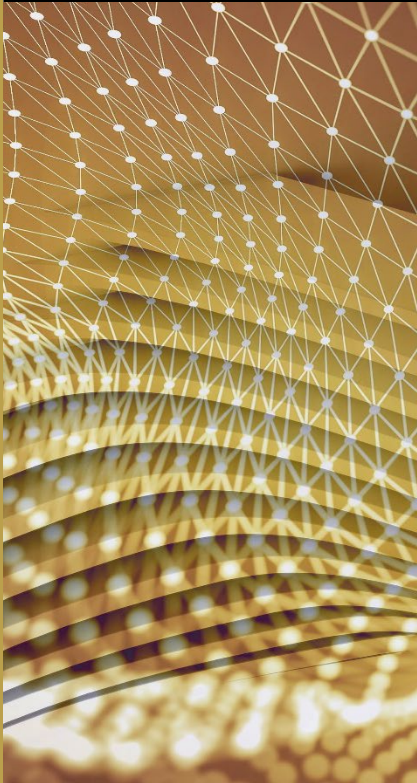
WASHINGTON D.C. | 2024

Carnegie Mellon University Software Engineering Institute

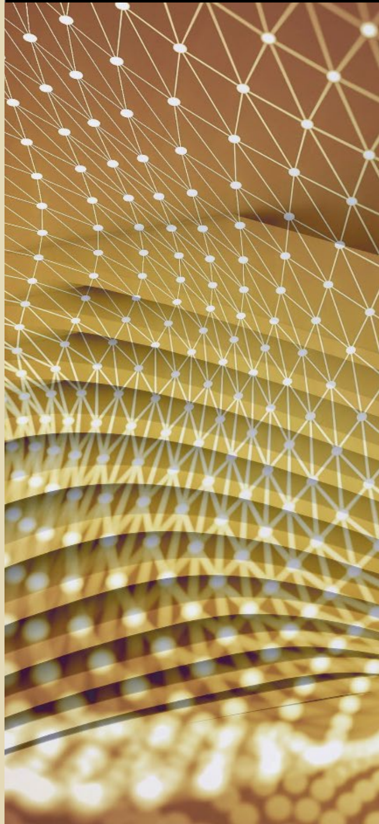


Don't let SBOMs become Yet Another Compliance Activity

SEPTEMBER 18, 2024

Camdon Cady
CTO, Platform One, USAF

Carnegie Mellon University Software Engineering Institute



Agenda

- What are we currently required to do regarding SBOMs?
- What would a compliance-focused SBOM future look like?
- What's missing?
- What are some ideas for what we should be doing?

SBOM Requirements Drivers

- Executive Order on Improving the Nation’s Cybersecurity (EO 14028)
  - “…[NIST] shall issue guidance...[including] providing a purchaser a (SBOM) for each product directly or by publishing it on a public website”
  - “the term ‘Software Bill of Materials’ or ‘SBOM’ means a formal record containing the details and supply chain relationships of various components used in building software.”
  - “Within 60 days…[NTIA] shall publish minimum elements for an SBOM”
- Minimum Elements For a Software Bill of Materials (SBOM) (NTIA)
  - Prescribes a minimum set of Data Fields, Automation Support, and Practices and Processes
    - Data Fields include Supplier Name, Component Name, Version of the Component, Other Unique Identifiers, Dependency Relationship, Author of SBOM Data, and Timestamp
    - Automation support prescribes SPDX, CycloneDX, and SWID tags
    - Practices and Processes discusses Frequency, Depth, Delivery, etc.

Carnegie Mellon University Software Engineering Institute

Possible Compliance-Focused Outcomes

- The Good
  - We will have some sort of SBOM for software that is completely opaque today
  - Some sort of basic vulnerability enumeration is possible independently of the vendor
- The Bad
  - Vulnerability queries will only be as good as the SBOM
    - If a vendor misses some exposure to a vulnerability, it’s probably because they aren’t accurately tracking that that the dependency is in their product
  - It’s just another step to add friction to an acquisition process that’s already difficult
- The Ugly
  - Large bureaucracies often allow the *means* to become the *end*
  - Worst possible outcome: we build a system that parallels the ATO process for SBOM tasks

What’s missing or wrong?

