

# EU CRA: Time to Engineer Up!

# Up Your Engineering Game: Why You Need SBOMs and Dependency Management

Peter N. M. Hansteen



<https://nxdomain.no/sbom.pdf>



# EU CRA: It's Later Than You Think, Time to Engineer Up!

- *December 12 2027, it's too late.*
- On December 11 2027, the European Union Cyber Resilience Act (CRA) enters fully into force

=> Products with digital elements must come with full overview of and insight into

- Description and origin of all components
- Description and origin of all dependencies

Or, no CE mark of approval for you or your product

Reporting required for all vulnerabilities and incidents will have been required since September 11, 2026.

[Full text article: *EU CRA: It's Later Than You Think, Time to Engineer Up!*]

# Upping Your Engineering Game

- Are you
  - a coder, or
  - an engineer who codes?
- This is about upping your engineering game
  - Taking your software engineering to “real engineering” level
  - Focus on doing things right, proper practices for quality engineering

The anecdote about the iron ring worn by Canadian engineers is useful to calibrate your mindset and level of ambition.

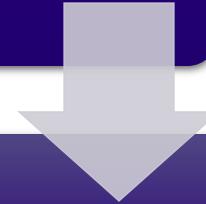
# Dear Developer, do you know what your code does?

You wrote it, but do you know \*all\* it does? –

TL; DR:

Your code has dependencies.

The rest of the world, including customers recently started to care about which ones.



But first, a bit of history.

# “Just a bit of typing”

Software is a new phenomenon.

Poorly understood by the public

Considered “just a bit of typing”

Not important in itself, but a necessary evil

Poorly understood by non-techies

# But then suddenly software turned important

First, the Internet happened and raised our profile

Bugs in local code started biting

Bugs in dependencies started biting – log4j comes to mind

Process bugs started happening – Solarwinds SUNBURST comes to mind



# Dependencies became a thing

- Next up, memes like ->  
(see [XKCD #2347](#), but also the [explainer](#))

