Revisiting Software Bill of Materials

Lars Bendix, Lund University
Andreas Göransson, QCM
sneSCM.org

https://fileadmin.cs.lth.se/cs/Personal/Lars_Bendix/Research/SBoM/

Sponsor & Org

UNIVERSITÀ DEGLI STUDI DI PARMA

engage LABS

innprojekt

iSolutions

DRILLING PERGEMINE CONTRACTOR

LIME ware

daniela malvisi

getlatestversion.it

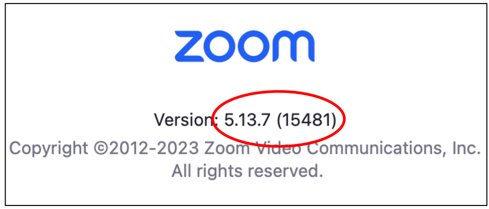
Agenda

- DevOps Heroes SBoM motivation
- SBoM history
- SBoM - Use Case categories
- General SBoM considerations
- Lessons learned
- Discussion points
- Q&A

Motivation

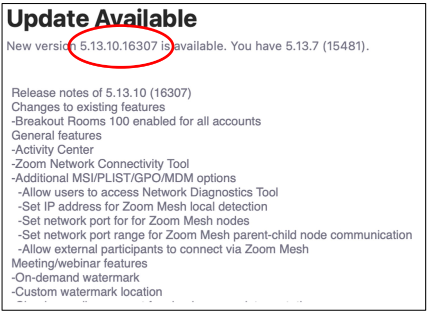
- Will be an external requirement - US & EU
- Should have been an ***internal*** requirement since the 1980’s
- ***Dev*** are the optimal *producers* of SBoMs
- ***Ops and Dev*** should be heavy *consumers* of SBoMs
- SBoMs needed for all systems in environment (know what you have)
- If you know what you have, then change control (and cyber security) is easier
- SBoM knowledge is “fragmented”
- How you can ***use/exploit*** SBoM - the full story

These are examples of SBoMs



Version: 5.13.7 (15481)

Copyright ©2012-2023 Zoom Video Communications, Inc.
All rights reserved.

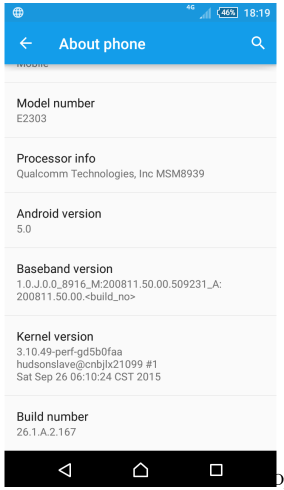


Update Available

New version 5.13.10.16307 is available. You have 5.13.7 (15481).

Release notes of 5.13.10 (16307)
Changes to existing features
-Breakout Rooms 100 enabled for all accounts
General features
-Activity Center
-Zoom Network Connectivity Tool
-Additional MSI/PLIST/GPO/MDM options
 -Allow users to access Network Diagnostics Tool
 -Set IP address for Zoom Mesh local detection
 -Set network port for for Zoom Mesh nodes
 -Set network port range for Zoom Mesh parent-child node communication
 -Allow external participants to connect via Zoom Mesh
Meeting/webinar features
-On-demand watermark
-Custom watermark location

H&SBoM



Model number
E2303

Processor info
Qualcomm Technologies, Inc MSM8939

Android version
5.0

Baseband version
1.0.J.0.0_8916_M:200811.50.00.509231_A:
200811.50.00.<build_no>

Kernel version
3.10.49-perf-gd5b0faa
hudsonslave@cnbjlx21099 #1
Sat Sep 26 06:10:24 CST 2015

Build number
26.1.A.2.167

SBoM anno 2020s

- Executive Order 14028 signed in 2021
- Strengthen the US' ability to respond quickly and efficiently to cybersecurity vulnerabilities
  - Heartbleed, SolarWinds, Colonial Pipeline hack
- US Department of Commerce and NTIA
  - The Minimum Elements For a Software Bill of Materials
  - "List of ingredients" with focus on vulnerability scans

Fast rewind - Back to the future

SBoM anno 1980s - part I

Wayne Babich: “Many times the fastest approach to finding a bug is not analysis of the program itself, but analysis of the *history* of the program – how it was created. The history of the program is called its *derivation*.”

