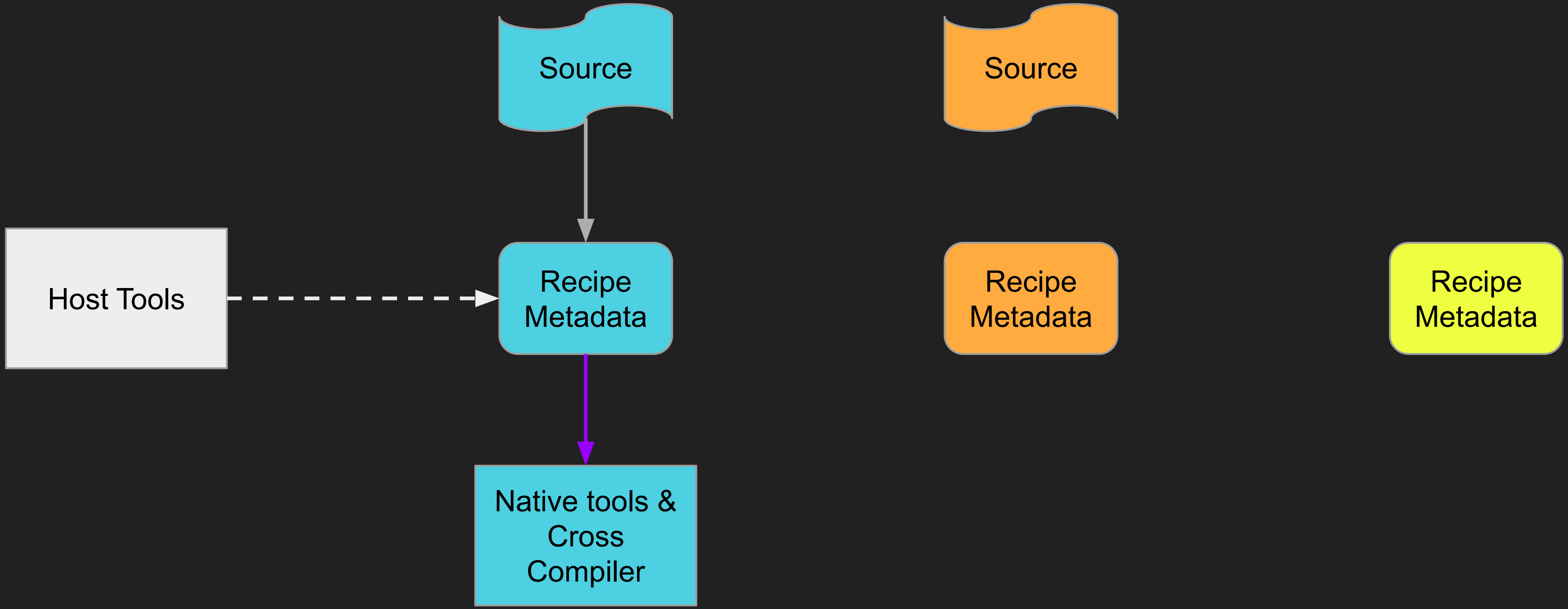
Host Tools

Recipe Metadata

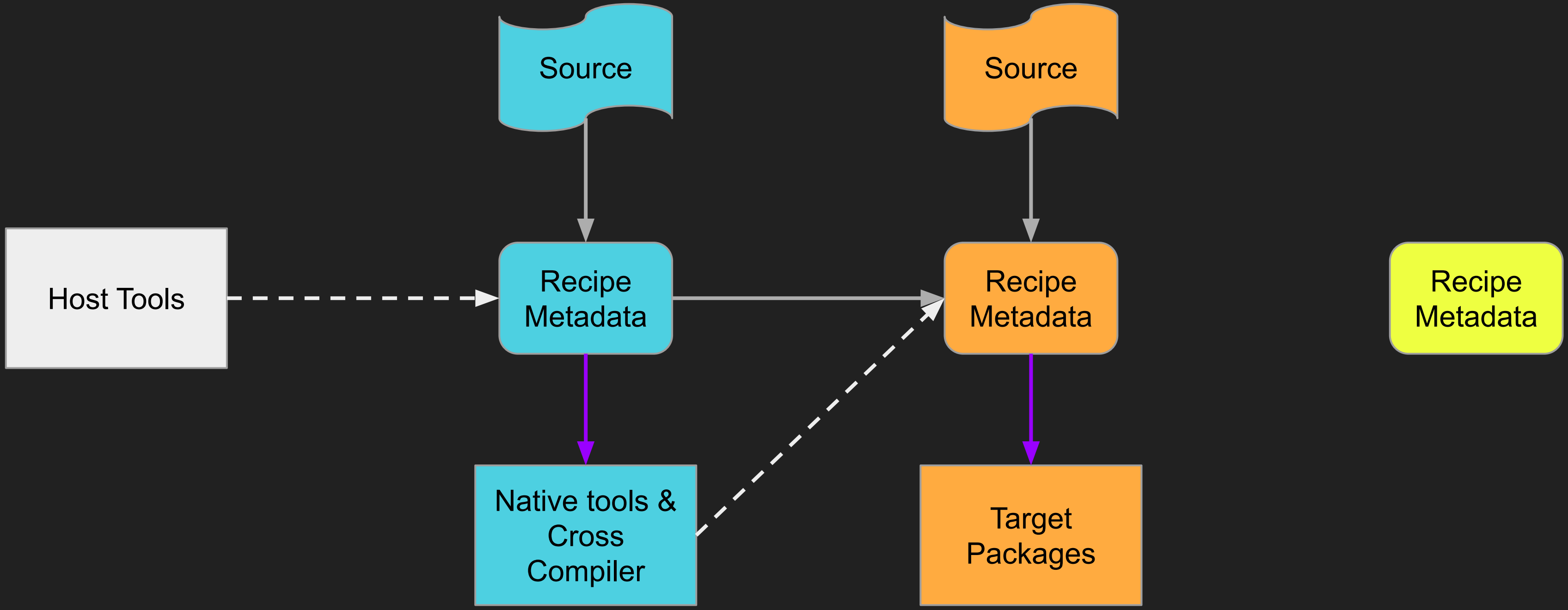
Recipe Metadata

Recipe Metadata

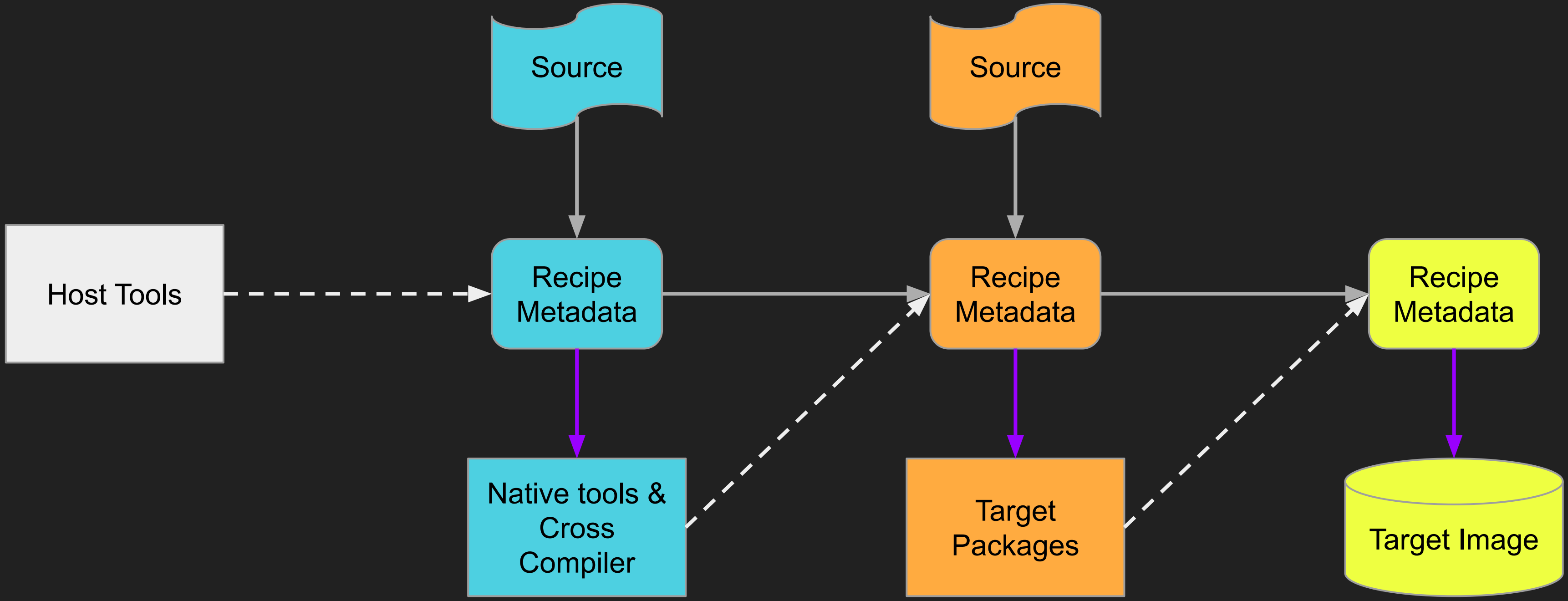
Simplified Build Flow