= >

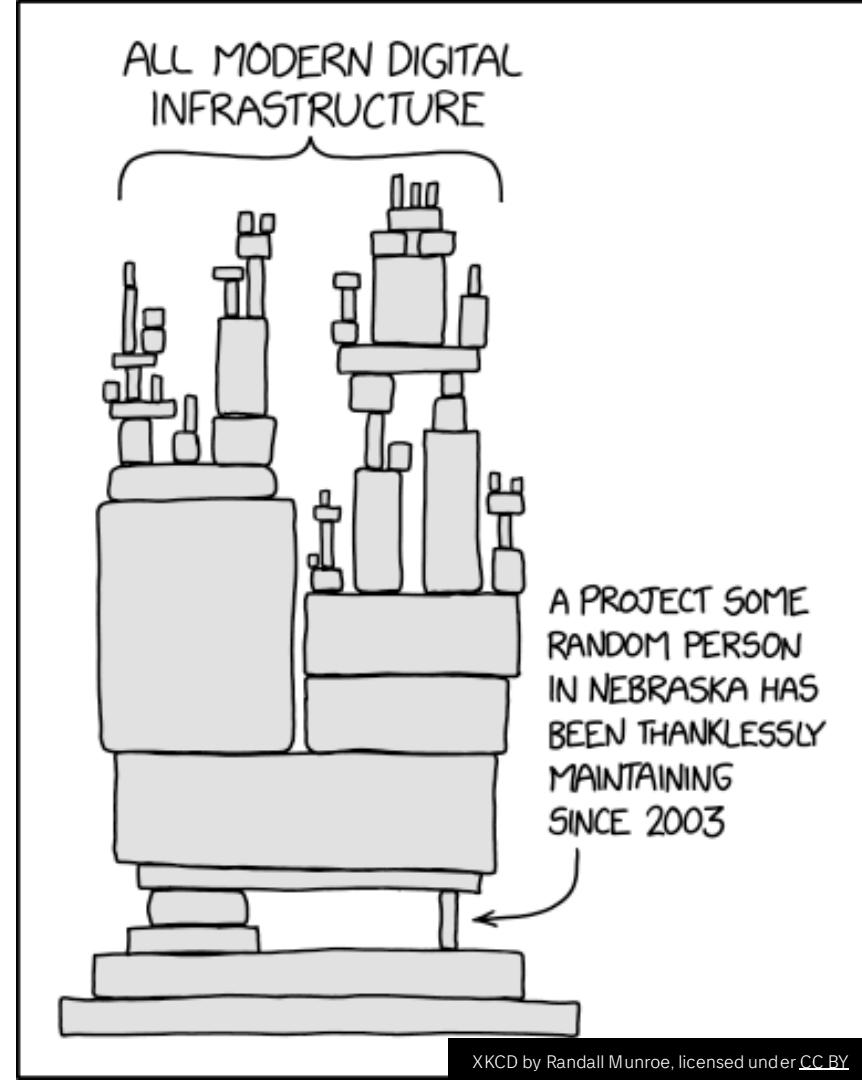
Supply chain management

and

*dependency management*

came into focus

!! Known your components and dependencies !!



# No project is an island

- We write software
- Which depends on other software
- Which interacts with other software
- Which again interacts with other components (hardware, humans)
- To run important stuff
- Nothing exists in actual isolation – No project is an island

# Learn from those who build important things

- What to those who build important things do?
- Real engineers with real plans

are required to submit a

“Bill of Materials”

