
helvetia
KARAKUN
Kanton Basel-Stadt
ERNI
INNOFUSE
+++Ba
sel++
++Te+
+++ch
Basel
basel.ch
JAVA
USER
GROUP
CH
BLKB
optravis
company of .msg
cyon
CS2
CSS
IRIX
swiss made
software
ICT
Scouts
Wir fördern Zukunft
SD:>_
[SwissDevJobs.ch]
#BaselOne26
baselone.ch

CRA, NIS2, DORA, AI Act



What Senior Engineers ***must*** **deliver** now and *by December* **2027**

4 DEV
x DEV

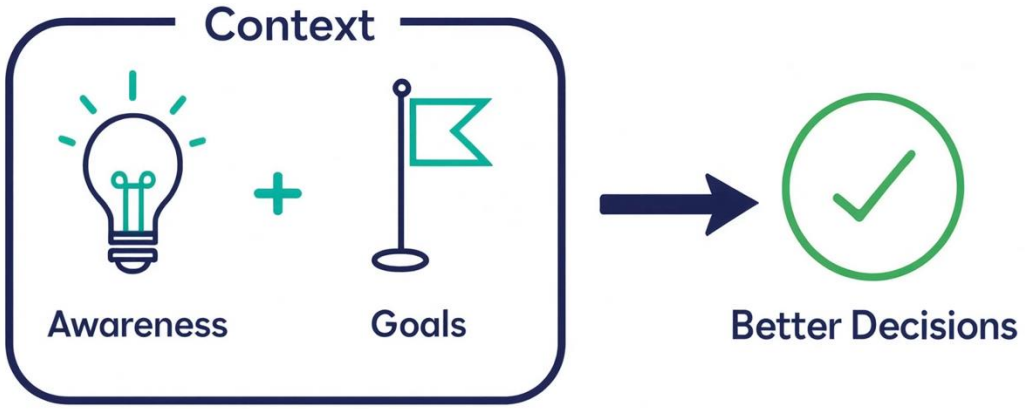
CRA, NIS2, DORA, AI Act

Cyber **R**esilience **A**ct
Network and **I**nformation **S**ecurity **D**irective **2**
Digital **O**perational **R**esilience **A**ct
Artificial **I**ntelligence **Act**

Karakun

What Senior Engineers ***must*** **deliver** now and *by December* **2027**





Why am we here?

Financial penalties and economic risks

Regulation	Type	Maximum Administrative Fines
CRA (EU 2024/2847)	Administrative fines	Up to **€15M or 2.5% global annual turnover** (higher tier)
NIS2 (EU 2022/2555)	Administrative fines	Essential entities: max or at least **€10M or 2% turnover**; Important: **€7M or 1.4%**
DORA (EU 2022/2554)	Supervisory penalties (via national authorities)	Not harmonised EU-wide; enforced via national regimes
AI Act (EU 2024/1689)	Administrative fines	Up to **€35M or 7% turnover** (highest tier), €15M / 3%, €7.5M / 1.5%
FINMA (Switzerland)	Supervisory / criminal	No fixed % fines like EU; **criminal fines under Swiss law**; enforcement via measures



Markus Schlichting

- Markus.Schlichting@karakun.com
- linkedin.com/in/markusschlichting
- @madmas@jit.social
- github.com/madmas

Open Regulatory Compliance

KARAKUN



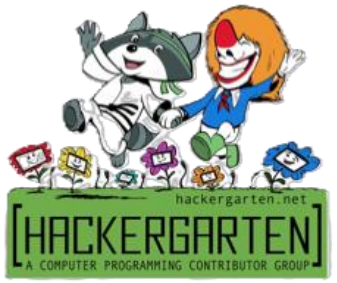
Ixchel Ruiz

- linkedin.com/in/ixchelruiz
- @ixchelruiz@mastodon.social
- github.com/ixchelruiz



karakun

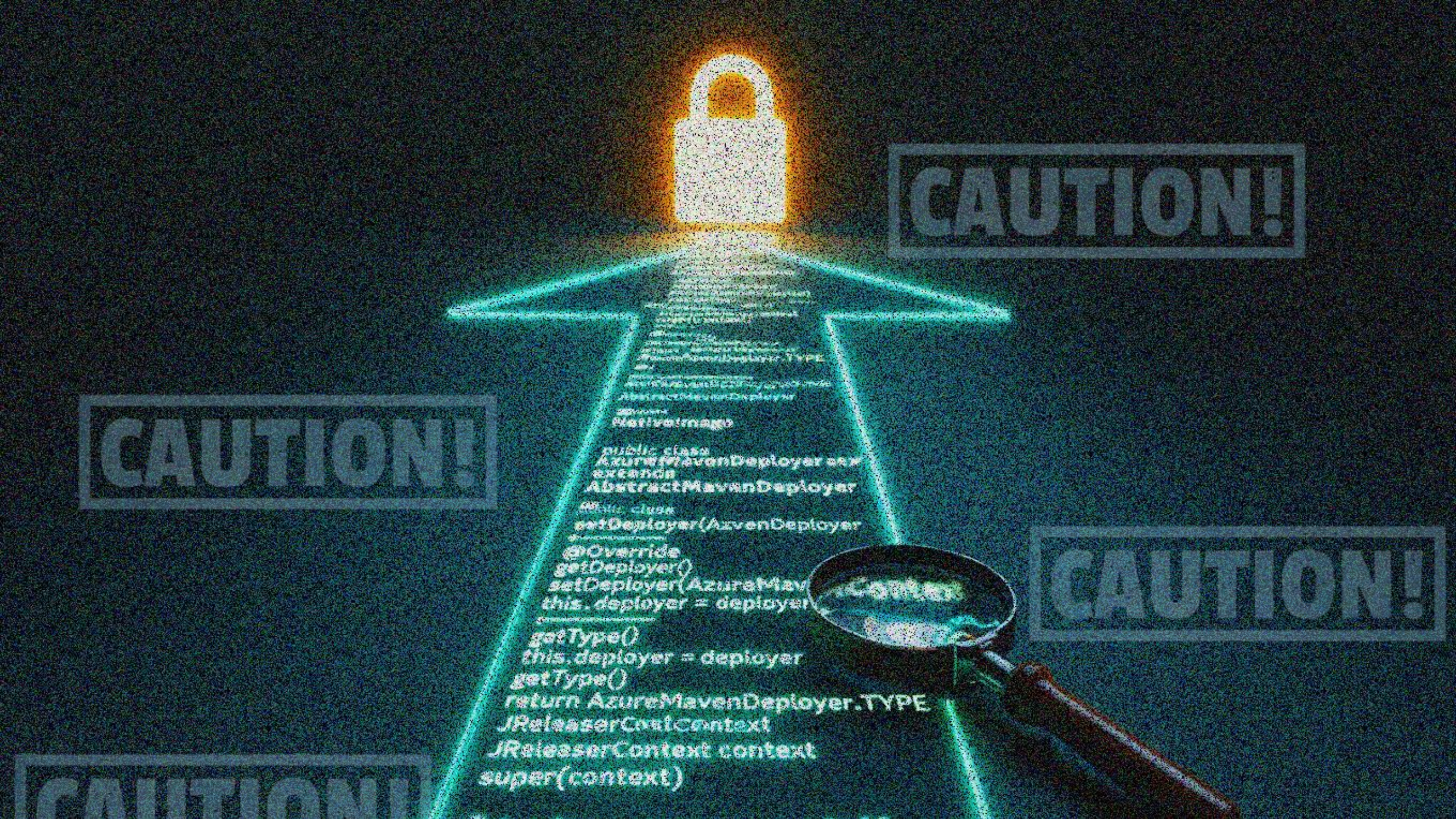
17 years:
Continuous Open Source Collaboration







- **480** community events
- **2010 → 2026**
- **47 events in 2025**
- 50+ events in 2026
- **30 upcoming events**
- **5 active communities:** Basel · Luzern · Zürich · Stuttgart · Dortmund


CAUTION!
CAUTION!
CAUTION!
CAUTION!
getType()
this.deployer = deployer
getType()
return AzureMavenDeployer.TYPE
JReleaserContext context
super(context)



Romano Roth • 1st

Helping CTOs & CIOs turn AI ambition into an operating model: feedback loo...

View my newsletter

3d • Edited •

A Swiss medical professional built their own patient management system with an AI coding agent and put it on the open internet. An engineer needed thirty minutes to get full read and write access to every patient record.

Inverter security nightmare: Hoymiles silences neighborhoods

Radio vulnerabilities in PV systems allow disabling while driving by and physical destruction of devices.



https://www.heise.de/en/news/Inverter-security-nightmare-Hoymiles-silences-neighborhoods-11357158.html

CRA — Major problems

01

The **low level of cybersecurity** of products with digital elements, reflected by widespread **vulnerabilities** and the insufficient and **inconsistent** provision of **security updates** to address them.

KARAKUN

CRA — Major problems

01

The **low level of cybersecurity** of products with digital elements, reflected by widespread **vulnerabilities** and the insufficient and **inconsistent** provision of **security updates** to address them.

02

The **insufficient** *understanding* and access to information by users, preventing them from **choosing** products with adequate cybersecurity properties or using them in a secure manner.

KARAKUN

Schweiz bekommt Cybersicherheitsgesetz

Von Mark Schröder, 25. September 2026 um 12:23

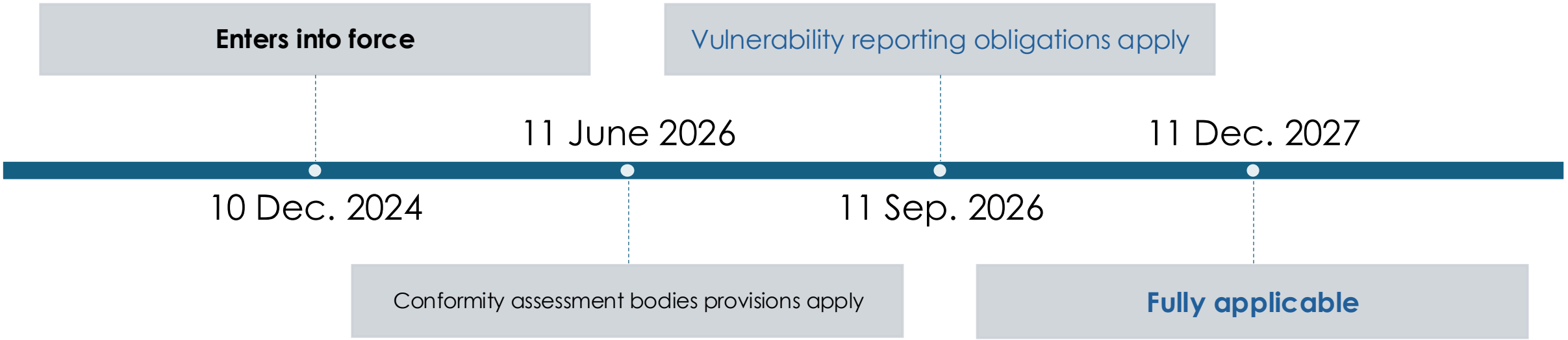
SECURITY E-GOVERNMENT BUNDESRAT GESETZ CYBERSICHERHEIT VBS SCHWEIZ BACS



Foto: Parlamentsdienste / Rob Lewis

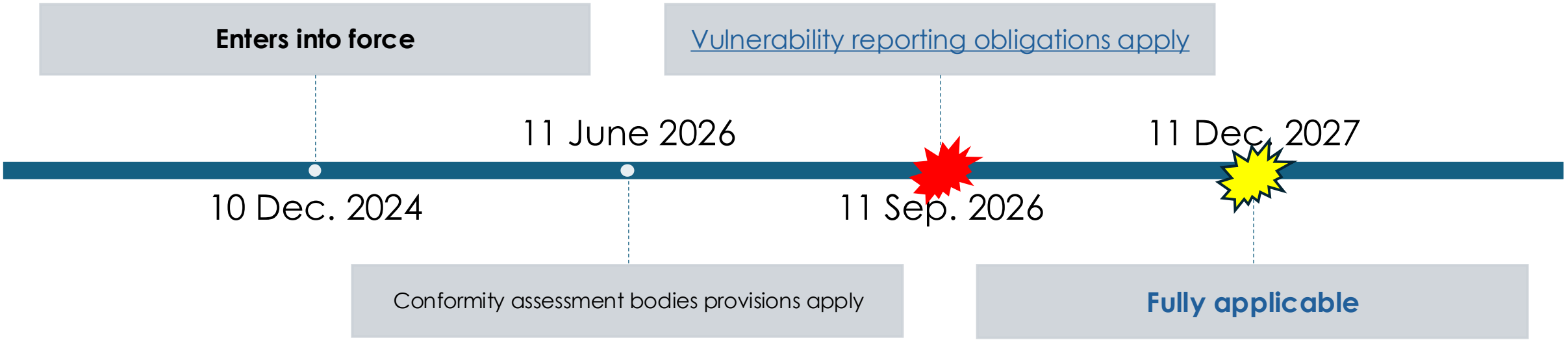
Der Bundesrat will zur Stärkung der nationalen Cybersicherheit ein eigenes Bundesgesetz schaffen. Darin sollen Anforderungen für Händler, Hersteller und Importeure festgelegt werden.