- Assessment of SBOM quality and completeness
  - It’s a harder problem than some people suspected
- Any actual supply chain model
  - Most dependencies aren’t purchased
- Any discussion whatsoever around different language and runtime ecosystems
  - Vastly different norms and capabilities
- A sense of the scale of the problem
  - Preview: it’s really big

Assessment of SBOM Quality and Completeness

- NTIA asserts that build-time SBOMs are higher quality than other SBOMs
  - This makes intuitive sense, because the build system needs to know about every component
  - In reality, most of our tooling today doesn't read, transmit, or create SBOMs

- Errors and mistakes acknowledged, but no criteria for assessing SBOMs
  - For a popular cloud-native security tool that provides an official SBOM with each release:
    - The official SBOM contained 311 components
    - One OSS tool created an SBOM with 319 components
    - Another OSS tool created an SBOM with 320 components

  - The GNU Binutils has a complete copy of zlib in the source tree
    - Currently it lags behind the Binutils version by about a year and a half
    - The vast majority of scanning tools don't find it
    - There's a high-severity CVE in that version of zlib, but it isn't exploitable in this context

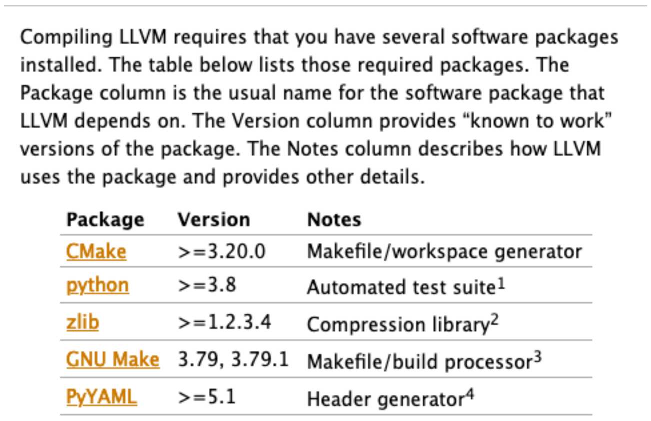
“Supply Chain” Model

- NTIA takes an interesting view of “Supplier” and supply-chain relationships:
  - A supplier is “an entity that creates SBOMs…which may also be known as a manufacturer, vendor, developer, integrator, maintainer, or provider”
  - The only supply-chain relationship captured today is “depends on”. The suggested extension is limited to some form of “derived from” for relationships like forks
- Some (most) maintainers don’t consider themselves a supplier to the USG
  - *I Am Not A Supplier*: https://www.softwaremaxims.com/blog/not-a-supplier
    - “You are not buying from a supplier, you are a raccoon digging through dumpsters for free code.”
  - In the Linux model of having shared libraries on a system, most projects don’t actually distribute their dependencies or know what version will be available
- Proposition: the software supply chain encompasses at least the entire journey from a source-control repository to a deployable artifact
  - If nothing else, the attack against xz should prove this to be true

“Supply Chain” Example

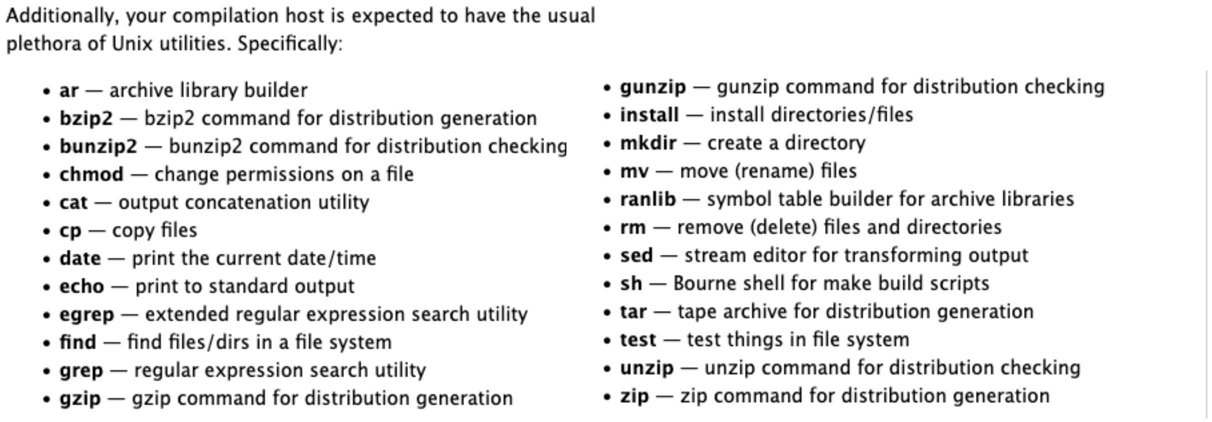
Carnegie Mellon University Software Engineering Institute

- If we accept that a compiler is an essential part of our supply chain, let’s look at what it takes to compile a compiler
  - https://llvm.org/docs/GettingStarted.html#getting-the-source-code-and-building-llvm



Compiling LLVM requires that you have several software packages installed. The table below lists those required packages. The Package column is the usual name for the software package that LLVM depends on. The Version column provides “known to work” versions of the package. The Notes column describes how LLVM uses the package and provides other details.

