What has (can) the EU Cyber Resilience Act done (do) for you?

Peter N. M. Hansteen

June 19, 2026 at BSDCan 2026 in Ottawa, Canada

https://nxdomain.no/~peter/BSD_CRA.pdf

TL; DR: It’s Not the End of Open Source

- You may have encountered the claim that the *European Union Cyber Resilience Act (CRA)* *taking effect across 2026 and 2027 would mean*

THE END OF OPEN SOURCE
AND THE WORLD AS WE KNOW IT

It will

not.

In this session: an overview of the practical implications, context, and opportunities, with a particular focus on the BSDs

Yes, It's Later Than You Think

- *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*.

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

The European Union Sets the Standard From Here On

The CRA is part of an expanding body of legislation(*) on IT and *products with digital elements* in the EU and associated countries and territories, to ensure

- Product safety
- Personal health and wellbeing
- Civil rights

Early work on cyber security and digital resilience was coordinated across the Atlantic, with US Executive Order 14028 of May 12, 2021, *Improving the Nation's Cybersecurity* (2021), and *EU Cyber Resilience Act (CRA)* in 2024 with timeframes and rules in reasonable sync.

The US partially withdrew Executive Order 14028 January 29, 2026, making specifications optional.

The EU CRA now sets the standard worldwide.

(*) Including *General Data Protection Regulation (GDPR)*, *Regulation of the Digital Operational Resilience of the Financial Sector (DORA)*, and *Directive on Network and Information Systems (NIS2)*

Manufacturers, Stewards and Developers

- **Manufacturers**: Businesses that put products on the market to be available in the EU, and
  “...and markets them under its name or trademark, whether for payment, monetisation or free of charge (see *CRA article 3 (13)*)”
  ” ... in the course of commercial activity " (see CRA Article 3(22) “

  Open Source Stewards: Organizations that coordinate open source projects, typically foundations (not for profit corporations),
  “.. other than a manufacturer, ... providing support for the development .. of open-source software” (see *CRA article 3 (14)*)

  Open Source Developers and Maintainers: People who write and maintain open source software.
  “... This Regulation does not apply to natural or legal persons who contribute with source code to products with digital elements qualifying as free and open-source software that are not under their responsibility. (from *CRA recital 18*)”

Your Duties as an Open Source Developer or Steward

- Individual open source developers
  - no formal obligations under CRA
  - may be encouraged to grab the opportunity and adopt best practices
- Open source stewards
  - Usually foundations (not for profit corporations)
  - Usually maintain infrastructure and organization to support developers
  - Should assist and facilitate security and issues reporting

  Also see *CRA article 24* and recitals *17*, *18* and *19* for more nuance

On the Other Hand, If You Are a Manufacturer

- Manufacturers bear accountability for the product. Ensure that it is
  - Fit for the purpose it is designed to serve
  - Aligned with all applicable requirements (certifications) within the European Union
  - Supported by the necessary documentation and technical information, including
    - Bills of Materials listing all components and dependencies,
    - Software Bill of Materials (SBOM) detailing the product's digital components and the dependencies between them.

  After market entry, maintain the product's security on an ongoing basis
  - Notify the reporting authority and customers of incidents and known defects
  - Deliver updates and remediation measures to customers
  - Violations and failures to comply can trigger mandatory product recalls and fines of up to *15 million Euros* or *2.5% of global turnover*, depending on which amount is greater

But Why? Why Now?

Historically, software was

- Poorly understood by the public
- Generally considered unimportant
- When ‘not unimportant’, treated as trade secret, precluding real security / quality assessment

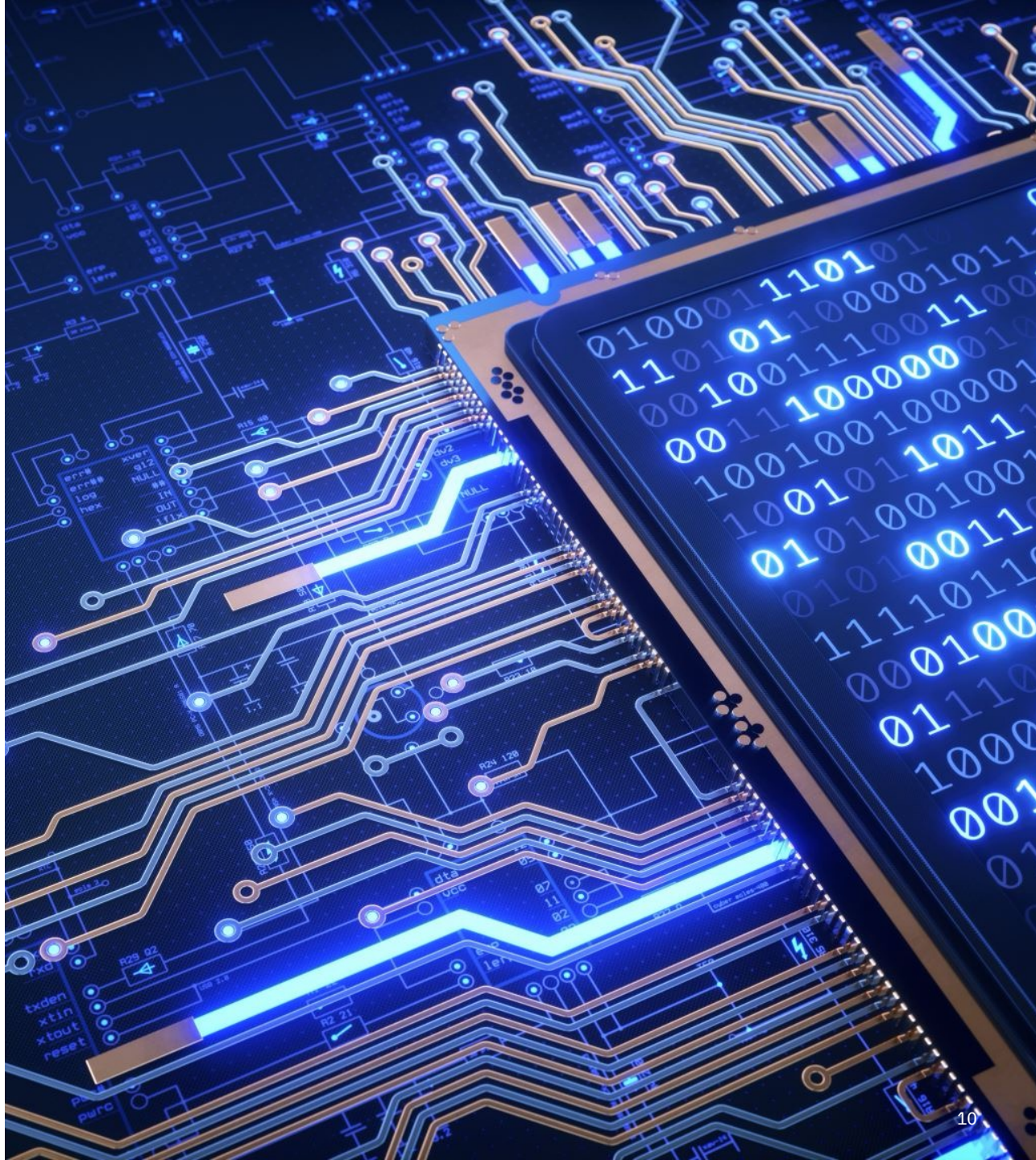
Leading to a ‘security industry’ hunting malware signatures in file systems and network traffic