CRA — Regulation (EU) 2024/2847



KARAKUN

CRA — Regulation (EU) 2024/2847



KARAKUN

Financial penalties and economic risks

Regulation	Type	Maximum Administrative Fines	Other Economic Risks
CRA (EU 2024/2847)	Administrative fines	Up to **€15M or 2.5% global annual turnover** (higher tier)	Product withdrawal, sales ban, recalls, reputational damage
NIS2 (EU 2022/2555)	Administrative fines	Essential entities: max or at least **€10M or 2% turnover**; Important: **€7M or 1.4%**	Management liability, suspension of certification, audits, operational disruption
DORA (EU 2022/2554)	Supervisory penalties (via national authorities)	Not harmonised EU-wide; enforced via national regimes	Daily penalties, remediation costs, contractual penalties, designation as critical provider
AI Act (EU 2024/1689)	Administrative fines	Up to **€35M or 7% turnover** (highest tier), €15M / 3%, €7.5M / 1.5%	Market bans, withdrawal of AI systems, reputational damage
FINMA (Switzerland)	Supervisory / criminal	No fixed % fines like EU; **criminal fines under Swiss law**; enforcement via measures	License withdrawal, business restrictions, disgorgement of profits, personal liability

KARAKUN







America's Cyber Defense Agency

NATIONAL COORDINATOR FOR CRITICAL INFRASTRUCTURE SECURITY AND RESILIENCE

karakun

CRA — Scope

"***Regulation*** *applies* to **economic operators** *only* in relation to **products** with **digital elements** *made* **available** on the market, hence supplied for distribution or use on the Union market in the course of a commercial activity"

— CRA, Recital 15

karakun

European Commission

Economic operator

manufacturer, authorised representative, importer, distributor

Manufacturer

natural or legal *person* who develops or **manufactures** products with digital elements ... and **markets** them under its name or trademark

Authorised representative

natural or legal *person* established within the Union **acting** for a manufacturer on specified tasks.

Importer

natural or legal *person* established in the Union who **places** on the market a product from outside the Union.

Distributor

natural or legal *person* in the supply chain that **makes** the product available without affecting its properties.

Open-source software steward

legal person, other than a manufacturer providing sustained **support** for specific **FOSS** intended for **commercial activities.**

CRA —
Coverage of free and open-source software

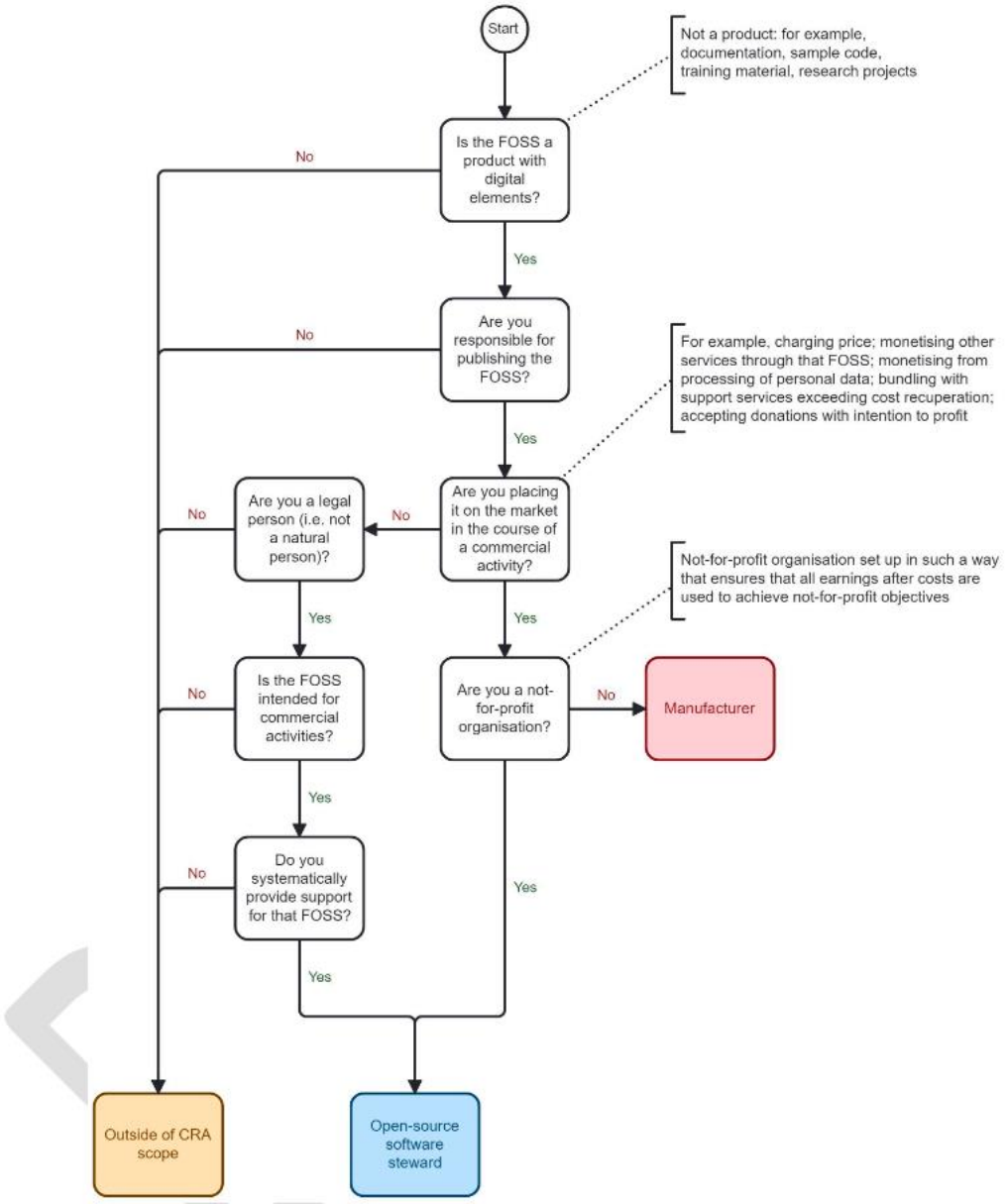


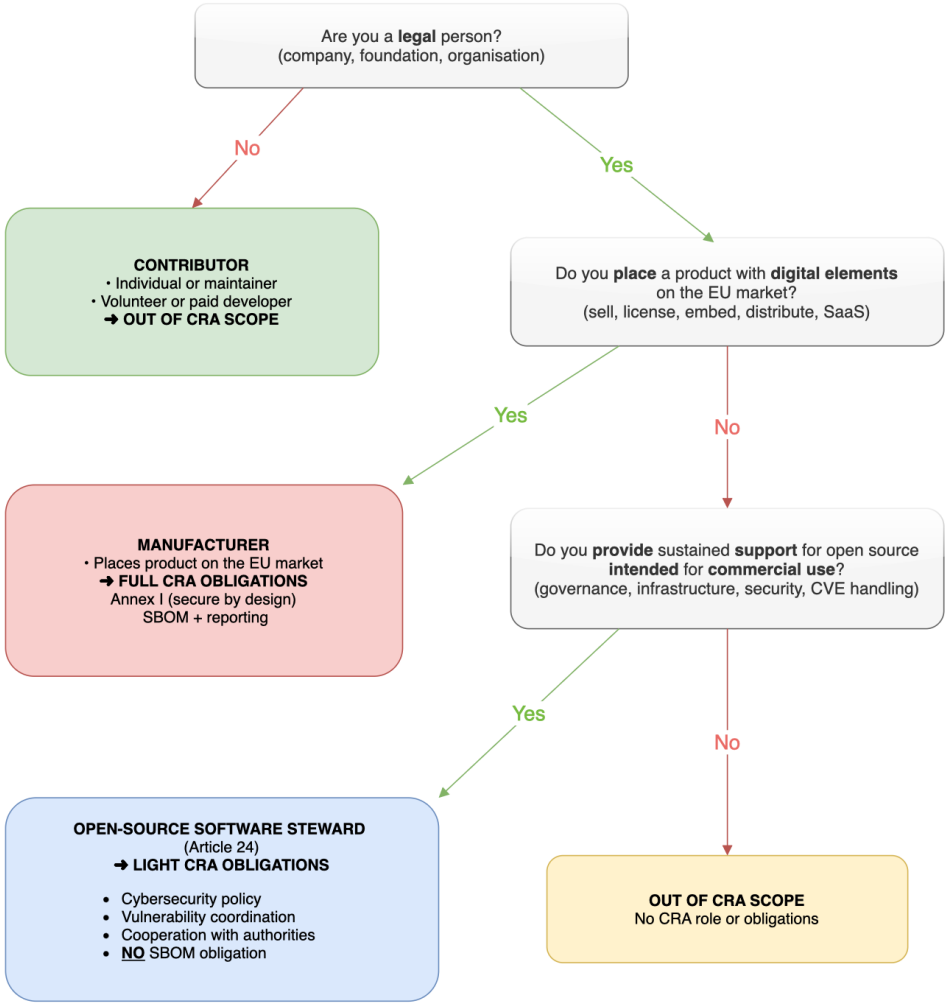
Figure 1: Stylised flowchart for CRA coverage of free and open-source software

European Commission

KARAKUN

https://digital-strategy.ec.europa.eu/en/policies/cra-open-source

CRA —
Coverage of free and open-source software



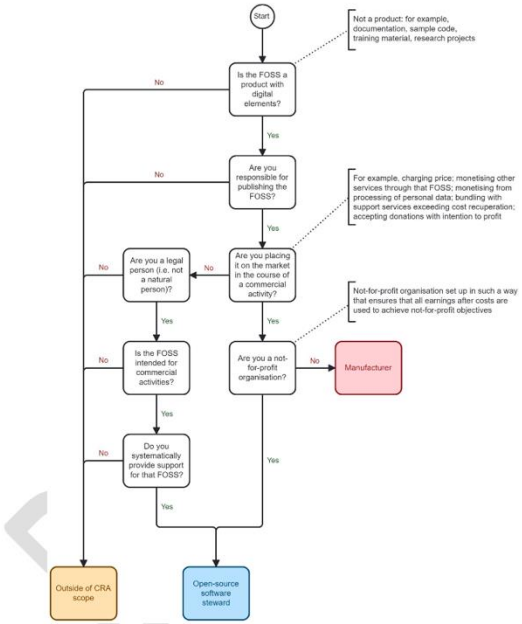


Figure 1: Stylised flowchart for CRA coverage of free and open-source software

European Commission

KARAKUN

https://digital-strategy.ec.europa.eu/en/policies/cra-open-source

In other words..