Package	Version	Notes
CMake	>=3.20.0	Makefile/workspace generator
python	>=3.8	Automated test suite[1]
zlib	>=1.2.3.4	Compression library[2]
GNU Make	3.79, 3.79.1	Makefile/build processor[3]
PyYAML	>=5.1	Header generator[4]



Additionally, your compilation host is expected to have the usual plethora of Unix utilities. Specifically:

- **ar** — archive library builder
- **bzip2** — bzip2 command for distribution generation
- **bunzip2** — bunzip2 command for distribution checking
- **chmod** — change permissions on a file
- **cat** — output concatenation utility
- **cp** — copy files
- **date** — print the current date/time
- **echo** — print to standard output
- **egrep** — extended regular expression search utility
- **find** — find files/dirs in a file system
- **grep** — regular expression search utility
- **gzip** — gzip command for distribution generation
- **gunzip** — gunzip command for distribution checking
- **install** — install directories/files
- **mkdir** — create a directory
- **mv** — move (rename) files
- **ranlib** — symbol table builder for archive libraries
- **rm** — remove (delete) files and directories
- **sed** — stream editor for transforming output
- **sh** — Bourne shell for make build scripts
- **tar** — tape archive for distribution generation
- **test** — test things in file system
- **unzip** — unzip command for distribution checking
- **zip** — zip command for distribution generation

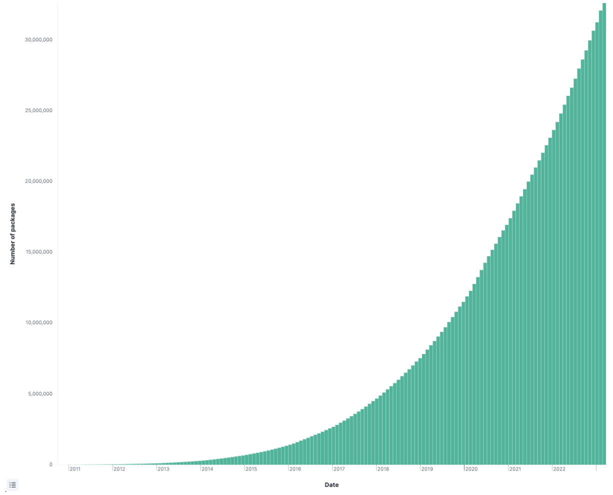
- Every single application listed there can impact the compiler binary, which can impact everything the compiler ever compiles

Carnegie Mellon University Software Engineering Institute

Ecosystem Differences and Scale

- The core infrastructure of the “supply chain” are the package managers
  - **Policy and tools need to align to workflows and norms within the communities**
  - **We have to find some way to tackle the fact that system libraries are utilized by most languages**

- To make SBOMs work at scale, we probably need support from the registries
  - As of March 2013: there were 32M NPM packages
  - The vast majority are single-maintainer
    - Including the ones that are commonly used
  - https://anchore.com/blog/open-source-is-bigger-than-you-imagine/



What do we do instead?

- VirusTotal for SBOMs
  - The reality of open source is that millions of projects use the same dependencies
  - For any given component version/source, hundreds of entities could submit SBOMs
    - Anchore, Aqua, Prisma, Phylum, RapidFort, etc.
    - Security researchers
- Kalman Filters for SBOMs
  - With enough data, we could start to assess with some confidence interval that *foo* depends on *bar*
  - What do you do when a vendor tells you their product includes *libpng* but not *libz*?
- SBOM Challenge Problems
  - List of Go projects that only report pseudo-versions, projects with vendored dependencies, etc.
  - Reference corpus to do comparison, like we do with SAT, SMT, etc.

What do we do instead (continued)?

- It’s not about **if** you’re collecting an SBOM…
  - It’s about how good the SBOM is
  - It’s about what you’re doing with the SBOM
- Maybe it’s more science than commerce
  - Instead of an SBOM shipping with every software transaction…
  - As a community we work to discover the SBOM for every piece of software
- Re-frame the “Software Supply Chain” conversation
  - Are these part of your supply chain:
    - Compilers, Linkers, and Loaders?
    - Runtimes and Standard Libraries?
- Get involved!
  - As I was writing this, the OpenSSF announced Bomctl
  - **https://openssf.org/blog/2024/09/05/simplify-sbom-management-for-developers-introducing-bomctl/**

Contact Info

Carnegie Mellon University Software Engineering Institute

Camdon Cady

Email: camdon.cady@us.af.mil
@camdoncady@infosec.exchange