But Why? Why Now?

In contrast, open source software is *available in source form*

- Source code study encouraged, to find and correct bugs
- If your open source code is good enough, 3rd party review *will* happen
- Stronger focus on *code quality* and *correctness*
- Even to prevent future, *unknown* bugs from exploitation, the *OpenBSD* project stands out in innovations in exploit *mitigation* and *prevention*.
- Inspires best practices.

But first, a bit of history.



Historically, '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

The perceptions changed over time, but in the pre-Internet age, software was either unimportant or proprietary.

But Then Suddenly Software Turned Important

The Internet happened

Two important things happened:

First, (obvious to developers),

> *Our infrastructure became dependent on open source (mainly BSD) software*

The infrastructure was built on *rough consensus and working code*

Instead of 10,000 chimps typing …

The *other* thing that happened was

- Devices and software not designed for network use
- Operated by the unwashed masses
- *… connected to the Internet.*
- Bad things started happening
- The public was not prepared for buggy software to bite
- Software security was on part of the public's lexicon at all

Some bugs may be shallow

- *"Given enough eyes, all bugs are shallow"*
  – *Linus Torvalds*

"Open source software can never be secure"
- Anonymous Pundit

Neither assertion stands up to scrutiny.

Cases in point:
Solarwinds SUNBURST (2020)
Log4j's Log4Shell incident (2021)

Both supply chain compromises

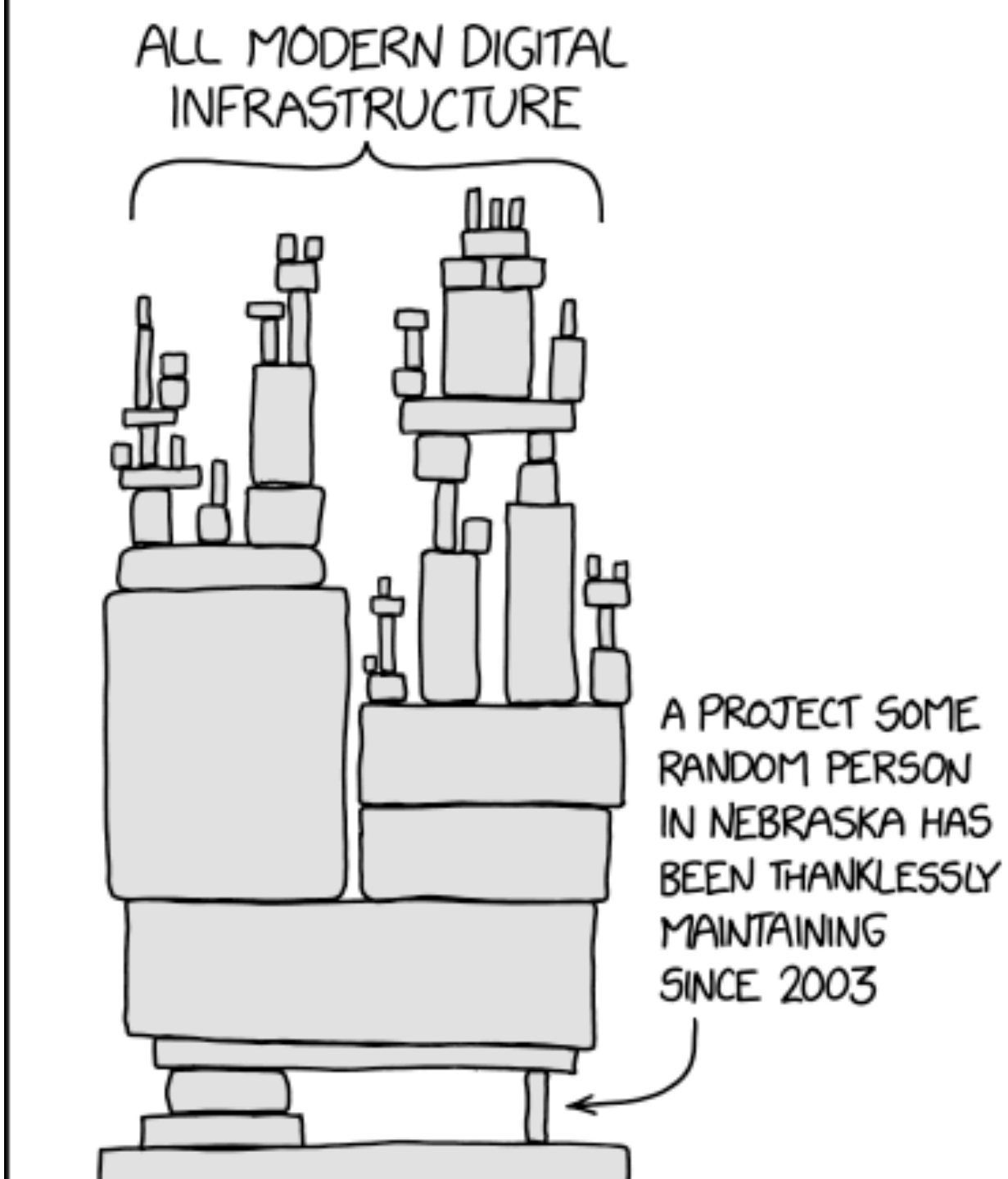
Dependencies Became A Thing

- *You may choose to trust, but you still need to verify.*

  That goes for open source and proprietary software both.