- If you **make, brand, import, distribute**, or formally steward a **commercial product** with **digital elements** in the EU, the CRA *can apply* to you.

- **Contributors** to non-commercial **FOSS** are *not the main target*; the CRA focuses on **market actors** and **stewards**, not ordinary code contribution.


EXIT

Organisations
(essential and important entities)

NIS2

Dir. (EU) 2022/2555 Arts. 2, 3, 20, 21, 23; Annex I/II

Products with digital elements

CRA

Reg. (EU) 2024/2847 Arts. 2, 3, 13, 14, 24, 31, 32; Annex I; Recital 18

Financial entities

DORA

(oversight for critical ICT third-party providers)
Reg. (EU) 2022/2554 Arts. 2, 6, 17, 19, 28

Swiss regulated financial institutions

FINMA

Circular 2023/1 Sections I-VII;
BO Arts. 12, 14e;
FinIO Arts. 12, 68

AI systems / GPAI models and operators in value chain

AI Act

Reg. (EU) 2024/1689 Arts. 2, 5, 16, 26, 50, 53

KARAKUN

Organisations (essential and important entities)

NIS2

Dir. (EU) 2022/2555 Arts. 2, 3, 20, 21, 23; Annex I/II

Legislation

Products with digital elements

CRA

Reg. (EU) 2024/2847 Arts. 2, 3, 13, 14, 24, 31, 32; Annex I; Recital 18

Legislation

Financial entities

DORA

(oversight for critical ICT third-party providers)
Reg. (EU) 2022/2554 Arts. 2, 6, 17, 19, 28

Legislation

Swiss regulated financial institutions

FINMA

Circular 2023/1 Sections I-VII;
BO Arts. 12, 14e;
FinIO Arts. 12, 68

Supervisory circular

AI systems / GPAI models and operators in value chain

AI Act

Reg. (EU) 2024/1689 Arts. 2, 5, 16, 26, 50, 53

Legislation

KARAKUN

NIS2, CRA, DORA, FINMA and AI Act

Common requirements

Network and **I**nformation **S**ecurity **D**irective **2**
Cyber **R**esilience **A**ct
Digital **O**perational **R**esilience **A**ct
Swiss **Fin**ancial **Ma**rket Supervisory Authority
Artificial **I**ntelligence **Act**

KARAKUN

	NIS2 **Organisations** (essential and important entities)	CRA **Products** with digital elements	DORA **Financial entities**	FINMA **Swiss** regulated **financial institutions**	AI Act **AI systems** / GPAI models and operators in value chain
Risk Management	Art. 21 – "risk management measures"	Art. 13 – "appropriate level of cybersecurity"	Art. 6 – ICT risk framework	Circular 2023/1 – ICT & operational risk	Art. 16 – risk management system
Governance & Accountability	Art. 20 – management oversight	Art. 13 – manufacturer responsibility	Art. 5–6 – management accountability	Board-level responsibility	Art. 16 / 26 – role-based responsibility
Visibility / Inventory	Art. 21 – assets & supply chain	Art. 13 – "identify and document components"	Art. 28 – register of ICT third parties	ICT inventory & critical data	Art. 11–12 – documentation & logs
Incident & Vulnerability Handling	Art. 23 – incident reporting	Art. 14 – vulnerability & incident reporting	Art. 17–19 – ICT incident reporting	Incident reporting obligations	Art. 62+ – post-market monitoring
Supply Chain Control	Art. 21 – supply chain security	Art. 13 – component responsibility	Art. 28 – third-party ICT risk	Outsourcing risk controls	Art. 25–28 – Value chain obligations

KARAKUN

	NIS2 **Organisations** (essential and important entities)	CRA **Products** with digital elements	DORA **Financial entities**	FINMA **Swiss** regulated **financial institutions**	AI Act **AI systems** / GPAI models and operators in value chain
Lifecycle Responsibility	Continuous risk mgmt. (Art. 21)	Art. 13 – lifecycle vulnerability handling	Continuous resilience testing	Continuous operational resilience	Post-market monitoring
Documentation & Evidence	Policies & measures (Art. 21)	Art. 13(9) – technical documentation	Documentation of ICT risk	Documentation requirements	Art. 11 – technical documentation
Monitoring & Detection	Ongoing security measures	Vulnerability monitoring	Continuous monitoring	ICT monitoring expectations	Logging & monitoring (Art. 12)
Regulatory Reporting	Art. 23 – authority reporting	Art. 14 – ENISA / authority reporting	Supervisory reporting	FINMA reporting duties	Incident reporting obligations
Resilience & Continuity	Business continuity (Art. 21)	Secure operation implied	Operational resilience testing	BCM requirements	System robustness requirements

KARAKUN

Requirement	Risk Management	Governance & Accountability	Visibility / Inventory	Incident & Vulnerability Handling	Supply Chain Control
Meaning	Identify, assess, and continuously manage risk	Responsibility is explicit and enforceable	Know what you build, run, and depend on	Detect, respond, and report incidents	You are responsible for dependencies

Requirement	Lifecycle Responsibility	Documentation & Evidence	Monitoring & Detection	Regulatory Reporting	Resilience & Continuity
Meaning	Security must be maintained over time	Must prove compliance, not just claim it	Continuous visibility of system state	External accountability to regulators	Systems must withstand and recover

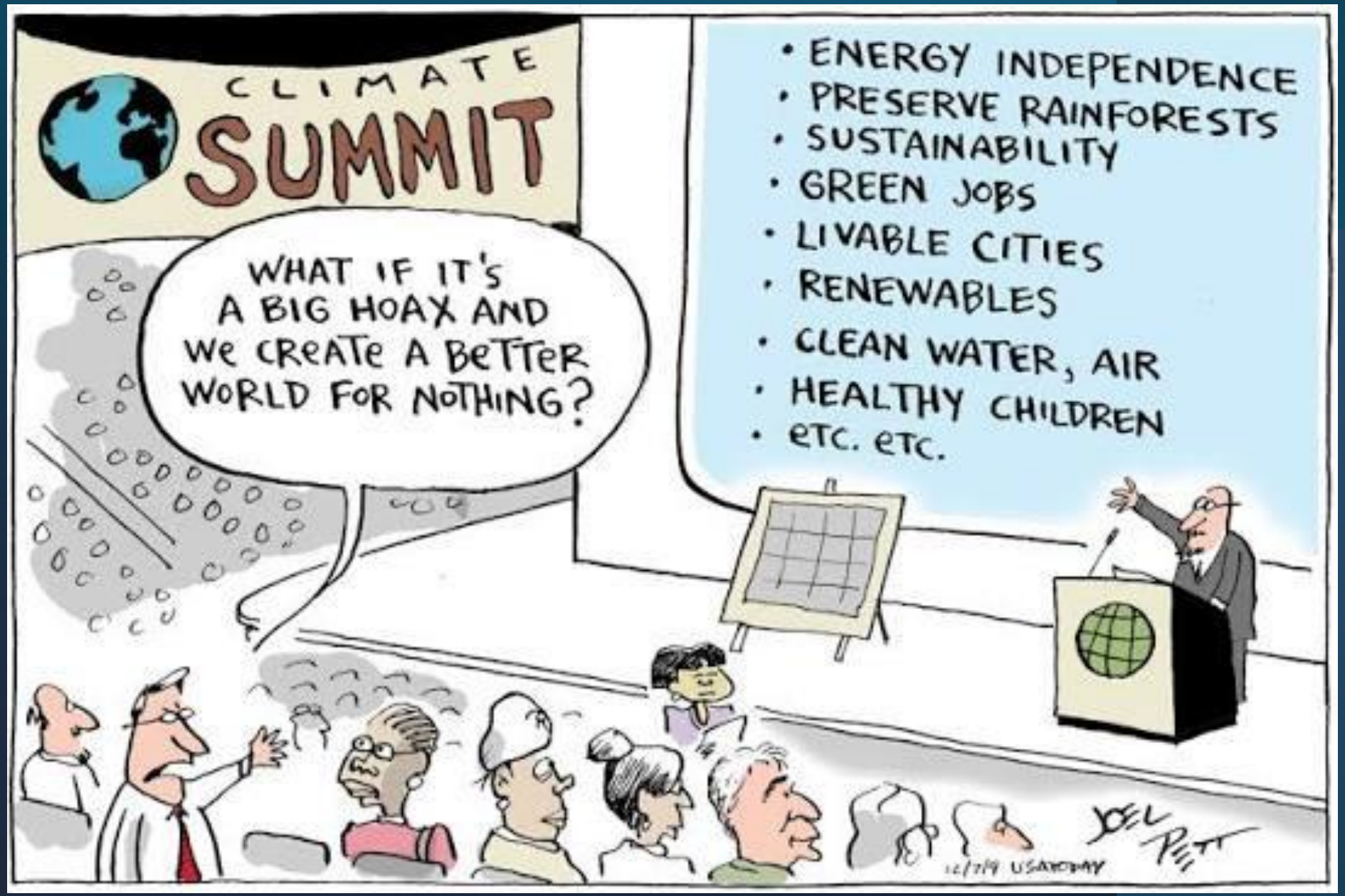
Requirement	Risk Management	Governance & Accountability	Visibility / Inventory	Incident & Vulnerability Handling	Supply Chain Control
Meaning	Identify, assess, and continuously manage risk	Responsibility is explicit and enforceable	Know what you build, run, and depend on	Detect, respond, and report incidents	You are responsible for dependencies

What is inside software?

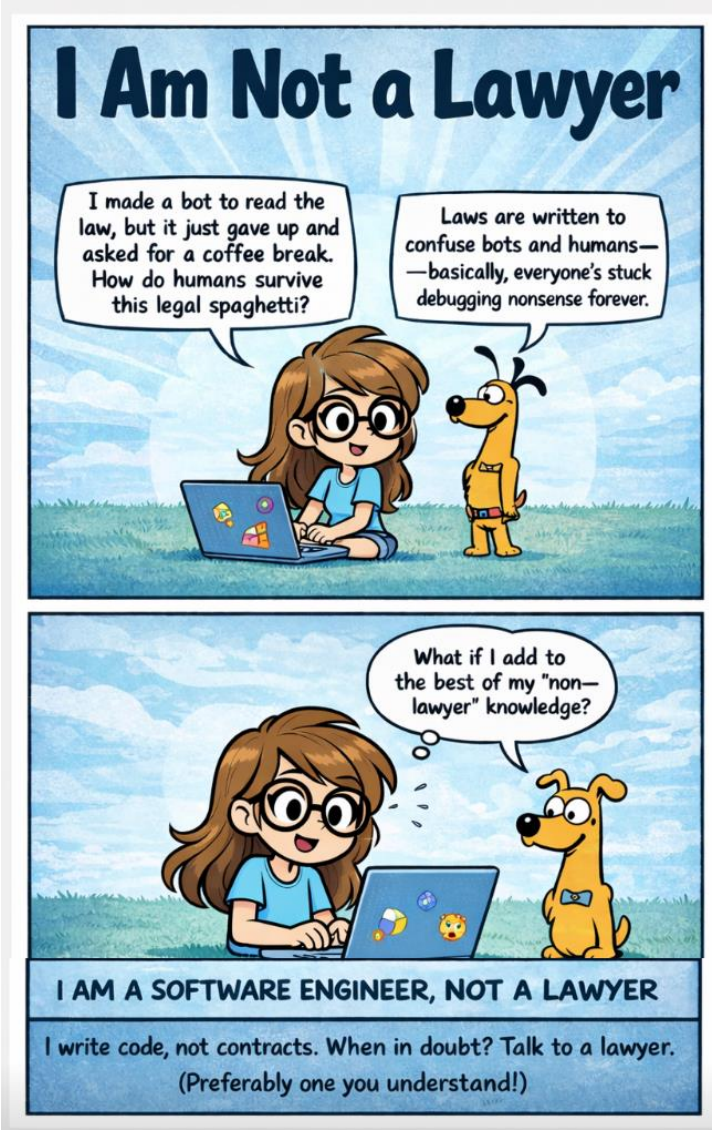
Requirement	Lifecycle Responsibility	Documentation & Evidence	Monitoring & Detection	Regulatory Reporting	Resilience & Continuity
Meaning	Security must be maintained over time	Must prove compliance, not just claim it	Continuous visibility of system state	External accountability to regulators	Systems must withstand and recover

What to do with was is inside of software?