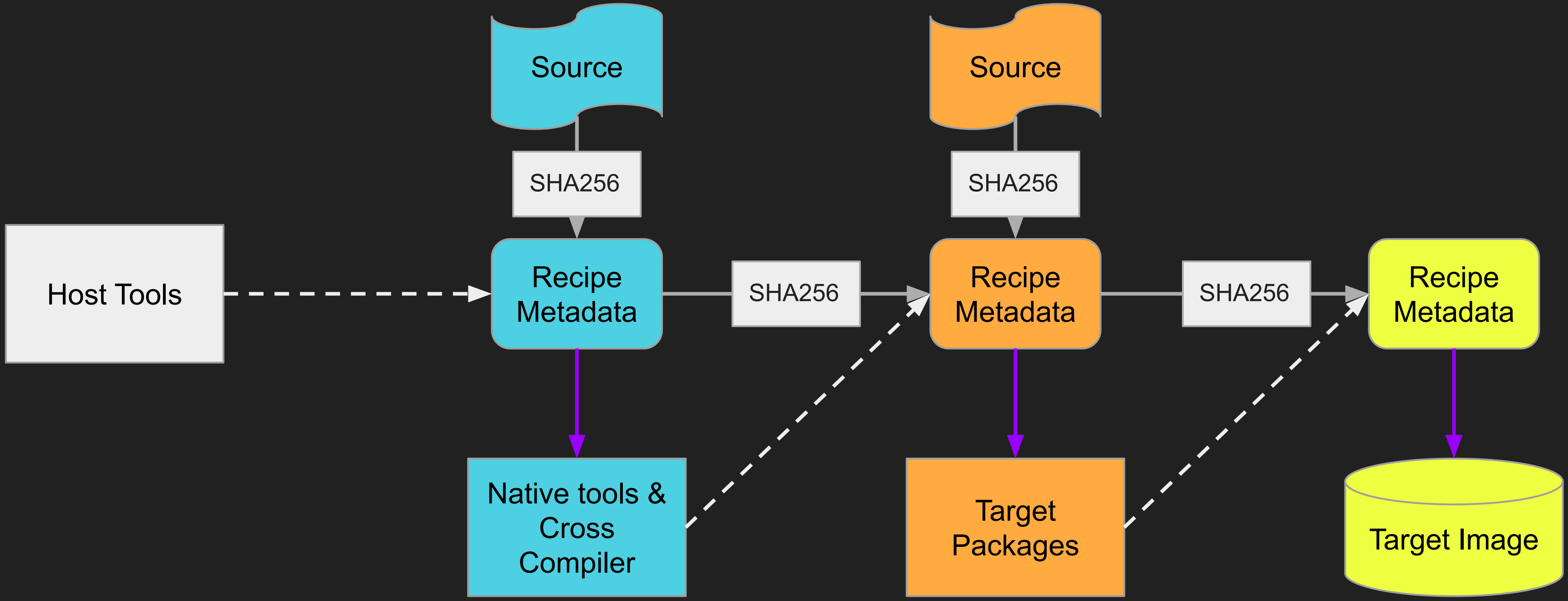
Simplified Build Flow



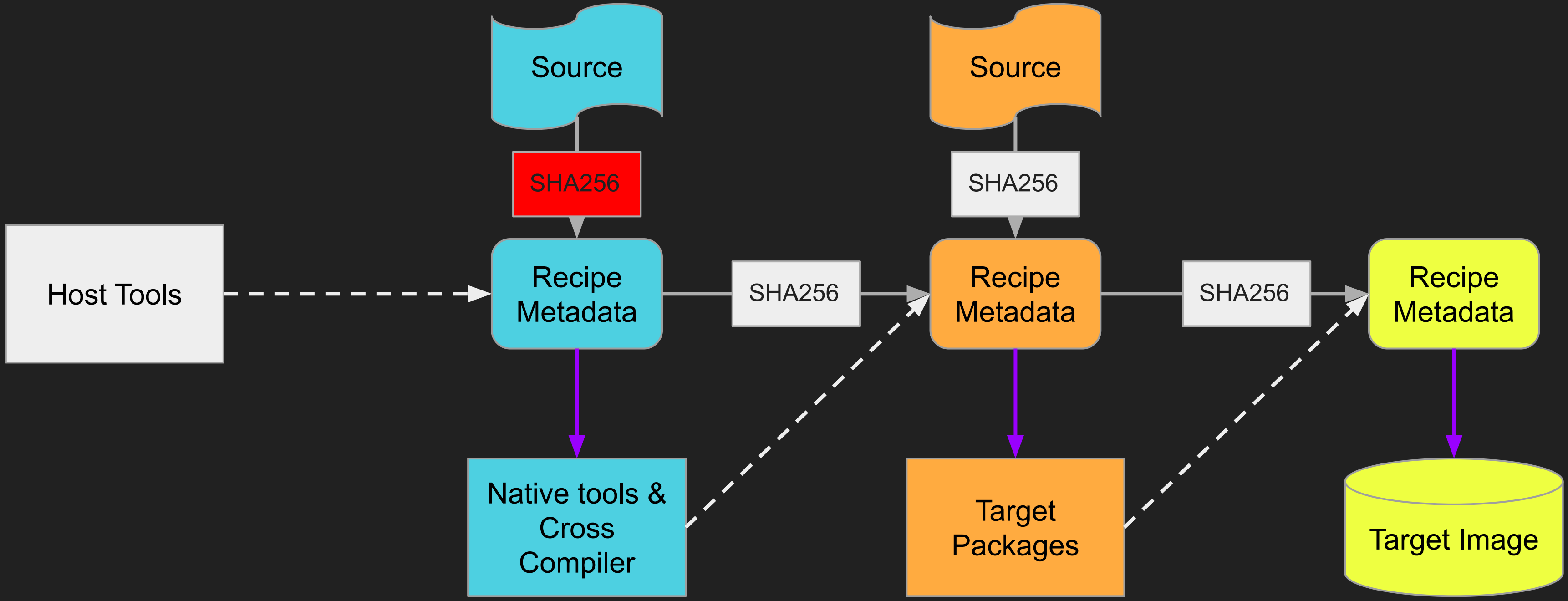
Simplified Build Flow



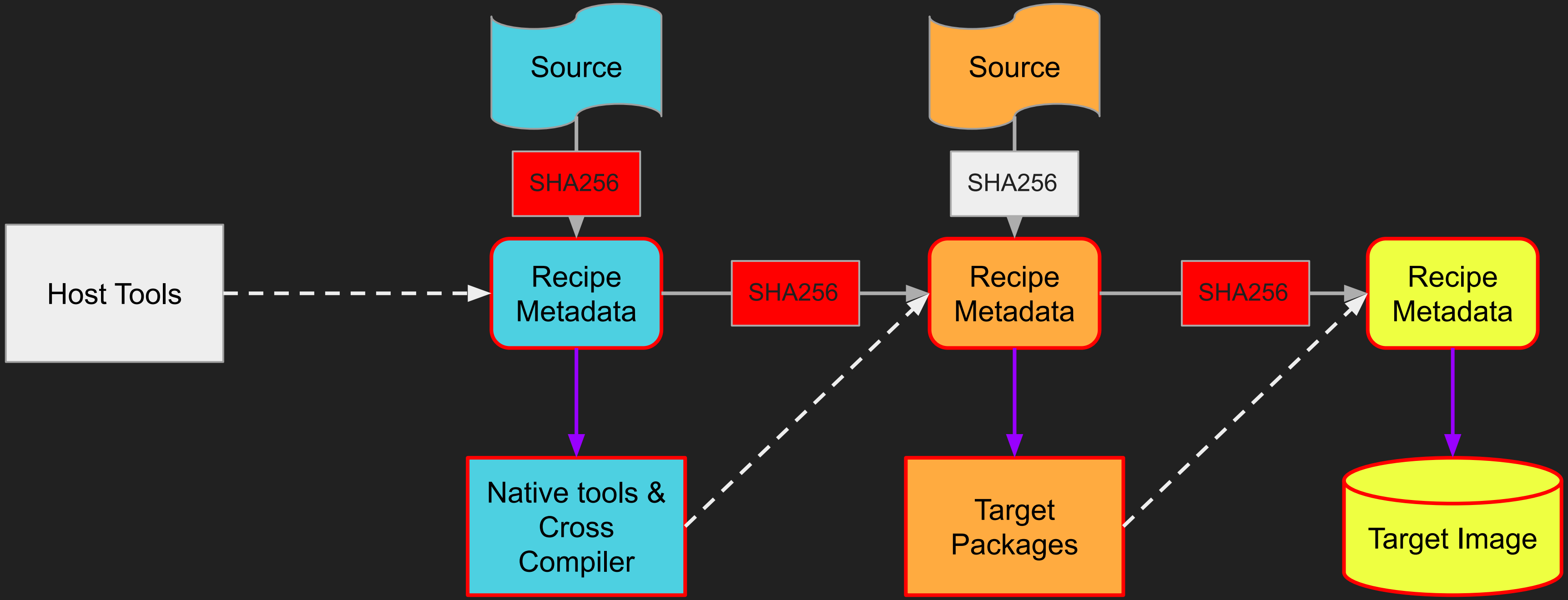
Simplified Build Flow



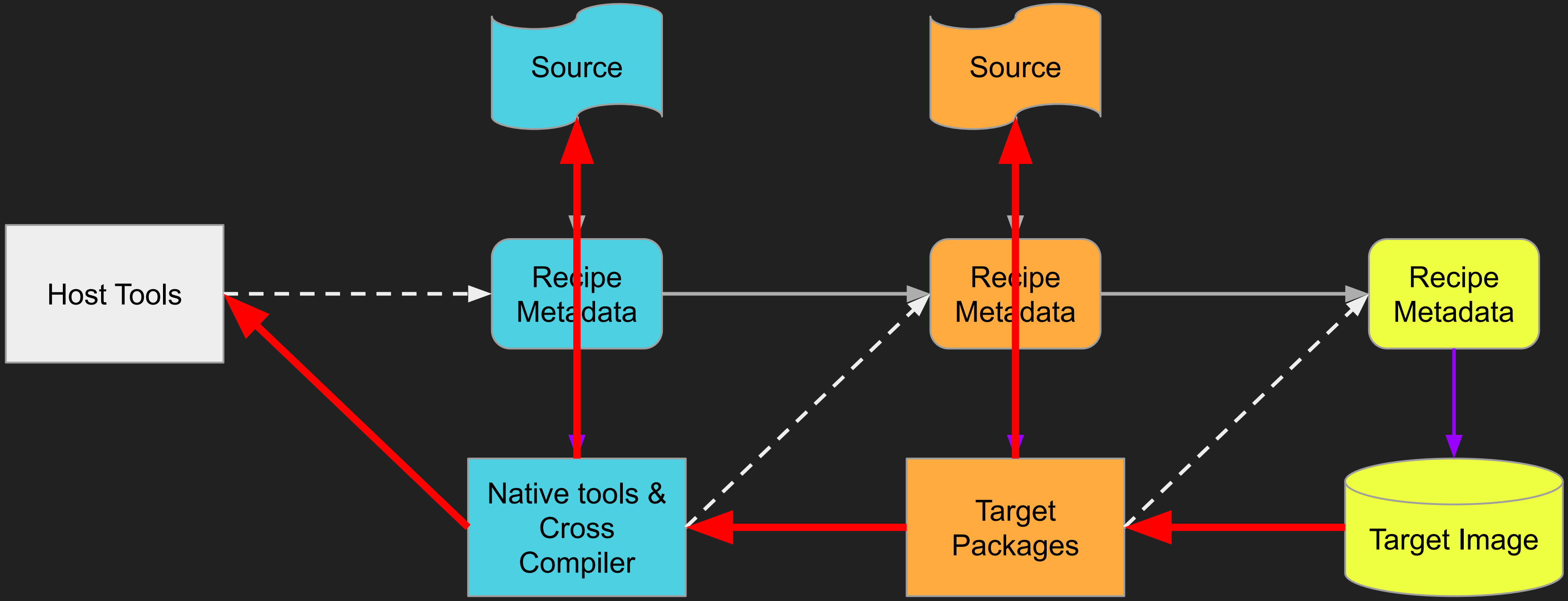