  You need to know your dependencies and actual environment.

The original meme:



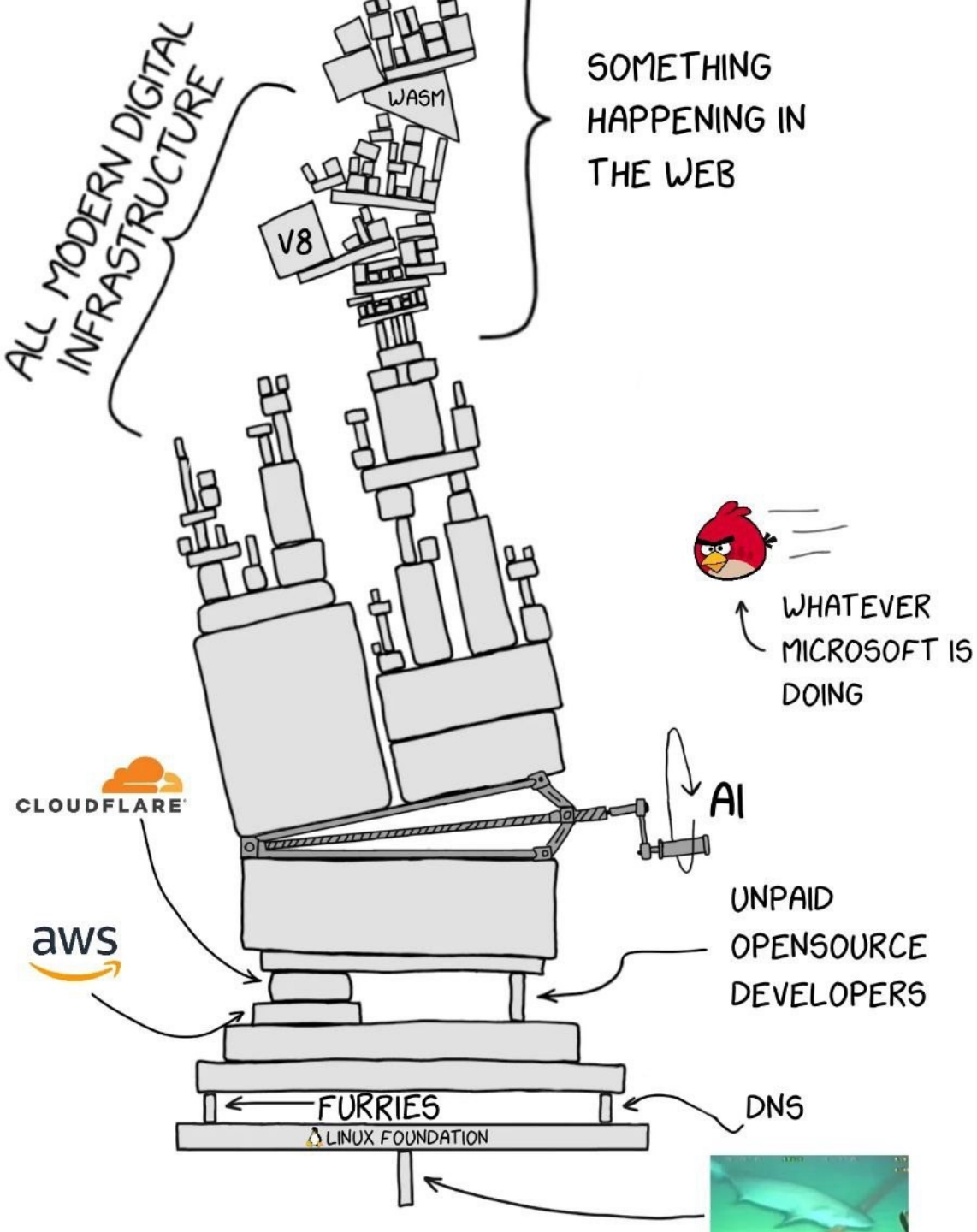
Dependencies Became A Thing

- *You may choose to trust, but you still need to verify.*

  That goes for open source and proprietary software both.

  You need to know your dependencies and actual environment.

  This one may be closer to the truth:



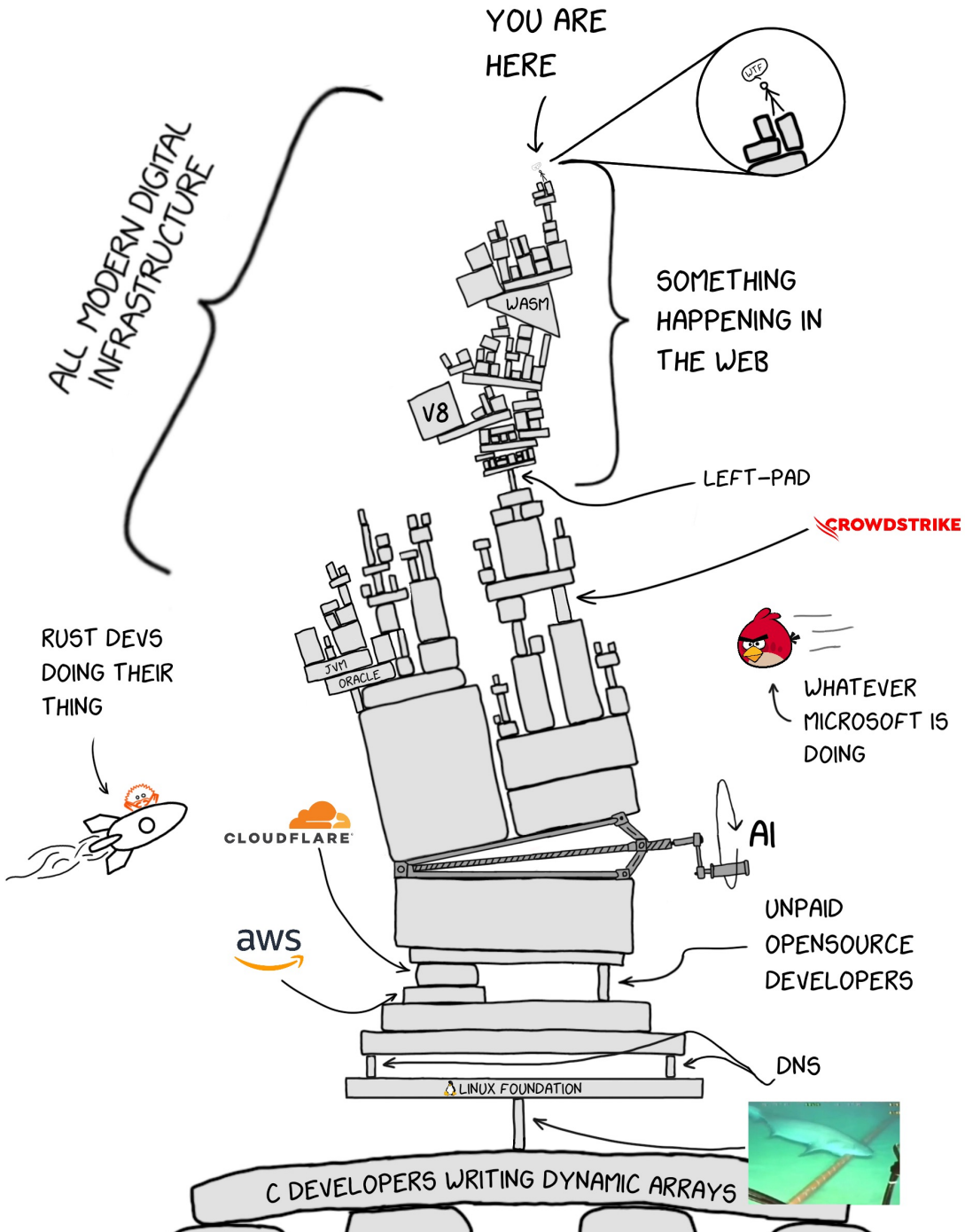
Dependencies Became A Thing

- *You may choose to trust, but you still need to verify.*

  That goes for open source and proprietary software both.

  You need to know your dependencies and actual environment.

  Your environment might look like this:



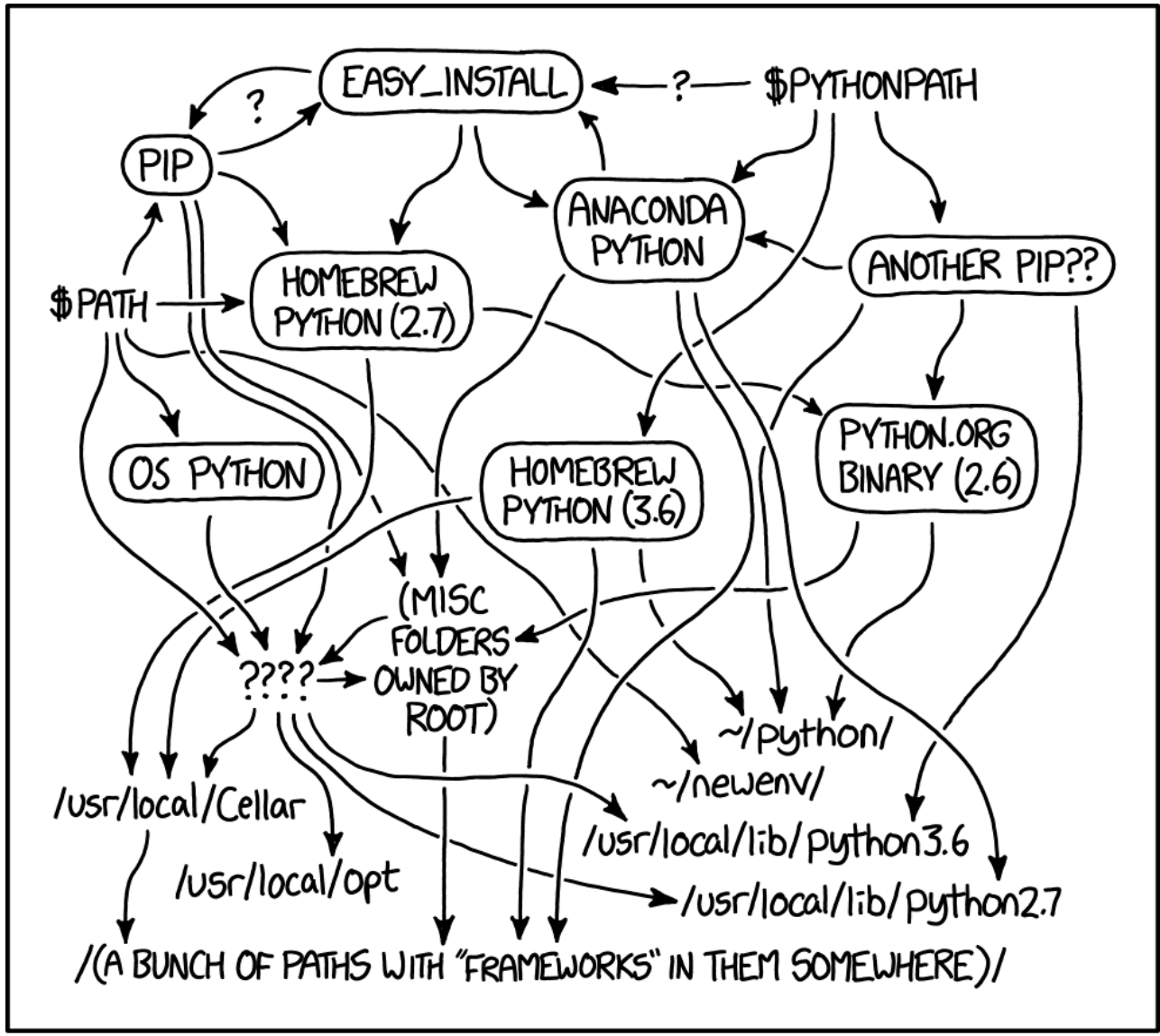
Upping Your Engineering Game

Dear developer,

- *Do you know what your code does?"*
- *"Do you know **everything** your code does?"*.