aka BOM for each delivery

HMS QUEEN ELIZABETH



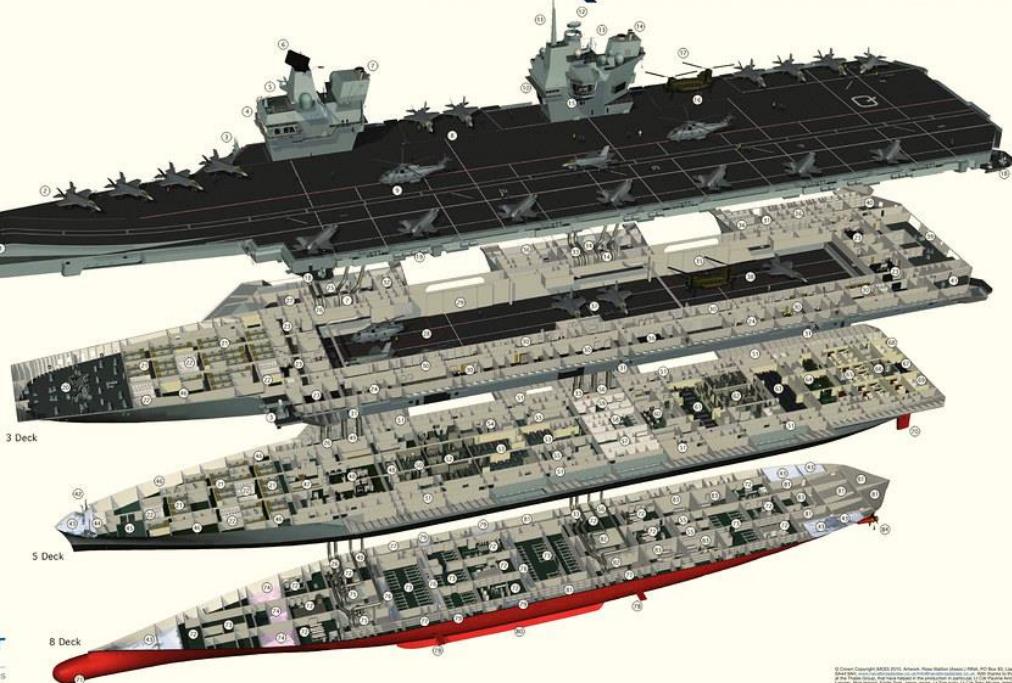
- 1 Aviation store
- 2 F35 Lightning II
- 3 Forward deck edge close-in weapons system
- 4 Forward deck edge close-in weapons system
- 5 Navigation radar
- 6 Radar mast
- 7 Forward engine and gas turbine uptakes
- 8 Merlin helicopter
- 9 After aircraft lift
- 10 Emergency landing bridge
- 11 Medium range radar
- 12 After engine room and gas turbine uptakes
- 13 Merlin helicopter
- 14 After aircraft lift
- 15 Automatic small craft lane gate
- 16 Forward landing deck
- 17 Jetty
- 18 Jetty
- 19 Ship's office complex
- 20 Ship's office complex
- 21 Ship's office complex
- 22 Ship's office complex
- 23 Ship's office complex
- 24 Forward gas turbine space
- 25 Ship's office complex
- 26 Air filtration units
- 27 Ship's office complex
- 28 Fire protection system
- 29 Ship's office complex
- 30 Ship's office complex
- 31 Air squadrons complex
- 32 Ship's office complex
- 33 RN police office and cells
- 34 Ship's office complex
- 35 After gas turbine space
- 36 Air filtration units
- 37 Ship's office complex
- 38 All landing deck
- 39 All landing deck
- 40 All landing deck
- 41 Port mooring deck
- 42 All landing deck
- 43 Water ballast compartment
- 44 Control room
- 45 Junior officer recreation space
- 46 Mess area
- 47 Senior officer recreation space
- 48 Mission systems complex
- 49 Navigation equipment in relatives
- 50 Bakery
- 51 Dining room
- 52 Junior officer galley
- 53 Senior officer galley
- 54 Main mess area
- 55 Main mess area
- 56 Main mess area
- 57 Headspace
- 58 General medical area
- 59 General medical area
- 60 Service officer driving bay
- 61 Commanding officer's suite
- 62 Commanding officer's suite
- 63 Wardroom areas
- 64 Wardroom areas
- 65 Flag and commanding officers' gallery
- 66 Flag and commanding officers' gallery
- 67 Commanding officer's suite
- 68 Flag and commanding officers' gallery
- 69 Commanding officer's suite
- 70 Rubber point and starboard
- 71 Rubber point and starboard
- 72 Forward engine rooms
- 73 After engine room
- 74 Forward engine rooms
- 75 Stabiliser compartment (port and starboard)
- 76 Stabiliser compartment (port and starboard)
- 77 Gear connection tank (port and starboard)
- 78 Gear connection tank (port and starboard)
- 79 Propeller (port and starboard)
- 80 Propeller (port and starboard)
- 81 Propeller (port and starboard)
- 82 Propeller (port and starboard)
- 83 Propeller (port and starboard)
- 84 Propeller (port and starboard)

aircraft carrier  
alliance

Delivering the Nation's Flagships

HMS QUEEN ELIZABETH

# inside HMS QUEEN ELIZABETH



© Crown Copyright 2010. Aircraft Carrier Alliance. Royal Holloway (00000000000000000000000000000000). All rights reserved. This material is provided under the terms of the Defence Copyright Licence. You have neither the permission nor the authority to copy or otherwise use this material without the express written consent of the Defence Copyright Licensing Agency. This material may not be reproduced in whole or in part, stored in a retrieval system or transmitted in any form or by any means without the prior permission of the Defence Copyright Licensing Agency.

# What do engineers do?

- Create a plan
- A Bill of Materials (BOM) is required for all deliveries
- The BOM lists all component parts
- The BOM typically also references and serves as reference for maintenance docs

OM and can make changes.