Simplified Build Flow



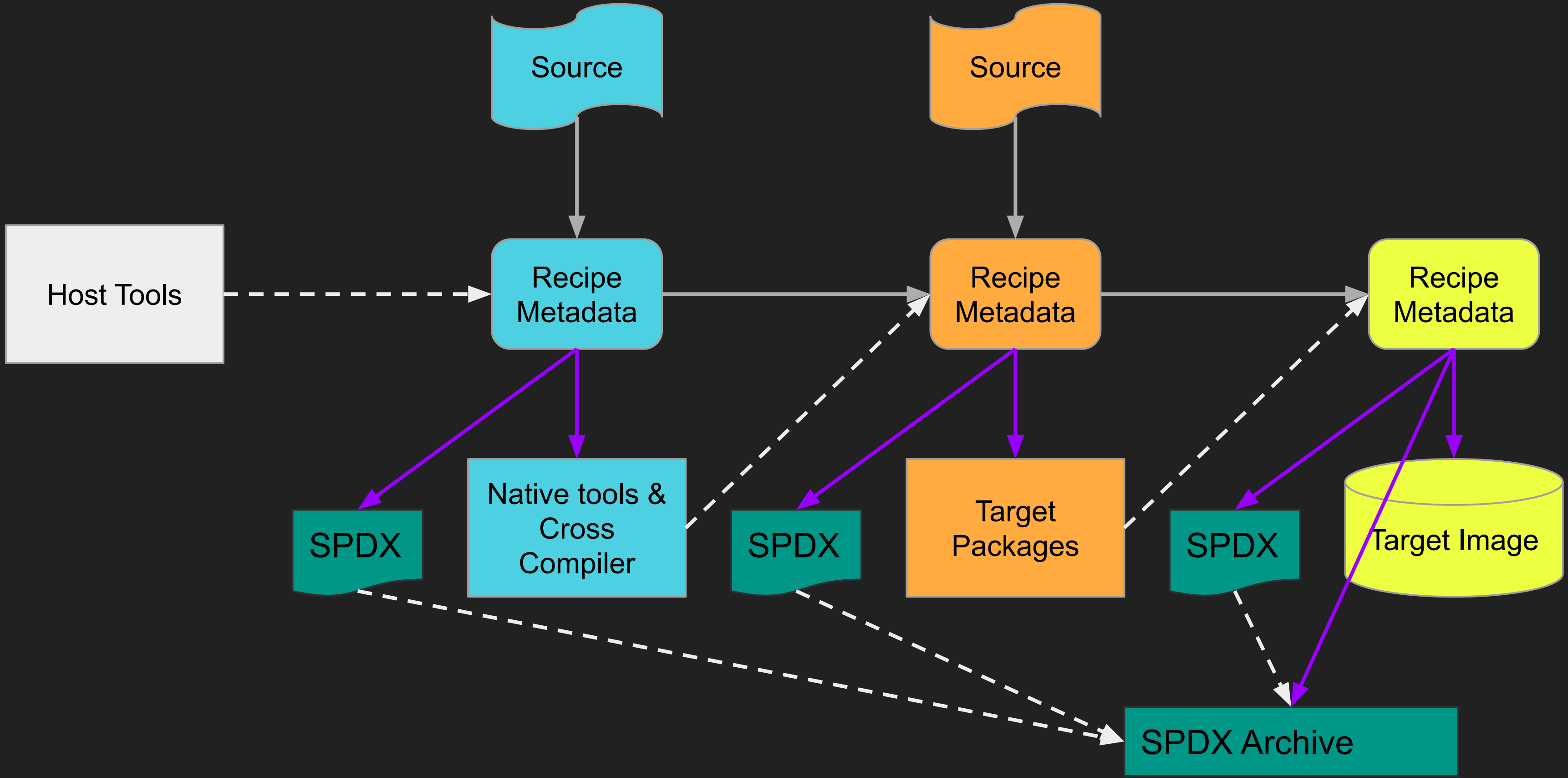
Simplified Build Flow



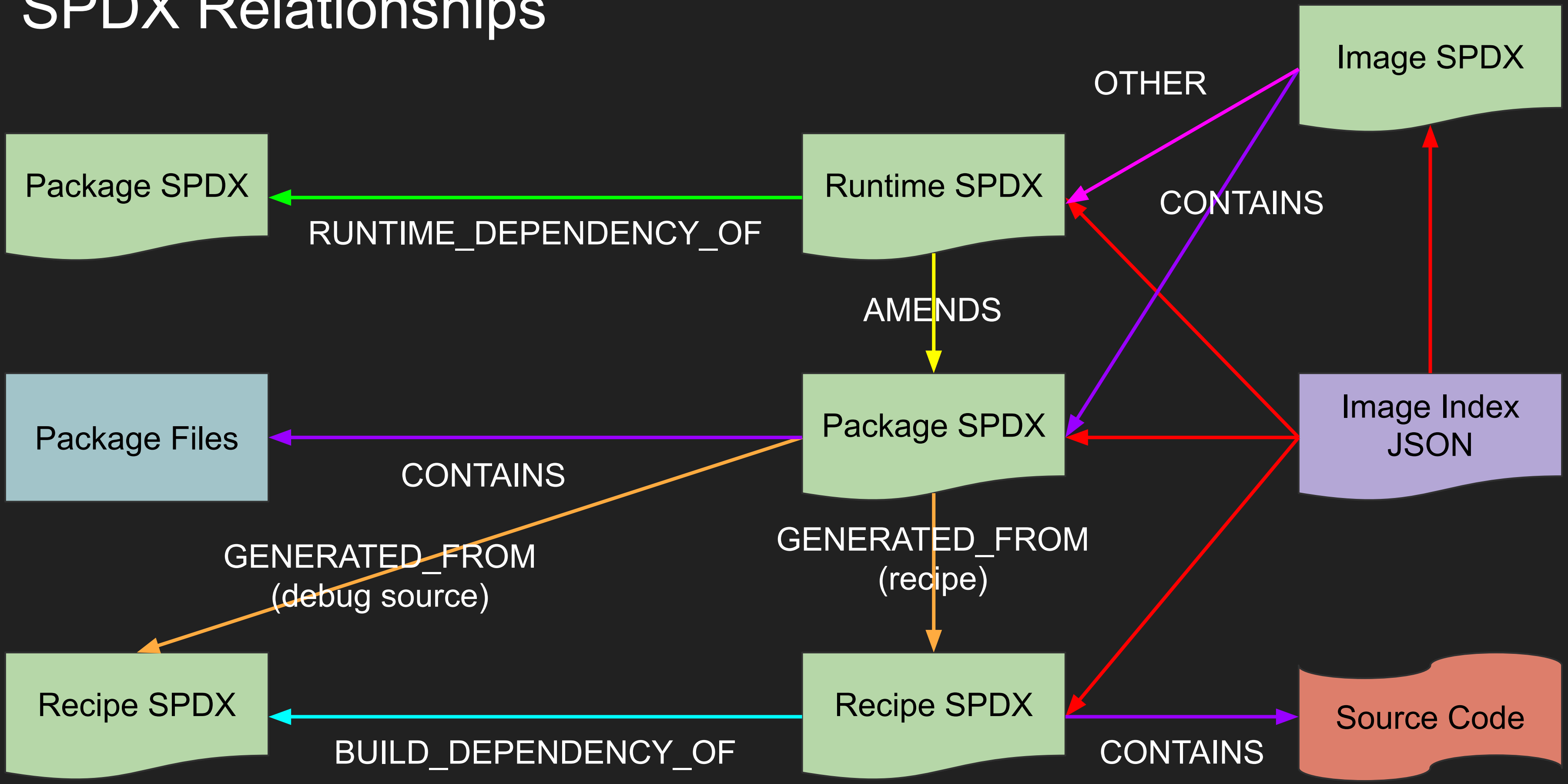
Software Supply Chain derived from build flow



SPDX Generation



SPDX Relationships



More information

Other talks that are specifically about SBoM generation in OpenEmbedded

- https://youtu.be/8X5PWa7A6pY
- https://youtu.be/6zms_qGmVqg
- https://youtu.be/h6PRf4zxnR4

Build Results

- SPDX 2.2 JSON
- Minimal qemu AArch64 system
- Root file system: 14 MB (uncompressed; 2.8 MB compressed)
- Linux Kernel: 20 MB
- SPDX SBoM: 158 MB (uncompressed; 15MB compressed)
  - Sample available on request 😀

Do we really need all this data?

- It's a **lot** of data
- Maybe your end consumers don't care about this
- If you are trying to track down a supply chain attack, you probably do care
- Regulatory requirements may also want a supply chain

Much like the nutrition label vs supply chain: End consumers don't always see the supply chain, but the manufacturer does

If you work on a build tool, consider adding SBoM support

Questions?