Possibly not.

You also need to watch all the things your code depends on in order to work.



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE

Upping Your Engineering Game

Summing up so far,

We write software

Which depends on other software

Which interacts with other software

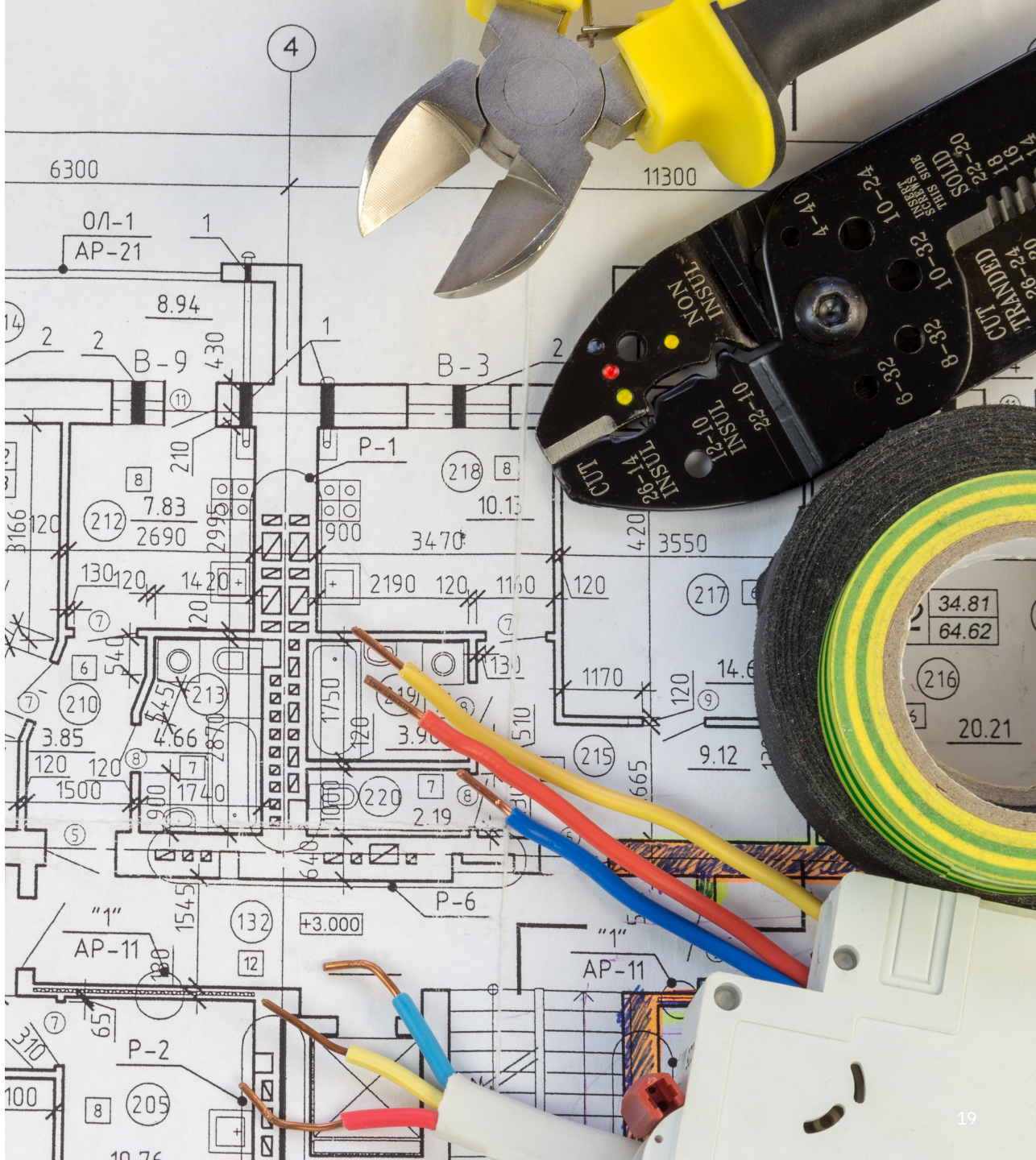
Which again interacts with other components (hardware, humans)

To run important stuff

Nothing exists in actual isolation

– *No project is an island*

What we do is important.
What do we do about that?