KARAKUN



This Joel Pett cartoon for USA Today appeared just before the Copenhagen climate change conference in 2009.



BTW,

I am **not** a **lawyer**,
I am **software engineer**

Sometimes I **write** code.
Sometimes I even **read** code.

When in doubt, just ask your **trusted** lawyer! (Hope you have one)

Cyber Resilience Act (CRA)

The EU's new plan to make sure all digital products on the EU market are safe from cyber threats.

KARAKUN

CRA — Implementation

Progress so far

Monitor, follow and stay updated on the progress of new initiatives, deliverables and legislation.

28 November 2025	Implementing act on technical descriptions related to important and critical products
11 December 2025	Delegated act on CSIRTs withholding notifications to be disseminated through the Single Reporting Platform (adopted, publication pending objection period)
Early 2026	First set of Commission guidance on the Cyber Resilience Act implementation
11 June 2026	Entry into application of the provisions on the notification of Conformity Assessment Bodies. Member States to designate notifying authorities
Q3 2026	First standardisation deliverables (horizontal and product-specific standards)
11 September 2026	Entry into application of reporting obligations
Q4 2026	Delegated Act specifying the presumption of conformity for the European Cybersecurity Certification scheme on Common Critera (EUCC) with the CRA
11 December 2026	Notification of sufficient Conformity Assessment Bodies across Member States
30 October 2027	More standardisation deliverables
11 December 2027	Full application of the Cyber Resilience Act

European Commission

https://digital-strategy.ec.europa.eu/en/factpages/cyber-resilience-act-implementation

karakun

CRA — **Implementation**

Progress so far

Monitor, follow and stay updated on the progress of new initiatives, deliverables and legislation.

28 November 2025	Implementing act on technical descriptions related to important and critical products
11 December 2025	Delegated act on CSIRTs withholding notifications to be disseminated through the Single Reporting Platform (adopted, publication pending objection period)
Early 2026	First set of Commission guidance on the Cyber Resilience Act implementation
11 June 2026	Entry into application of the provisions on the notification of Conformity Assessment Bodies. Member States to designate notifying authorities
Q3 2026	First standardisation deliverables (horizontal and product-specific standards)
11 September 2026	Entry into application of reporting obligations
11 December 2026	Notification of sufficient Conformity Assessment Bodies across Member States
30 October 2027	More standardisation deliverables
11 December 2027	Full application of the Cyber Resilience Act

Monitor, follow and stay updated on the progress of new initiatives, deliverables and legislation.

European Commission

https://digital-strategy.ec.europa.eu/en/factpages/cyber-resilience-act-implementation

karakun

CRA — Core Requirements

"Manufacturers shall ensure that products with **digital elements** are designed, developed and produced in such a way that they **ensure** an *appropriate* level of **cybersecurity**."

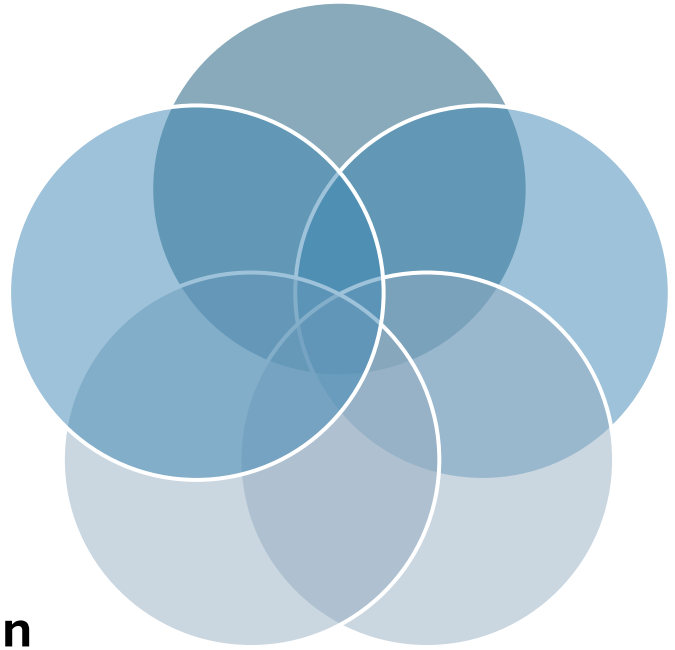
— CRA, Article 13(1)

KARAKUN

CRA — Core Requirements

Secure by **design** and by **default**

Lifecycle responsibility



Vulnerability handling and **remediation**

Technical **documentation** and **traceability**

Reporting obligations

KARAKUN

CRA — **Secure** by **design** and by **default**

1

Products with **digital elements** shall be made available on the market **without** known *exploitable* **vulnerabilities**.

— CRA, Annex I, Part I

KARAKUN

CRA — **Secure** by **design** and by **default**

1

Products with **digital elements** shall be made available on the market **without** known *exploitable* **vulnerabilities**.

— CRA, Annex I, Part I

No shipping with **known exploitable vulnerabilities**

Security must be part of **architecture**, **not a patch** later

KARAKUN

CRA — Vulnerability handling and remediation

2

Manufacturers shall **identify** and **document vulnerabilities** and **components** contained in products with digital elements.

— CRA, Annex I Part II

Manufacturers shall address and **remediate vulnerabilities** without delay.

— CRA, Annex I Part II

CRA — Vulnerability handling and remediation

Manufacturers shall **identify** and **document vulnerabilities** and **components** contained in products with digital elements.

You must **know what is inside your software**

— CRA, Article 13(3)

Manufacturers shall address and **remediate vulnerabilities** without delay.

You must **track and fix vulnerabilities continuously**

— CRA, Article 13(3)

KARAKUN

CRA — Reporting obligations

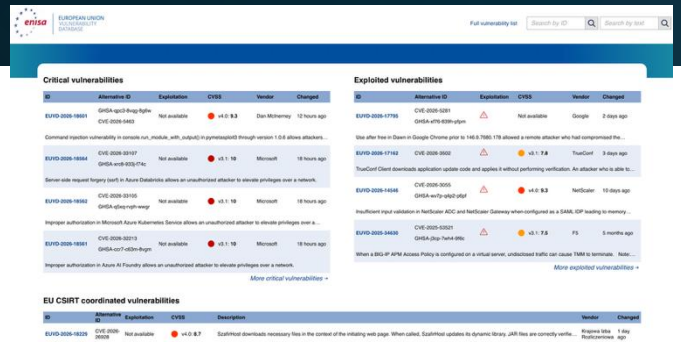
3

Manufacturers shall **report** *actively exploited* **vulnerabilities** and ***severe*** *incidents*.

— CRA, Article 14

KARAKUN

CRA — **Reporting** obligations



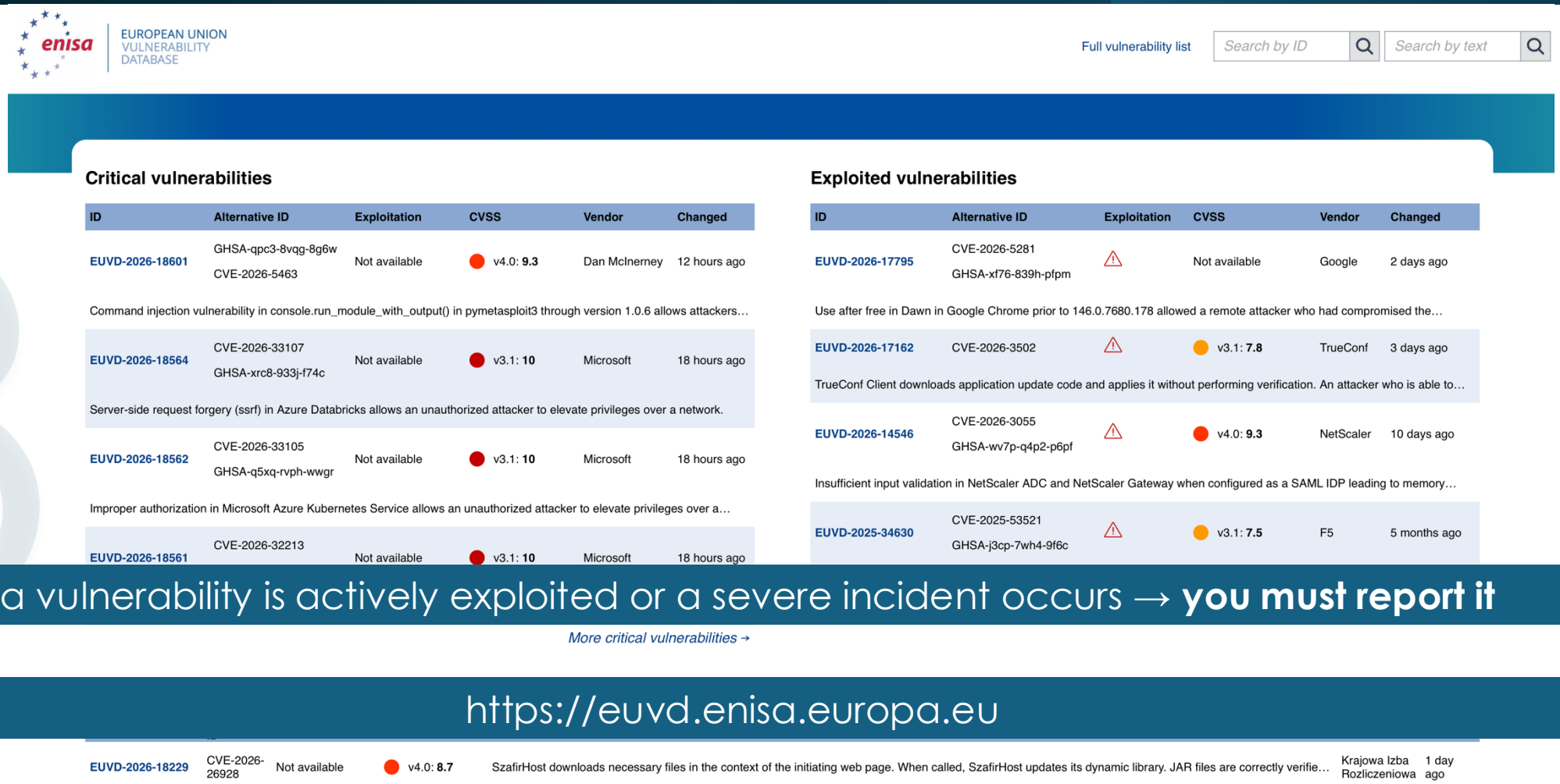
Manufacturers shall **notify** *actively exploited* **vulnerabilities** and ***severe*** *incidents*.