[Upgrade to Team](#)

Assembly Name: PUMP S55-Main-Boat Pump Assembly

Revisions (Latest state) | **Production planning:** Order BOMs | Part and Catalogs

Material	Quantity	Part ...	Link to...	Description	Mass ..	Den..	Supplier V...	Unit Cost
Stainless Steel	1	Pump 1_2 ...	<a href="#">Boat Pump...</a>	Pump body	3.56	0.28	<input type="checkbox"/>	40
	1	D-85-STV...	<a href="#">Boat Pump...</a>	Pump Cap ZZZZ	0		<input checked="" type="checkbox"/>	8.5
Stainless Steel	5	SH100	<a href="#">Socket butt...</a>	Socket Head S...	0.01	0.28	<input checked="" type="checkbox"/>	1.2
Stainless St...	1	D85-55XU...	<a href="#">Boat Pump...</a>	Gasket/Spacer ...	0.01	0.28	<input checked="" type="checkbox"/>	3.2
Stainless St...	1	Imp 1-2-KK	<a href="#">Boat Pump...</a>	Impeller descrip...	0.57	0.28	<input type="checkbox"/>	13.3
Stainless St...	1	L-104	<a href="#">Boat Pump...</a>	Shaft	0.34	0.28	<input type="checkbox"/>	50.4
Stainless St...	1	6203-02-0...	<a href="#">Boat Pump...</a>	Impeller part	0.17	0.28	<input type="checkbox"/>	20.3
Silicone Rub...	1	3222	<a href="#">Boat Pump...</a>	Seal	0.01	0.04	<input checked="" type="checkbox"/>	3.2
Stainless St...	1	6203-01-0...	<a href="#">Boat Pump...</a>	Radial bearing	0.17	0.28	<input type="checkbox"/>	20
Aluminum - ...	1	HDWRE C...	<a href="#">Boat Pump...</a>	Circlip 1.25	0	0.1	<input type="checkbox"/>	0.4
	4	NM100C2	<a href="#">Boat Pump...</a>	Socket Head S...	0		<input type="checkbox"/>	0.5

# Libre software has package management already



In the Open source / Free software / Libre Software world, we have a long history of package management systems that take care of *runtime dependencies* for installation and configuration



Those in turn depend on build systems that take care of *buildtime dependencies* and to generate installable packages



The information to build a Software Bill of Materials is right there in our source code



In addition, we tend to have runtime vulnerability scanners to look for bugs and reported weaknesses and CVEs

# Introducing: A Software Bill of Materials (SBOM)

- Nuts and bolts aside, the legal framework is:

[US Executive Order 14028 of May 12, 2021, Improving the Nation's Cybersecurity](#), (summarized) – USA: [NTIA.gov](#)

or

[EU Cyber Resilience Act \(CRA\)](#), also more reader friendly at the [Cyber Resilience Act](#) start page (EU/EEA)

- This legislation is in force or is soon to be, main concepts are taking the *dependency* information we have and present for compliance in actionable form. Inclusion of security info such as CVEs much appreciated.
- The main takeaway:

The information we need is in our code; we need to make generating for compliance effortless and painless.

# We're real engineers now, Sparky! We have tools!

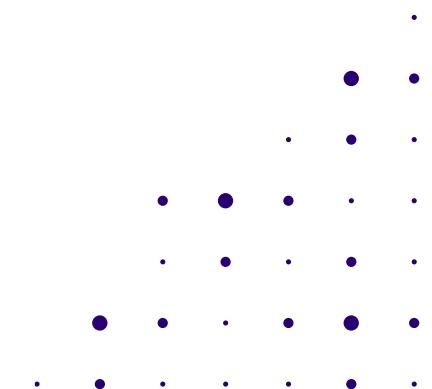
- Mostly from open source circles, two SBOM specs with tool suites emerged:

Open Worldwide Application Security Project (OWASP) [CycloneDX](#)

and

Linux Foundation's [System Package Data Exchange specification \(SPDX\)](#)

- For tools and guidance on both, [awesome-sbom](#) (github.com) is a good place to start.
- Conversion between the two formats is common and available from several tools



# Tools and how to use them

- Using syft to generate and bomber to present, go to a project directory and do

```
$ syft . -s all-layers -o cyclonedx-xml | xq
```

or the json equivalent

```
$ syft . -s all-layers -o cyclonedx-json | jq
```

- If you want a html report with known vulnerabilities

```
$ syft . -o cyclonedx-json | bomber scan --provider ossindex --output=html
```

**Note:** For this particular command to work, you also need to supply provider login credentials (available with free registration), see the [Bomber provider documentation](#).