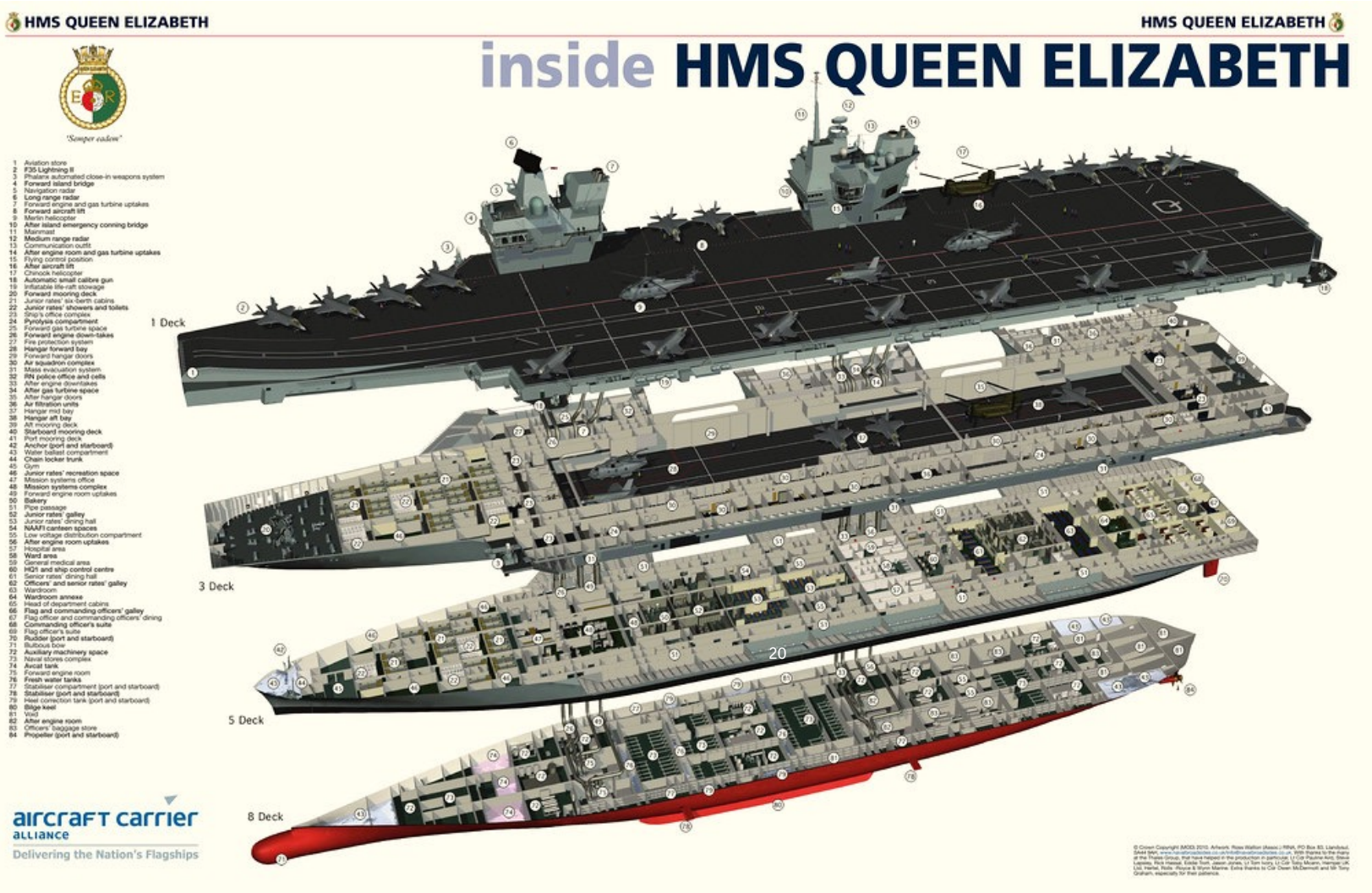
Learn from those who build important things

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

  are required to submit a

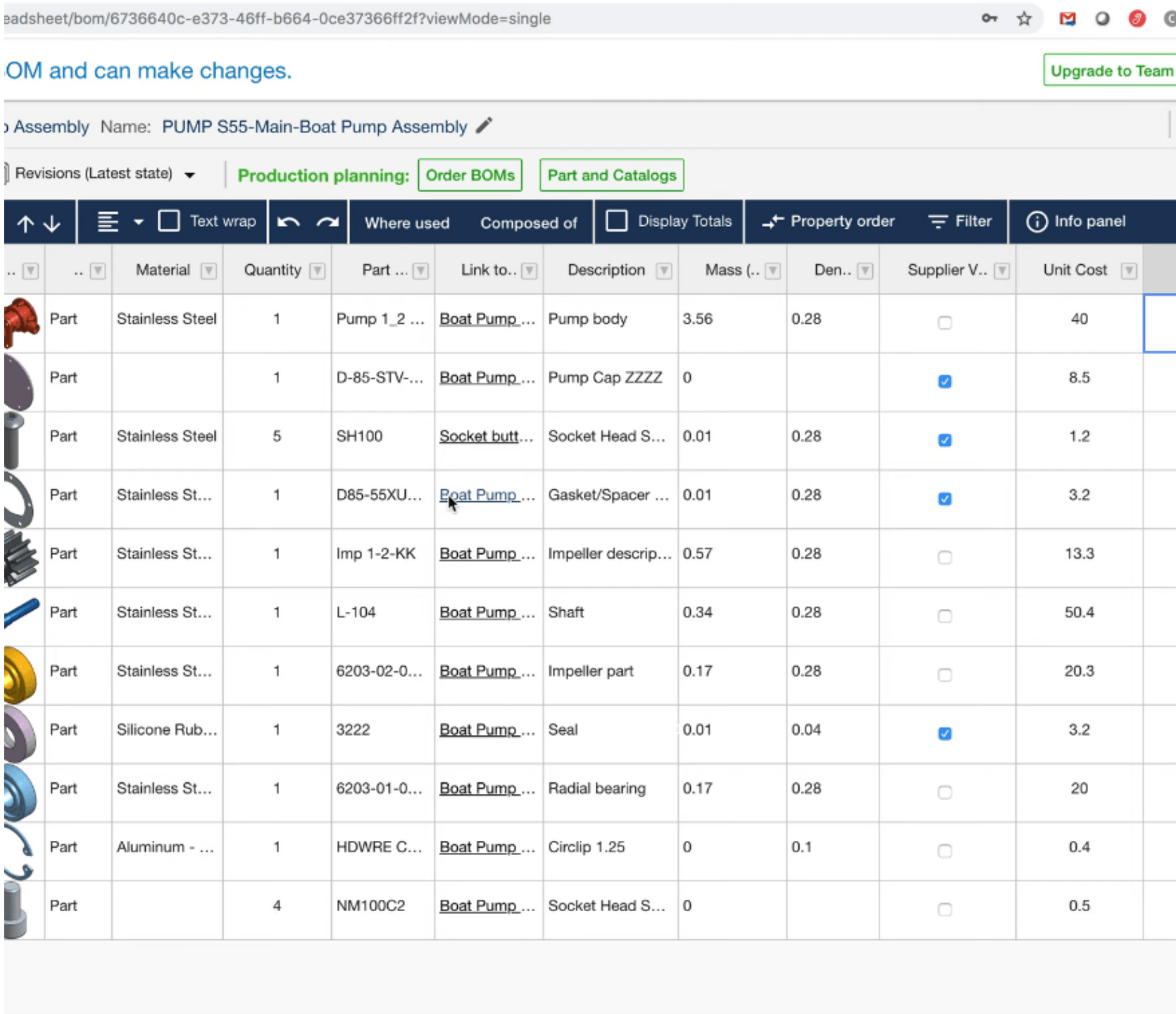
  “Bill of Materials”