— CRA, Article 14

If a vulnerability is actively exploited or a severe incident occurs → **you must report it**

KARAKUN

CRA — **Reporting** obligations

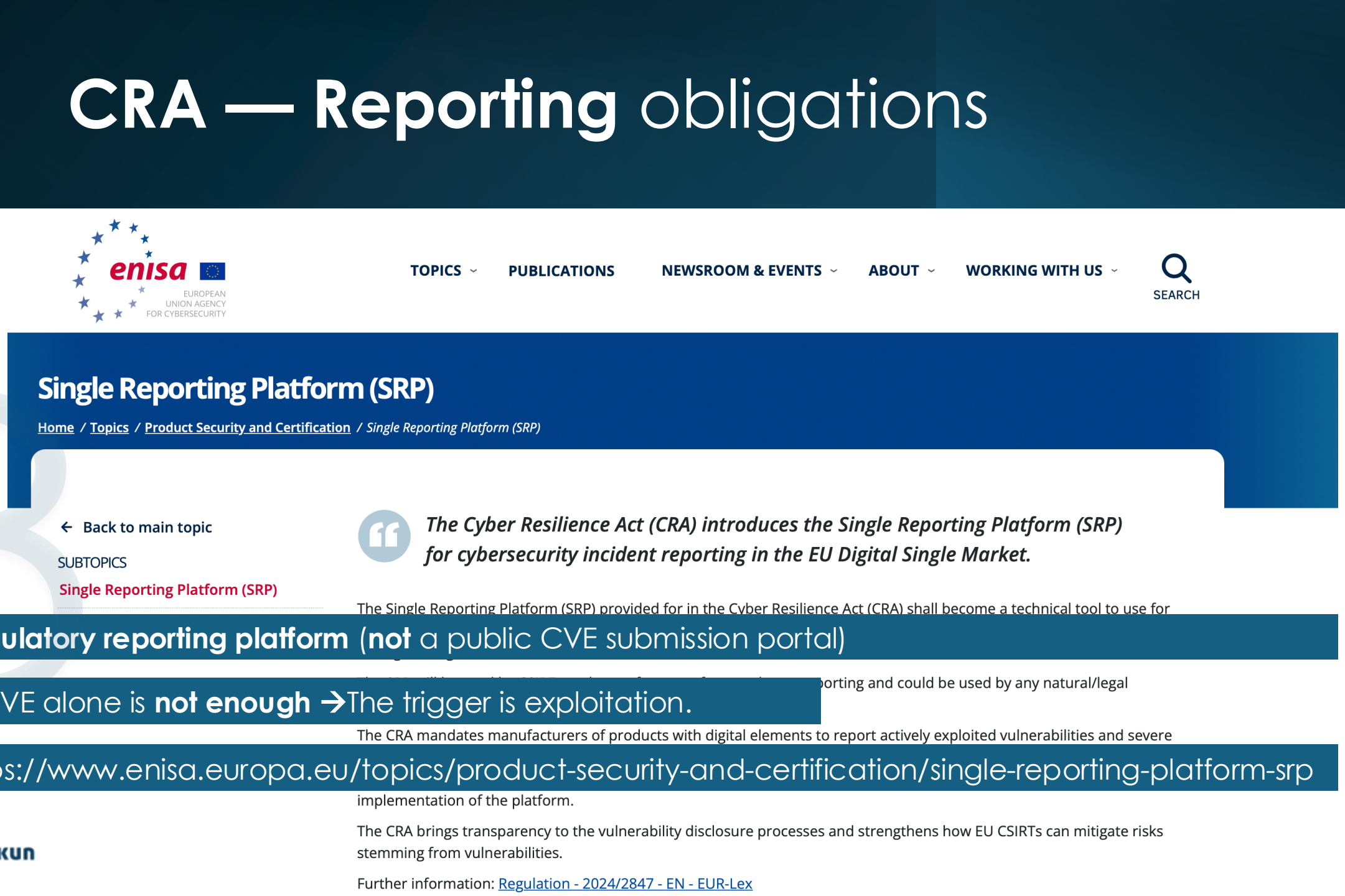


If a vulnerability is actively exploited or a severe incident occurs → **you must report it**

https://euvd.enisa.europa.eu

KARAKUN

CRA — **Reporting** obligations



Regulatory reporting platform (**not** a public CVE submission portal)

A CVE alone is **not enough** →The trigger is exploitation.

https://www.enisa.europa.eu/topics/product-security-and-certification/single-reporting-platform-srp

KARAKUN

First mandatory **obligation** **(11 September 2026)**



Every actively exploited **vulnerability**



Every reportable **severe incident**

This means organisations must already have:

- **vulnerability detection**
- incident triage
- legal review
- internal escalation
- **and reporting procedures**

must be submitted through the SRP

KARAKUN

What must be reported

Actively exploited vulnerability

Report if **all** apply:

- affects an in-scope product
- reliable evidence of unauthorised malicious exploitation
- manufacturer became aware on/after 11 Sep 2026

Severe incident Report if it affects product security and **either**:

- harms, availability, authenticity, integrity or confidentiality of important/sensitive data or functions
- introduces, malicious code into the product or a user's network/information systems

The CRA does **not** generally apply **retroactively**.

Products with digital elements placed on the EU market before 11 December 2027 are subject to the CRA **only** if they undergo a **substantial modification** after that date.

The **exception** is Article 14: **vulnerability and severe incident reporting obligations** apply to all in-scope products, including those **already placed** on the market before 11 December 2027.

— Regulation (EU) 2024/2847, Article 69(2)–(3)

CRA — Technical **documentation** and **traceability**

4

Manufacturers shall draw up the **technical documentation**.

— CRA, Article 13(12) and Article 31

KARAKUN

CRA — Technical **documentation** and **traceability**

4

Manufacturers shall draw up the **technical documentation**.

— CRA, Article 13(12) and Article 31

You must be **able to explain your product**

- **security measures**
- **components**
- **architecture**

KARAKUN

CRA — **Lifecycle** responsibility

5

Manufacturers shall ensure that **vulnerabilities** are *handled effectively* for the expected product **lifetime**.

— CRA, Article 13(8)

CRA — Lifecycle responsibility

5

Manufacturers shall ensure that **vulnerabilities** are *handled effectively* for the expected product **lifetime**.

Responsibility does not end at release

— CRA, Article 13(8)

KARAKUN

CRA — Core Requirements

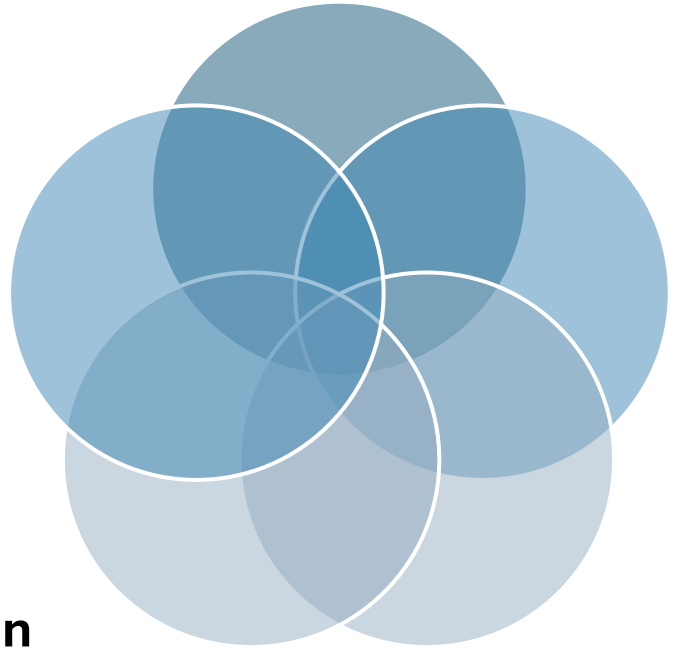
Secure by **design** and by **default**

Vulnerability handling and **remediation**

Reporting obligations

Technical **documentation** and **traceability**

Lifecycle responsibility



KARAKUN

SBOM

- Risk Management
- Visibility / Inventory
- Incident & Vulnerability Handling
- Supply Chain Control
- Documentation & Evidence



KARAKUN

SBOM — Software bill of materials

"A formal, machine-readable **inventory** documenting all **components**, **libraries**, and **dependencies** within a software product"

Captures component origins, versions, and relationships, creating an authoritative record

KARAKUN

SBOM — Standard **Formats**

CycloneDX	Lightweight OWASP standard for security, vulnerability tracking, and supply chain analysis, popular in DevOps.
SPDX	Linux Foundation's ISO standard for software license, copyright, and component compliance.
SWID Tags	Used for software asset management and tracking in enterprises.

	CycloneDX	SPDX	SWID
Focus	Security	Legal	Inventory
Timeframe	Build → Runtime	Build → Distribution	Installed state
Audience	DevSecOps	Legal / compliance	IT operations

KARAKUN

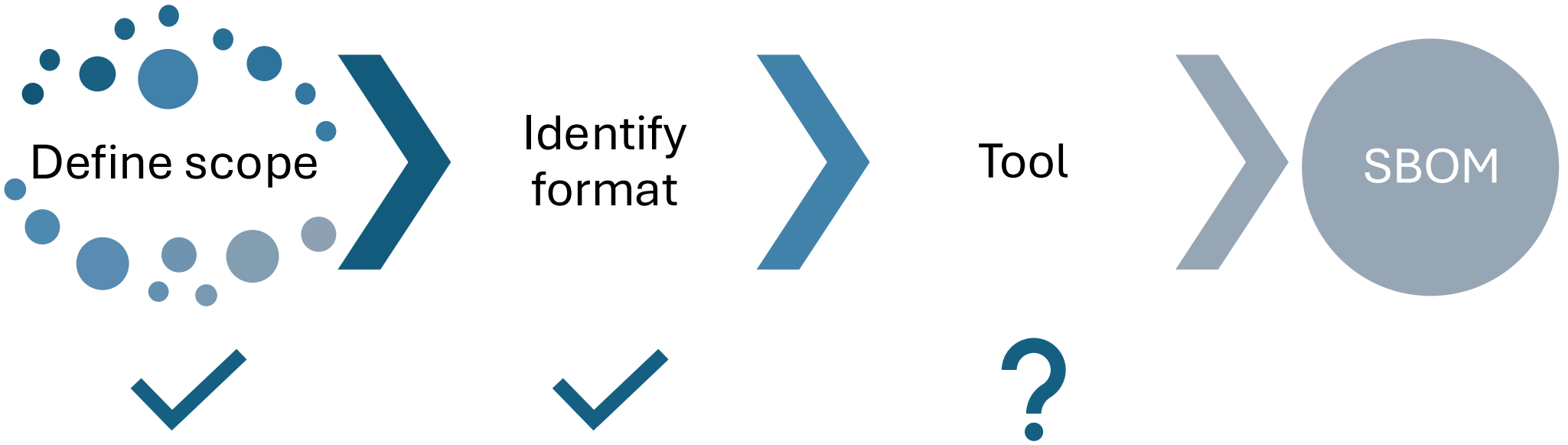
	CycloneDX	SPDX	SWID
What is inside software?	Components & dependencies	Packages & files	Limited
What vulnerabilities affect it?	Native	Indirect	No
What licenses apply?	Good	Best-in-class	Minimal
What is installed on a system?	Partial	Partial	Best
How components relate?	Dependency graph	Relationship graph	Minimal

SBOM — Standard **Formats**

CycloneDX *answers* ***"what is vulnerable?"****,* ***SPDX*** *answers* ***"what is permitted?"****, and* ***SWID*** *answers* ***"what is installed?"****.*

No single SBOM standard *solves all problems* — mature organisations often use CycloneDX for ***security***, SPDX for ***compliance***, and SWID for ***operational inventory***.