# Your SBOM, the build artifact

You probably want your pipeline to produce SBOM build artifacts.

For inclusion in a CI/CD pipeline you may want

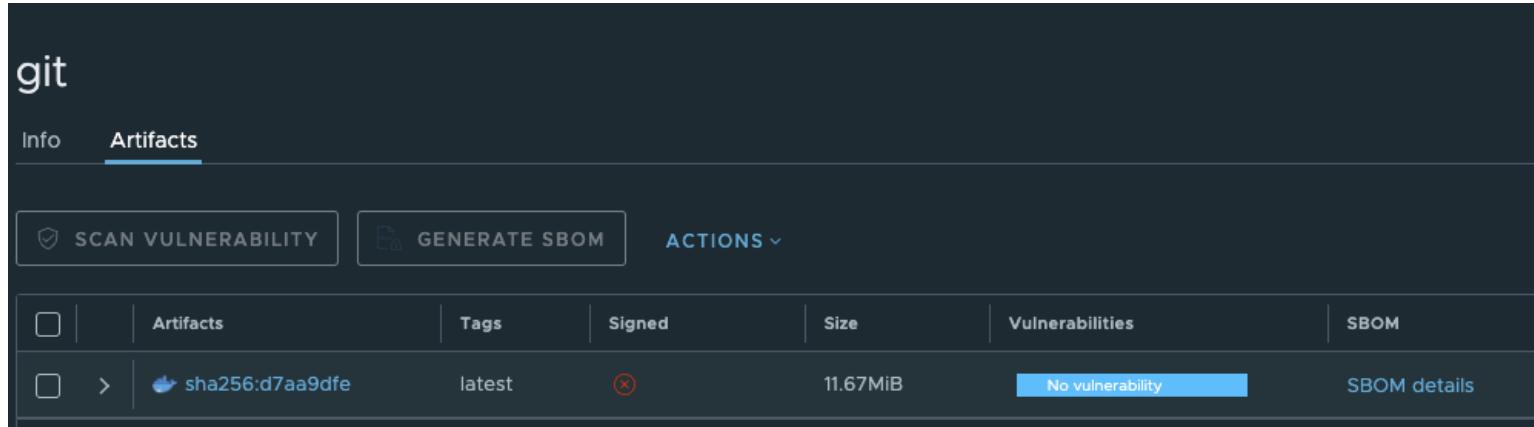
```
# Make sure you include the - character at the end of the command.  
# This triggers bomber to read from STDIN  
syft packages . -o cyclonedx-json | bomber scan --provider ossindex --output json -
```

**Note:** For this particular command to work, you also need to supply provider login credentials (available with a free registration), see the [\*Bomber provider documentation\*](#).

Also, reports and conversions, see the docs for your toolset or get inspired by looking at [awesome-sbom](#)

# Your tools may already have (some of) this

- Some tools have SBOM generating and presentation already. Harbor has automatic SBOM generation on image push using trivy:



The screenshot shows the Harbor UI for an artifact named 'git'. The 'Artifacts' tab is selected. At the top, there are buttons for 'SCAN VULNERABILITY' and 'GENERATE SBOM'. Below this is a table with columns: Artifacts, Tags, Signed, Size, Vulnerabilities, and SBOM. The 'git' artifact has a checked checkbox in the first column. The 'Tags' column shows 'latest'. The 'Signed' column has a red 'X' icon. The 'Size' column shows '11.67MiB'. The 'Vulnerabilities' column is highlighted in blue with the text 'No vulnerability'. The 'SBOM' column contains a link 'SBOM details'. To the right of the table, there are several small circular icons.