  aka BOM for each delivery



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



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

* This executive order has been partially retracted, making the specifications optional for the US federal sector

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, **but** there are gaps in mapping between the two

Your New Build Artifact, the SBOM

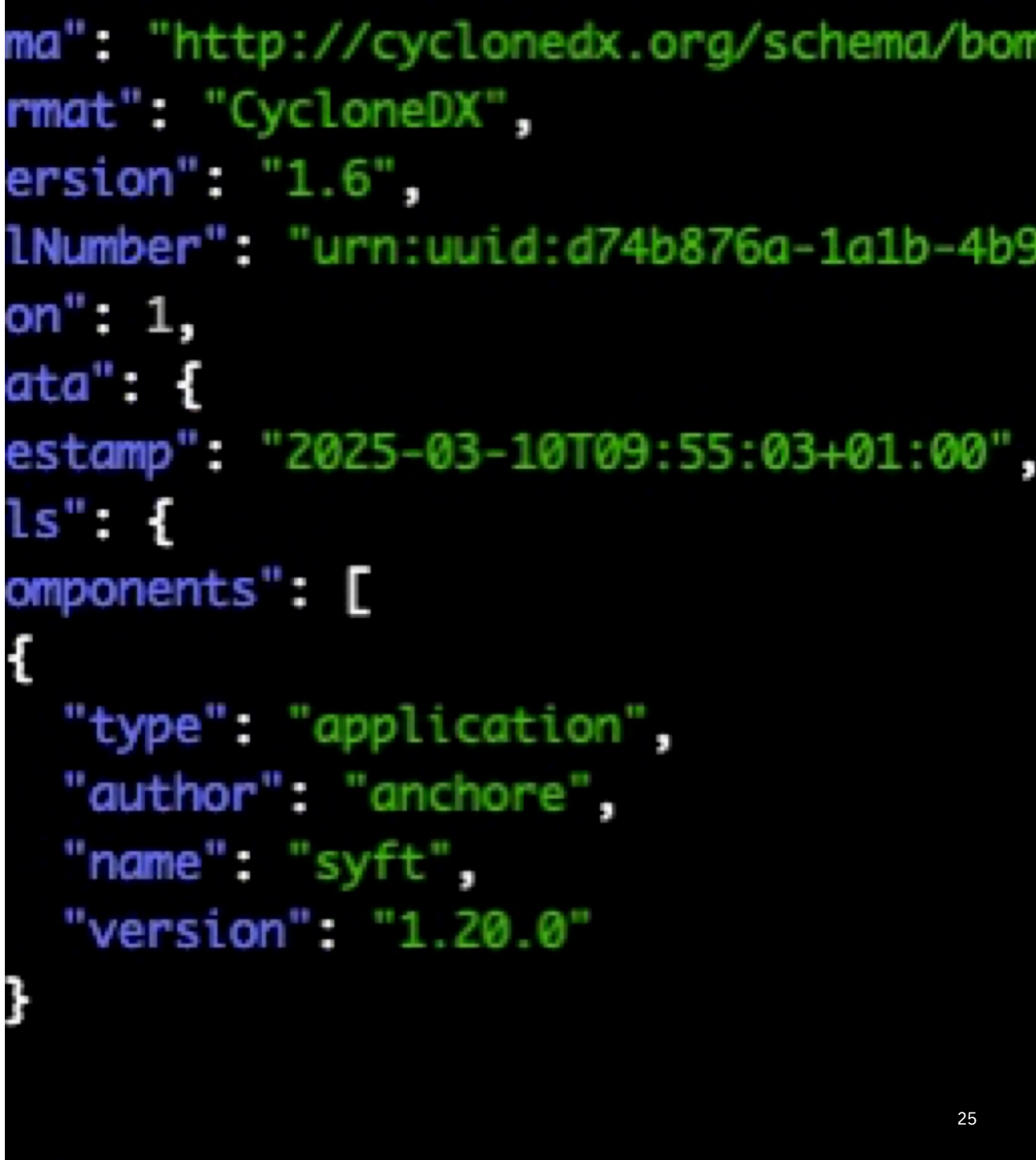
- There are numerous SBOM generating tools available, syft and cdxgen are popular choices.
- Your explorations could start with

```
$ cd myproject
$ cdxgen -t c .
```

Assuming a C language project, output to `bom.json`, viewable (pretty printed) with `jq` or similar.

This is when you discover your actual dependencies

- Add the generating as a build step, you're off to a good start



The Software Bill of Materials as a Tool for Insight and Transparency

The motivation for CRA is to ensure products are safe to use.