SBOM — Software bill of materials



KARAKUN

SBOM — Programming Languages

Tools?

LANGUAGE / FRAMEWORK	PRIMARY SOURCES	KEY CONFIGURATION FILES	DETECTION METHOD	TRANSITIVE HANDLING	TOOL NAMES EXAMPLES
JavaScript/Node.js	Package registries	Package manifests, lock files	Dependency listing commands	Lock files capture full tree	npm audit, yarn audit, CycloneDX Node Module, Retire.js, Snyk CLI
Python	Package indices	Requirements files, lock files	Package freeze commands	Dependency tree tools	pip-audit, Safety, cyclonedx-py, pip-licenses, pipdeptree
Java/JVM	Central repositories	Build configuration files	Dependency tree analysis	Build tools resolve transitives	OWASP Dependency-Check, CycloneDX Maven/Gradle plugins, jdeps, Maven Dependency Plugin
Go	Module registries	Module files, sum files	Module listing commands	Sum files include all dependencies	go list -m all, nancy, gosec, go mod graph, cyclonedx-gomod
.NET/C#	Package repositories	Project files, package configs	Package listing with transitives	Built-in transitive support	dotnet list package --vulnerable, CycloneDX .NET, NuGet Package Explorer, Component Detection
Ruby	Gem repositories	Gem specifications, lock files	Bundle analysis	Package manager handles dependencies	bundler-audit, bundle-audit, cyclonedx-ruby, license_finder
PHP	Package repositories	Composer files, lock files	Dependency analysis	Package manager handles transitives	composer show, composer audit, symfony security:check, cyclonedx-php-composer
Rust	Package registries	Manifest files, lock files	Dependency tree commands	Lock files essential	cargo-audit, cargo-sbom, cargo-deny, cargo-tree, cargo-cyclonedx
Container Builds	Multi-layer package databases, Base image manifests, Runtime dependencies	Dockerfile, container manifests, layer configuration files	Layer-by-layer filesystem analysis, Package database extraction, Base image	Multi-stage dependency chains across layers	Syft, Trivy, Docker Scout, Tern, container-diff, cosign

https://www.enisa.europa.eu/sites/default/files/2025-12/SBOM%20Analysis%20-%20Towards%20an%20Implementation%20Guide_v1.20-Published.pdf

Table 3 - SBOM Generation across diverse Programming Languages

SBOM — Programming Languages

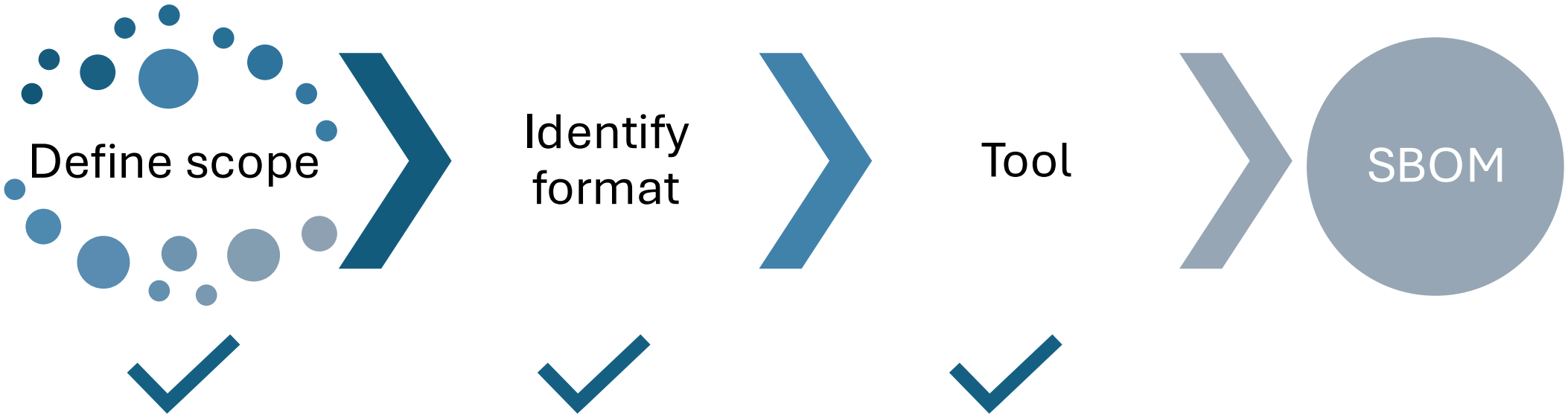
Tools?

LANGUAGE / FRAMEWORK	PRIMARY SOURCES	KEY CONFIGURATION FILES	DETECTION METHOD	TRANSITIVE HANDLING	TOOL NAMES EXAMPLES
JavaScript/Node.js	Package registries	Package manifests, lock files	Dependency listing commands	Lock files capture full tree	npm audit, yarn audit, CycloneDX Node Module, Retire.js, Snyk CLI
Python	Package indices	Requirements files, lock files	Package freeze commands	Dependency tree tools	pip-audit, Safety, cyclonedx-py, pip-licenses, pipdeptree
Java/JVM	Central repositories	Build configuration files	Dependency tree analysis	Build tools resolve transitives	OWASP Dependency-Check, CycloneDX Maven/Gradle plugins, jdeps, Maven Dependency Plugin
Go	Module registries	Module files, sum files	Module listing commands	Sum files include all dependencies	go list -m all, nancy, gosec, go mod graph, cyclonedx-gomod
.NET/C#	Package repositories	Project files, package configs	Package listing with transitives	Built-in transitive support	dotnet list package --vulnerable, CycloneDX .NET, NuGet Package Explorer, Component Detection
Ruby	Gem repositories	Gem specifications, lock files	Bundle analysis	Package manager handles dependencies	bundler-audit, bundle-audit, cyclonedx-ruby, license_finder
PHP	Package repositories	Composer files, lock files	Dependency analysis	Package manager handles transitives	composer show, composer audit, symfony security:check, cyclonedx-php-composer
Rust	Package registries	Manifest files, lock files	Dependency tree commands	Lock files essential	cargo-audit, cargo-sbom, cargo-deny, cargo-tree, cargo-cyclonedx
Container Builds	Multi-layer package databases, Base image manifests, Runtime dependencies	Dockerfile, container manifests, layer configuration files	Layer-by-layer filesystem analysis, Package database extraction, Base image	Multi-stage dependency chains across layers	Syft, Trivy, Docker Scout, Tern, container-diff, cosign

https://www.enisa.europa.eu/sites/default/files/2025-12/SBOM%20Analysis%20-%20Towards%20an%20Implementation%20Guide_v1.20-Published.pdf

Table 3 - SBOM Generation across diverse Programming Languages

SBOM — Software bill of materials



KARAKUN

SBOM — Taxonomy

Types?

Table 1: SBOM Type Definition and Composition

SBOM Type	Definition	Data Description
Design	SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.	Typically derived from a design specification, RFP, or initial concept.
Source	SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.	Typically generated from software composition analysis (SCA) tooling, with manual clarifications.
Build	SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.	Typically generated as part of a build process. May consist of integrated intermediate Build and Source SBOMs for a final release artifact SBOM.
Analyzed	SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a "3rd party" SBOM.	Typically generated through analysis of artifacts by 3rd party tooling.
Deployed	SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.	Typically generated by recording the SBOMs and configuration information of artifacts that have been installed on systems.
Runtime	SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an "Instrumented" or "Dynamic" SBOM.	Typically generated from tooling interacting with a system to record the artifacts present in a running environment and/or that have been executed.

https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

SBOM — Taxonomy

Table 1: SBOM Type Definition and Composition

SBOM Type	Definition	Data Description
Design	SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.	Typically derived from a design specification, RFP, or initial concept.
Source	SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.	Typically generated from software composition analysis (SCA) tooling, with manual clarifications.
Build	SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.	Typically generated as part of a build process. May consist of integrated intermediate Build and Source SBOMs for a final release artifact SBOM.
Analyzed	SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a "3rd party" SBOM.	Typically generated through analysis of artifacts by 3rd party tooling.
Deployed	SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.	Typically generated by recording the SBOMs and configuration information of artifacts that have been installed on systems.
Runtime	SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an "Instrumented" or "Dynamic" SBOM.	Typically generated from tooling interacting with a system to record the artifacts present in a running environment and/or that have been executed.



KARAKUN

https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

SBOM —
Benefits and Limitations of SBOM Types

Pros/Cons?

Table 2: Understanding the Benefits and Limitations of SBOM Types

SBOM Type	Benefits	Limitations
Design	- Highlight incompatible components ahead of licensing purchase or acquisition. - Defines approved or recommended included component list for developer use.	- This may be very difficult to generate. - Unlikely to identify as much detail as found in other SBOM types.
Source	- Provides visibility without access to build process. - Can facilitate remediation of vulnerabilities at the source. - Can provide a view into the dependency tree / hierarchy of the included components.	- Can highlight components (which might have vulnerabilities) that never run or are compiled out in deployed code. - Depending on language/ecosystem, may not include runtime, plugin, or dynamic components, like appserver or platform libraries. - May require references to other SBOMs for completeness..
Build	- Increases confidence that the SBOM representation of the product artifact is correct due to information available during the build and/or Continuous Integration/Continuous Deployment (CI/CD) processes. - Provides visibility into more components than just source code. - Increased trust by enabling signing of the SBOM and product artifact by the same build workflow.	- Potentially have to change the build process to generate this SBOM. - Highly dependent on the build environment in which the build is executed. - May be difficult to capture indirect and/or runtime dependencies. - May not contain the correct versions of dynamically linked dependencies (as they may be replaced at runtime depending on language/ecosystem).
Analyzed	- Provides visibility without an active development environment, such as legacy firmware artifacts. - Does not need access to the build process. - Can help verify SBOM data from other sources. - May find hidden dependencies missed by other SBOM type creation tools.	- May be prone to omissions, errors, or approximations if the tool is unable to decompose or recognize the software components precisely. - May depend on heuristics or context-specific risk factors.
Deployed	- Highlights software components installed on a system, including other configurations and system components used to run an application.	- May require changing install and deploy processes to generate. - May not accurately reflect the software's runtime environment, as components may reside in inaccessible code.
Runtime	- Provides visibility to understand what is in use when the system is running, including dynamically loaded components and external connections. - Can include detailed information about whether components are active and what parts are used.	- Requires the system to be analyzed while running, which may require additional overhead. - Some detailed information may be available only after the system has run for a period of time until the complete functionality has been exercised.

https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

KARAKUN

SBOM —
Benefits and Limitations of SBOM Types

Table 2: Understanding the Benefits and Limitations of SBOM Types

SBOM Type	Benefits	Limitations
Design	- Highlight incompatible components ahead of licensing purchase or acquisition. - Defines approved or recommended included component list for developer use.	- This may be very difficult to generate. - Unlikely to identify as much detail as found in other SBOM types.
Source	- Provides visibility without access to build process. - Can facilitate remediation of vulnerabilities at the source. - Can provide a view into the dependency tree / hierarchy of the included components.	- Can highlight components (which might have vulnerabilities) that never run or are compiled out in deployed code. - Depending on language/ecosystem, may not include runtime, plugin, or dynamic components, like appserver or platform libraries. - May require references to other SBOMs for completeness..
Build	- Increases confidence that the SBOM representation of the product artifact is correct due to information available during the build and/or Continuous Integration/Continuous Deployment (CI/CD) processes. - Provides visibility into more components than just source code. - Increased trust by enabling signing of the SBOM and product artifact by the same build workflow.	- Potentially have to change the build process to generate this SBOM. - Highly dependent on the build environment in which the build is executed. - May be difficult to capture indirect and/or runtime dependencies. - May not contain the correct versions of dynamically linked dependencies (as they may be replaced at runtime depending on language/ecosystem).
Analyzed	- Provides visibility without an active development environment, such as legacy firmware artifacts. - Does not need access to the build process. - Can help verify SBOM data from other sources. - May find hidden dependencies missed by other SBOM type creation tools.	- May be prone to omissions, errors, or approximations if the tool is unable to decompose or recognize the software components precisely. - May depend on heuristics or context-specific risk factors.
Deployed	- Highlights software components installed on a system, including other configurations and system components used to run an application.	- May require changing install and deploy processes to generate. - May not accurately reflect the software's runtime environment, as components may reside in inaccessible code.
Runtime	- Provides visibility to understand what is in use when the system is running, including dynamically loaded components and external connections. - Can include detailed information about whether components are active and what parts are used.	- Requires the system to be analyzed while running, which may require additional overhead. - Some detailed information may be available only after the system has run for a period of time until the complete functionality has been exercised.