A precise *derivation* of a program or module requires:

- an identification of the *tool* that created it
- an identification of the data that was *input* to the tool
- an identification of the *options* and *arguments* given to the tool
- the *reason* why that particular data, arguments, and options were given to the tool
- the *person* who was responsible for creating the data
- the *date* and *time*

An ounce of derivation is worth a pound of analysis!

Back to the future II

MICHAEL J. FOX
CHRISTOPHER LLOYD
Getting back was only the beginning.
STEVEN SPIELBERG Presents
BACK TO THE FUTURE II
A ROBERT ZEMECKIS Film
MICHAEL J. FOX
CHRISTOPHER LLOYD BACK TO THE FUTURE PART II LEA THOMPSON
OPENS NOVEMBER 22ND

Imaged by Heritage Auctions, HA.com

SBoM anno 1980s - part II

- Clearmake automatically create SBoMs called Configuration Records
- Re-use build artifacts like object files in a smart and secure way
- Configuration Record contains information about input files, build environment and output files
- Configuration records can be read and used by machines and audited manually by developers

Overview – SBoM use case categories

- Bo-Materials
  - Search for object by UID
- Bo-Process
  - Reuse
  - Debugging
  - Rebuild
  - Build audit
- Bo-Information
  - Licence tracking
  - Export control
  - Legal aspects
  - Test-related matters
  - Information sharing

BoM use case category

Search for an object by UID

Software Composition Analysis (SCA):

- Are features 12, 15 and 19 included in this binary for QA?
- We get a new (binary) patch - is it already in there?
- What is “operating” on our systems (Ops)

Logon module - source code ver. 2.1 (compiled on several occasions)

SBoM = **BoM**

BoP use case category

Reproducibility - or producibility - is a core concept for software configuration management

For that we need the exact source code - obviously…

But that is not sufficient:

- Escrow development
- Fixing a bug in a 13-year old petro-chemical installation

To the degree of what you consider “identical” we must include the process:

- Tools, versions, options, environments, …

SBoM = BoM + **BoP**

BoI use case category

Information about the artifacts in the SBoM needed for communication, audits, certifications, …

Test related matters:

- In case a program or system can affect human safety, test information can be vital to keep in an SBoM
  - Test cases, test results, test environments (HW and SW), …
- Possible need to provide proof that test cases have been performed in a legal dispute
- Test information can also be used as a “quality stamp”
  - Our software passes these test and therefore complies to regulation X

SBoM = BoM + BoP + **BoI**

SBoM implementation considerations

- Level of detail of SBoM
- Availability of SBoM
- Automation - both in creation and using
- Static and dynamic status of SBoM data
- Keeping the SBoM up to date
- SBoM for tools and environments too - and Everything
- SBoM and Microservices
- Unique identifiers

Key takeaways

- SBoMs for **consumers**:
  - External consumers: List of ingredients & Vulnerability scan
  - Internal consumers: 100s of use cases, 10+ categories, 3 aspects
- SBoMs for **producers**:
  - "Builders" of a binary are at the origin of (most) SBoM data
  - Automation is at the heart of DevOps Heroes
- Doesn't it look a little like:
  - CMDB: Configuration Items with data (attributes and relations)
  - Configuration Status Accounting

Looking forward

- Those were the results – so far…
- Leaving a lot of interesting questions to dig deeper into
- Let's dive into it…..

Who can benefit from SBoMs?

- Externally (consumers):
  - End users
  - Buyers
    - Applications
    - Libraries
- Internally (producers):
  - Developers
  - Quality Assurance
  - Cyber Security / Supply Chain Security / SCA
  - Software Configuration Management

What is an SBoMs?