The SBOM is

- a tool for quality assurance
- required for certification purposes (CE mark)

SBOMs for a product with digital elements

- Need to *exist*
- Need *not* be made *public*, must be made available to regulators

For *open source*, SBOMs are tools for transparency:

=> advantage, Open Source

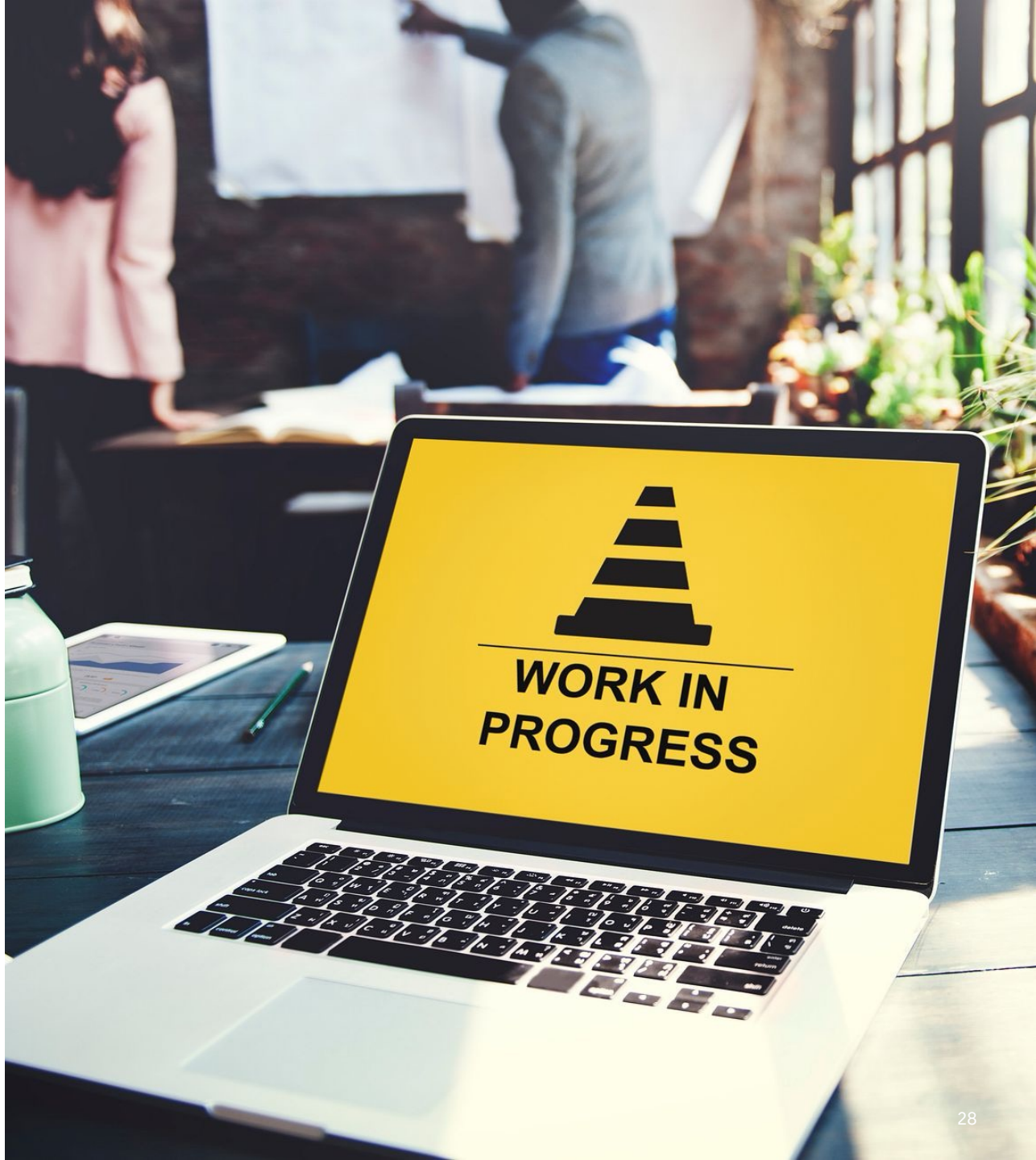
For Manufacturers and Suits, GUI Dependency Tracking Tools

- For analyzing and visualizing SBOM data, Dependency Track (https://dependencytrack.org) is the standard tool
- Designed to ingest SBOMs, ideally from CI/CD (build and deploy) process
- Available to run locally using Docker
- Suitable for cloud operation (see eg Azure Marketplace, https://marketplace.visualstudio.com/items?itemName=GSoft.dependency-track-vsts)

- Java code under a free license, (Apache 2), developed under the OWASP umbrella (https://owasp.org)

The State of Play in the BSD Projects

- The state of CRA readiness in open source projects varies.
- The Linux Foundation has sponsored tools and education work
- The FreeBSD Foundation is funding CRA readiness activities for FreeBSD
- No official public statements from NetBSD or OpenBSD (but *manufacturers* are tooling up)



`/bin` is full

- BSD projects tend to maintain all tools needed for development in the base system, self-contained
- The more popular SBOM tools are not written in C, (`syft` is **98.9%** Go, `cdxgen` is **95.7%** Javascript)
- "`/bin` is full":
  Including more languages in default install is problematic
- FreeBSD Foundation: *Getting ready for the Cyber Resilience Act (25 feb 2025)*
  - To include a modified (from NetBSD) `pkg-config`, leveraging existing package management tech to generate SBOM of FreeBSD default builds
    (so possibly not FreeBSD specific enough to be *useless* elsewhere?)
  - To appear in FreeBSD 16, ETA December 2027

Yes, the Future Is Bright If You Engineer Up

- The *European Union Cyber Resilience Act (CRA) is NOT the end of open source software*
- *The CRA is potentially extremely beneficial to well engineered free and open source software*
- *The CRA is a sign that in the European Union, software engineering is a proper branch of engineering*
- *The CRA and the SBOM requirement encourages transparency, to our advantage*
- *The CRA does NOT place any new burdens on free and open source software development*

It's time to engineer up!

Resources for Further Study and Development

- 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
- Cyber Resilience Act, official site for reference (full text of the act is here)
- European Union Cyber Resilience Act (CRA) resources page at the Open Source Security Foundation (OpenSSF)
- 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
- Awesome CRA Compliance is a curated list of CRA compliance resources
- 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!* (also *here* , *slides*)
- Peter N. M. Hansteen: *What has (can) the EU Cyber Resilience Act done (do) for you?* (this article) (also *here*)

Thank you

Peter N. M. Hansteen

June 19, 2026

https://nxdomain.no/~peter/BSD_CRA.pdf