Artifacts	Tags	Signed	Size	Vulnerabilities	SBOM
git	latest	✗	11.67MiB	No vulnerability	<a href="#">SBOM details</a>

# Track your dependencies on the fly

- When you have the tools in hand, when a customer asks about your tooling:

you could do

```
$ cd myproject  
$ cdxgen .
```

and get a `bom.json` with all your customer wants to know (see the article)

# There is more -



Your SBOM-savvy stakeholders may already have appropriate processing tools at their end. Poke them!



Other aspects of BOM generation are being worked on, including  
OBOM (Operating system Bill of Materials)  
SaaSBOM (Software as a Service Bill of Materials)  
CBOM (Cryptography Bill of Materials)  
... and others



There is more to SBOMs out there, to impress colleagues and customers!

# Now It's Your Turn: Get The Tools

- For further exploring, you need to get the tools:
- cdxgen lives at <https://github.com/CycloneDX/cdxgen>
- syft can be found at <https://github.com/anchore/syft>
- bomber is at <https://github.com/devops-kung-fu/bomber>

Follow the install instructions for each, On Linux, they are likely in your package system. On macOS, use brew,

```
$ brew install $toolname
```

Again, follow the install instructions for your environment.

Or use

```
$ git clone $toolname
```

(As one does)

# Tools in Hand, Dig Into a New Project

Find a project. Do you have one of your own? Get the code.

Go to the project main directory. Run

```
$ cdxgen .
```

Watch the output. It will be useful to run this in a script(1) session.

*This is where the fun starts -*

# Tools in Hand, Dig Into a New Project (2/2)

Watch the output. It will be useful to run this in a `script(1)` session.

- Find out:

What is the number of dependencies for this code base? How many direct dependencies? How many indirect ones (dependencies of dependencies)?

- Does the code base itself have any known problems, reported as `CVEs`? How many for the dependencies?

You may choose to inspect the tools themselves.

Remember, there may be challenges, like in [this session](#). Don't let noisy errors discourage you!

*This is where you pick up your engineering game*

# Resources for further reading

- Linux Foundation Training:  
Automating Supply Chain Security: SBOMs and Signatures (LFEL1007) is a short but information- and reference-filled introduction (free, requires registration, gives you a badge at the end)  
Understanding the EU Cyber Resilience Act (CRA) (LFEL1001) Focused on the EU CRA, gives an overview with lots of useful references, nominally a 1 hour course worth taking
- The Software Bill of Materials home page at NTIA is the mother ship of SBOM documentation
- Browse OWASP CycloneDX for all things about the CycloneDX specification and related tools, also their CycloneDX tool center
- Browse the System Package Data Exchange specification (SPDX) for all things SPDX (supported by the Linux Foundation), including copious linked reference material
- awesome-sbom is a curated list of SBOM tools and resources
- EU residents will want to poke around the Cyber Resilience Act site for reference
- Brewing Transparency: How OWASP's TEA Is Revolutionizing Software Supply Chains is a summary of recent work on OWASP Transparency Exchange API (TEA)
- SBOM buyer's guide: 8 top software bill of materials tools to consider is a readable overview of (some) SBOM tools
- Olle Johansson's FOSDEM presentations are among several good SBOM talks at that conference (search the site for more)
- Peter N. M. Hansteen: Open Source in Enterprise Environments - Where Are We Now and What Is Our Way Forward (2022, also here) has some insights on how open source software plays a crucial role in enterprise environments and elsewhere
- Peter N. M. Hansteen: No Project Is an Island: Why You Need SBOMs and Dependency Management (also here)
- Peter N. M. Hansteen: EU CRA: It's Later Than You Think. Time to Engineer Up! (this presentation in article form, also here, slides)

Thank you

↗ tietoevry