- A "list of ingredients"++
- A machine-readable inventory
- Flat – or hierarchical (SBoM of SBoMs)?
- Dependencies and relations (traceability)
- Metadata
- Static or dynamic?
- Format of SBoM

Who can create an SBoM – and how?

- Those who *build* the application/binary
- At the *time* when they build the application/binary

- Unique IDs (“globally”?) – PURLs?
- SBoM and blockchain
- Format – flat/hierarchical – static/dynamic – kind of data

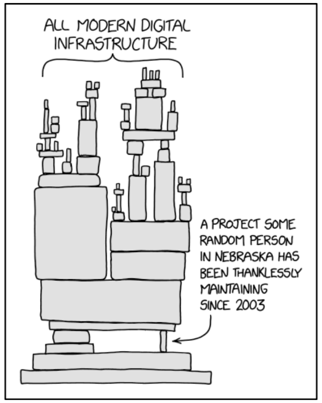
How to use (create value from) SBoMs?

- Diff on SBoMs – bug hunting
- Dependencies and relations (traceability)
- Reproducible builds
- Upgrading installations/production
- Risk analysis
- To identify – to notify – and to recall

SCM and SBoMs?

- We can deliver pizzas – any pizza you like – we know how to produce pizza – we hope that you know how to eat them
- CMDB: Configuration Items with attributes and relations
- CSA: all the questions you never had answers for before
- Keep **ALL** code in the CMDB
- If we know what we are using (have in our repo), we can ask interesting questions like: Do we need to use 84 versions of Spring? Do we need to have 15 different XML parsers?
- Adam Thornhill: Code as a Crime Scene (CMDB & CSA)

Missing Configuration Items…


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

https://xkcd.com/2347/

SBoM from an **SCM** point of view

- Developer: what a cool library, I could use that!
- ?When is EOL
- ?What is the maintenance rating
- ?What are the number of dependencies
- ?Will all our code become open source
- ?.....

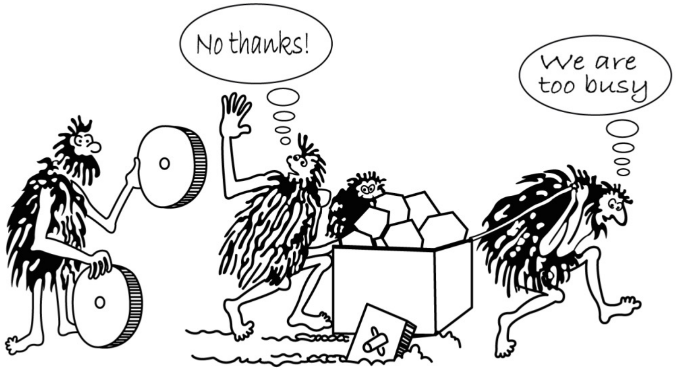
- Matteo Emili (101): la sicurezza è responsabilità comune

Types of SBoMs?

- Runtime
- Deployed
- Analyzed
- Build
- Source
- Design

- Have they never heard about the concept “baseline”?

Re-inventing the wheel/SBoM?



”Useless” pieces of information?

- 80-90% of an application’s code is “external”.
- Six out of seven vulnerabilities come from transitive dependencies.
- Organisations keep fetching known vulnerable versions of components that are already fixed (30% log4j after 18 months).
- CVE / VEX / VDR / GUAC / TACOS / SLSA
- ML-BoM

Thanks - Q&A

A couple of interesting quotes?

- Afraid of sharing SBoMs? Couldn't we just "share" the **libraries** we use (which is probably where the "public" vulnerabilities are).
- Log4j is **not** the normal case – "ordinary" bugs are.
- I thought that was a reputable software company. Why are they using these three **crappy** libraries?
- When we generate SBoMs all the garbage comes out, it helps the developers have better **hygiene**, your signal to noise ratio gets better when you scan repos.