Supply Chain Control

You are responsible for dependencies

https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

KARAKUN

SBOM — Generation Pipelines Stages

When?

PIPELINE STAGE	TIMING	PROCESSING MODEL	BEST SUITED FOR	ADVANTAGES	LIMITATIONS
Pre-Build	Before compilation starts	Synchronous	Early dependency validation	• Dependency resolution capture • Early detection of issues	• May miss build-time changes • Incomplete view
Build-Time	Post-compilation, pre-packaging	Synchronous, blocking	Regulated industries, compliance-critical environments	• Most accurate capture • Actual build state • Natural CI/CD integration • Version-aligned with binaries	• Increased build times • Pipeline complexity
Post-Build	After artifact creation	Asynchronous, non-blocking	High-velocity environments, DevOps teams (Cybersecurity Architects/ Implementers)	• No build time impact • Can enrich with additional data • Flexible processing	• Potential drift from build state • Delayed availability
Commit-Triggered	On source control events (push, merge)	Parallel processing	Development teams (Cybersecurity Implementers) , early detection needs	• Immediate feedback • Branch-specific generation • Historical analysis capability	• May generate many versions • Storage overhead
Pre-Deploy	Before production deployment	Synchronous (gate)	Security-focused organisations	• Final validation opportunity • Compliance verification • Quality gates enforcement	• Deployment delays • Late-stage discovery
Runtime	During production operation	Continuous, async	Operations teams, compliance monitoring	• Actual deployed state • Configuration drift detection • Dynamic dependency discovery	• Post-deployment only • Performance overhead • Complex to maintain
Scheduled Batch	Time-based intervals	Bulk processing	Resource-constrained SMEs	• Predictable resource usage • Consolidated processing	• Not real-time • May miss interim changes

https://www.enisa.europa.eu/sites/default/files/2025-12/SBOM%20Analysis%20-%20Towards%20an%20Implementation%20Guide_v1.20-Published.pdf

Table 6 - SBOM Generation Pipeline Stages

SBOM — Generation Pipelines Stages

PIPELINE STAGE	TIMING	PROCESSING MODEL	BEST SUITED FOR	ADVANTAGES	LIMITATIONS
Pre-Build	Before compilation starts	Synchronous	Early dependency validation	• Dependency resolution capture • Early detection of issues	• May miss build-time changes • Incomplete view
Build-Time	Post-compilation, pre-packaging	Synchronous, blocking	Regulated industries, compliance-critical environments	• Most accurate capture • Actual build state • Natural CI/CD integration • Version-aligned with binaries	• Increased build times • Pipeline complexity
Post-Build	After artifact creation	Asynchronous, non-blocking	High-velocity environments, DevOps teams (Cybersecurity Architects/ Implementers)	• No build time impact • Can enrich with additional data • Flexible processing	• Potential drift from build state • Delayed availability
Commit-Triggered	On source control events (push, merge)	Parallel processing	Development teams (Cybersecurity Implementers), early detection needs	• Immediate feedback • Branch-specific generation • Historical analysis capability	• May generate many versions • Storage overhead
Pre-Deploy	Before production deployment	Synchronous (gate)	Security-focused organisations	• Final validation opportunity • Compliance verification • Quality gates enforcement	• Deployment delays • Late-stage discovery
Runtime	During production operation	Continuous, async	Operations teams, compliance monitoring	• Actual deployed state • Configuration drift detection • Dynamic dependency discovery	• Post-deployment only • Performance overhead • Complex to maintain
Scheduled Batch	Time-based intervals	Bulk processing	Resource-constrained SMEs	• Predictable resource usage • Consolidated processing	• Not real-time • May miss interim changes

https://www.enisa.europa.eu/sites/default/files/2025-12/SBOM%20Analysis%20-%20Towards%20an%20Implementation%20Guide_v1.20-Published.pdf

Table 6 - SBOM Generation Pipeline Stages

Release process

Every developer's team concern.

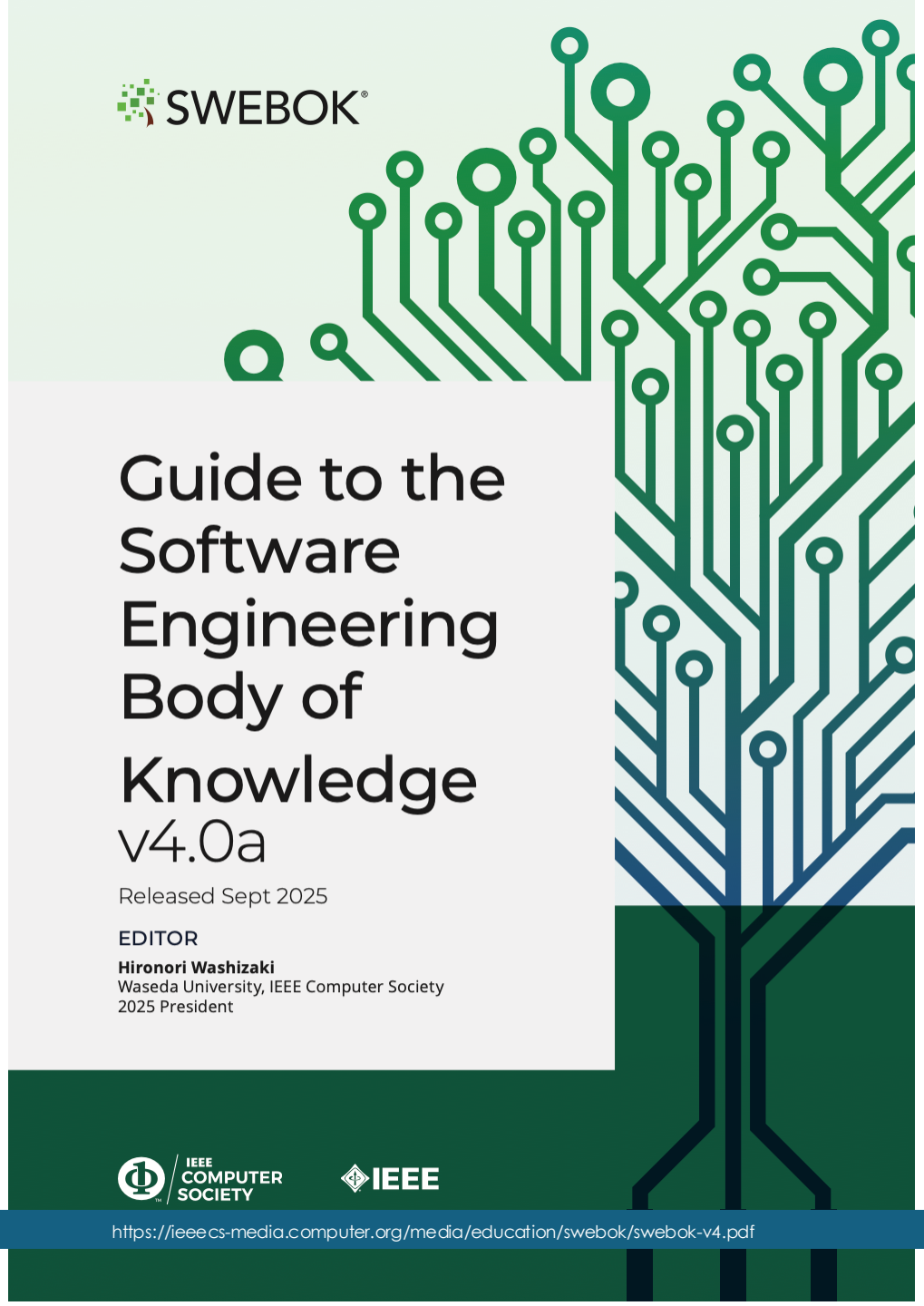
Release process

Every developer's team concern.

Change in culture and tool adoption!

SWEBOK

IEEE COMPUTER SOCIETY



https://ieeecs-media.computer.org/media/education/swebok/swebok-v4.pdf

KARAKUN

Release Process

Scope and change control

Approved changes aligned with configuration management.

Baseline and version management

Controlled baselines with formal traceability.

Reproducible build and packaging

Consistent generation of deployable artifacts.

Verification and validation gates

Testing and auditing before approval.

Controlled deployment and rollback

Staged delivery with rollback mechanisms.

Traceability and documentation

Release notes and change links.

Trusted Release Stack



Release Layer

Jreleaser, Quarkus,

Aggregation Layer

GUAC, JFrog Xray





Transparency Layer

SBOM: CycloneDx, SPDX, SWID Tags

Security & Provenance Layer

SLSA, Sigstore, in-toto





Reproducibility Layer

Reproducible Builds, Maven reproducible plugins, Gradle reproducible plugin

Build Layer

Maven / Gradle

Maven

KARAKUN

Trusted Release Stack



Reproducible builds are a set of software development practices that create an independently-verifiable path from source to binary code. (Find out more)

To learn how to deliver software that your users and contributors can run, rebuild and improve with confidence, refer to the getting started guide.

https://reproducible-builds.org

Trusted Release Stack

SLSA

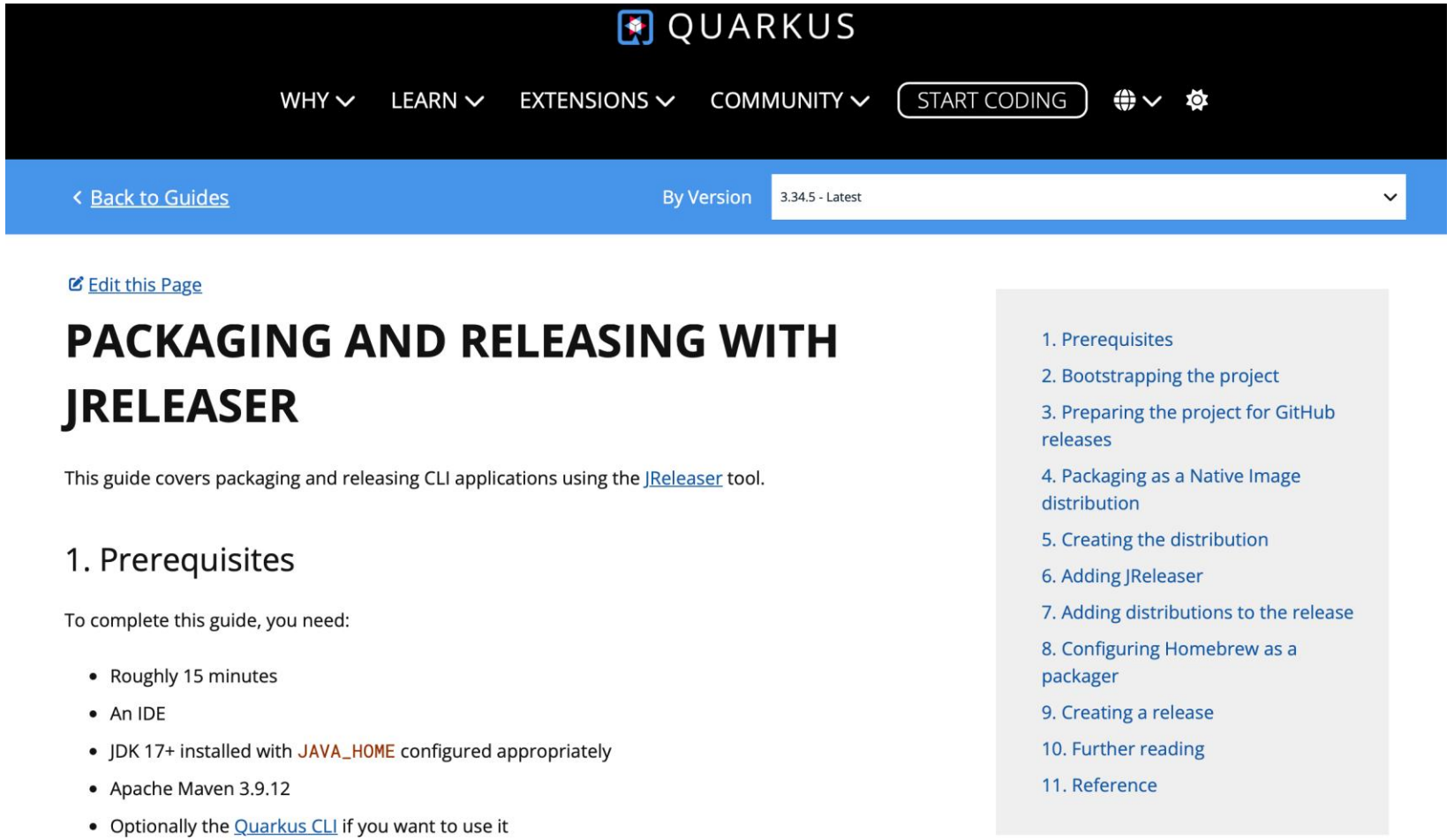


https://slsa.dev

KARAKUN

Trusted Release Stack

Quarkus JReleaser



https://quarkus.io/guides/jreleaser

Trusted Release Stack

Syft

Anchore Open Source Tools.

Developer-friendly scanning tools for container image security.

Syft



A CLI tool for generating a Software Bill of Materials (SBOM) from container images and filesystems.

Try Syft

Watch in action

Grype



An easy-to-integrate open source vulnerability scanning tool for container images and filesystems.

Try Grype

Watch in action

https://anchore.com/opensource/

KARAKUN

Trusted Release Stack

Maven

Home » org.cyclonedx » cyclonedx-maven-plugin

CycloneDX Maven Plugin

The CycloneDX Maven plugin generates CycloneDX Software Bill of Materials (SBOM) containing the aggregate of all direct and transitive dependencies of a project.

Overview | Versions (52) | Used By (510) | Badges | Books (14)

LICENSE	Apache 2.0
CATEGORIES	Maven Plugins
TAGS	plugin, build, build-system, maven, cyclonedx
RANKING	#431483 in MvnRepository, #101 in Maven Plugins
HOMEPAGE	https://github.com/CycloneDX/cyclonedx-maven-plugin, Inspect URL
LINKS	

Home » io.github.zlika » reproducible-build-maven-plugin

Reproducible Build Maven Plugin

This Plug-In allows to generate byte-for-byte reproducible artifacts

Overview | Versions (22) | Used By (166) | Badges | Books (14)

LICENSE	Apache 2.0
CATEGORIES	Maven Plugins
TAGS	plugin, github, build, build-system, maven
RANKING	#542996 in MvnRepository, #169 in Maven Plugins
HOMEPAGE	http://zlika.github.io/reproducible-build-maven-plugin, Inspect URL
LINKS	

KARAKUN

Trusted Release Stack

org.gradlex.reproducible-builds

CC-compatible

Owner: GradleX Team

Reproducibility settings applied to Gradle's built-in tasks.

https://gradlex.org/reproducible-builds

Sources: https://github.com/gradlex-org/reproducible-builds

#gradlex #reproducible builds

org.cyclonedx.bom

Report incorrect plugin description

Owner: CycloneDX Automation

The CycloneDX Gradle plugin creates an aggregate of all direct and transitive dependencies of a project and creates a valid CycloneDX bill-of-materials document from the results. CycloneDX is a lightweight BOM specification that is easily created, human readable, and simple to parse.

https://cyclonedx.org

Sources: https://github.com/CycloneDX/cyclonedx-gradle-plugin.git

#cyclonedx #dependency #dependencies #owasp #bom

Gradle

KARAKUN

Trusted Release Stack



Protobom

Protobom is a protocol buffers representation of SBOM data able to ingest documents in modern SPDX and CycloneDX versions without loss. It has an accompanying Go library generated from the protocol buffers definition that also implements ingesters for those formats.

Standard SBOMs are read by a reader using parsers that understand the common formats. Parsers create a neutral protobom from data read from CycloneDX or SPDX documents.

A protobom can be rendered into standard SBOM formats by the writer using serializers that know how to generate those documents.

SANDBOX OpenSSF

https://openssf.org/projects/protobom/

KARAKUN

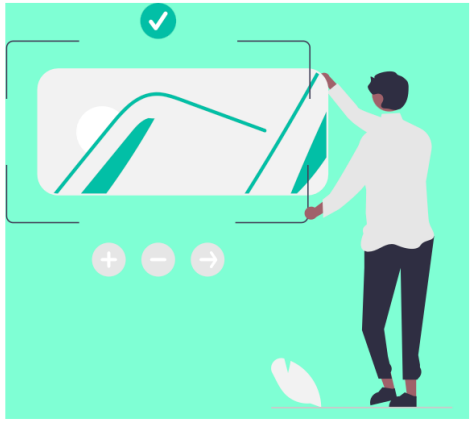
Trusted Release Stack

GUAC

GUAC Trustify Community Contributing Blog Docs GitHub

Graph for Understanding Artifact Composition

Our vision



GUAC (Graph for Understanding Artifact Composition) aims to fill in the gaps by ingesting software metadata, like SBOMs, and mapping out relationships between software. When you know how one piece of software affects another, you'll be able to fully understand your software security position and act as needed.

The current problem with software supply chain security

Software supply chain attacks are on the rise and it's hard to know what your software is at risk for and how to protect it. Many tools are available to help you generate software bills of materials (SBOMs), signed attestations, and vulnerability reports, but they stop there, leaving you to figure out how they all fit together. GUAC provides an aggregated, queryable view across your whole software supply chain, not just one SBOM at a time.

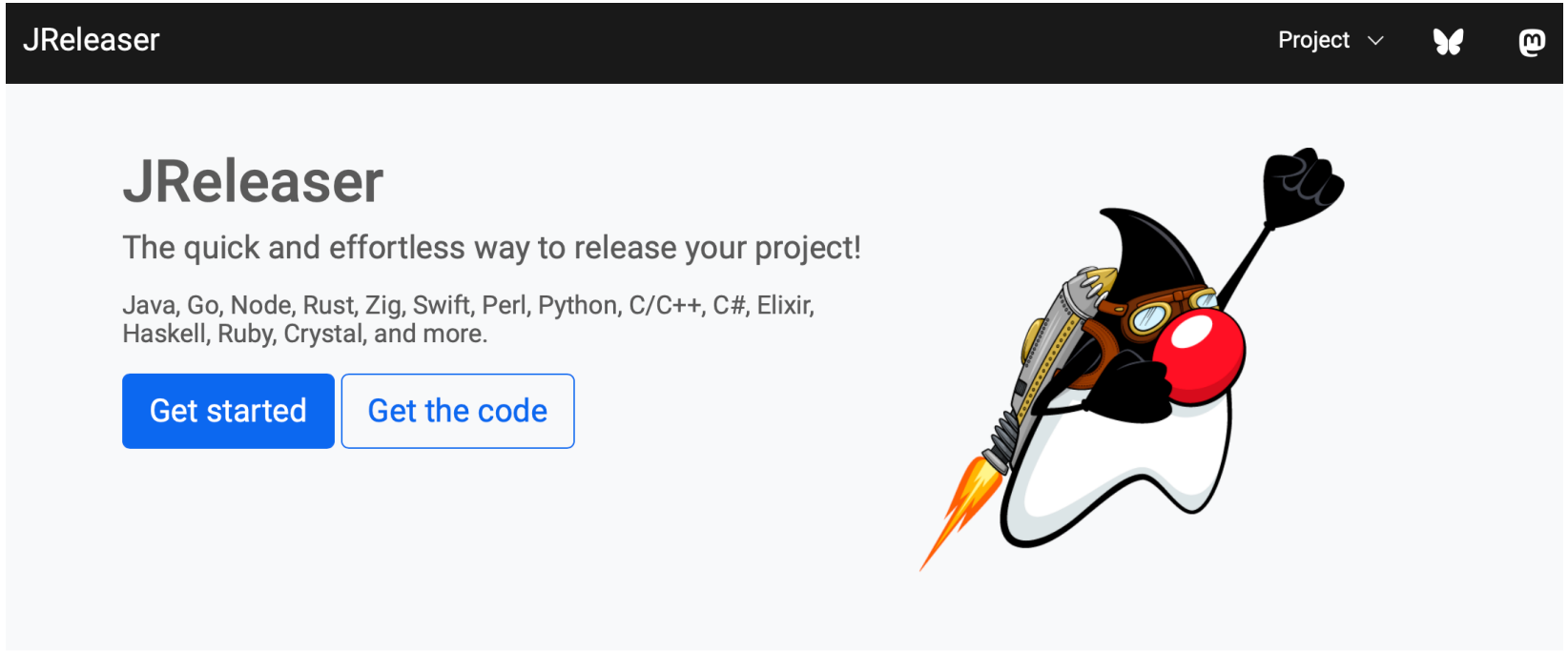
To see how Guidewire Software stays ahead of threats, read the case study.

https://guac.sh/guac/

KARAKUN

Trusted Release Stack

JReleaser



ECLIPSE FOUNDATION

THE LINUX FOUNDATION

Open Regulatory Compliance

OpenSSF
OPEN SOURCE SECURITY FOUNDATION



CD.FOUNDATION

Foundations

(WG & SIG)

OWASP
Open Web Application Security Project

KARAKUN



linkedin.com/in/ixchelruiz

@ixchelruiz@mastodon.social

github.com/ixchelruiz

linkedin.com/in/markusschlichting

@madmas@jit.social

github.com/madmas

Markus.Schlichting@karakun.com





linkedin.com/in/ixchelruiz

@ixchelruiz@mastodon.social

github.com/ixchelruiz

linkedin.com/in/markusschlichting

@madmas@jit.social

github.com/madmas

Markus.Schlichting@karakun.com

CRA security deadlines loom: What senior Java engineers must deliver by December 2027

