Co-funded by the European Union

ECCC EUROPEAN CYBERSECURITY COMPETENCE CENTRE

OCCTET

Open Source Compliance Comprehensive tools and resources designed to simplify and streamline the CRA compliance process for SME, allowing them to tackle the complexities of OSS compliance.

Project Title: Open Source Compliance Comprehensive tools and resources designed to simplify and streamline the CRA compliance process for SME, allowing them to tackle the complexities of OSS compliance.
Project Acronym: OCCTET
Grant Agreement / Contract No.: 101190474
Program: DIGITAL Europe Programme; DIGITAL-ECCC-2024-DEPLOY-CYBER-06
Instrument: DIGITAL JU SME Support Action
Granting Authority: European Cybersecurity Industrial, Technology and Research Competence Centre
Project Start Date: 1 November 2024
Project Duration: 24 months

Deliverable Number: D2.3
Deliverable Title: CRA Adoption Best Practice Document
Deliverable Type (DOA): R — Document, report
Deliverable Type (content): A guide outlining unified specifications for secure FOSS development, enabling SMEs to ensure compliance with the CRA through standardised benchmarks, developed in collaboration with the Eclipse Foundation and key standardisation bodies.
Work Package: WP2 – Define and support compliance procedures
Task Number(s): T2.4-Create Conformity Assessment Specifications
Dissemination Level: PU – Public
Due Date (DoA): 31 March 2026
Actual Submission Date: 31 March 2026
Version: 1.0

Lead Beneficiary: ECL- Eclipse Foundation
Main Author(s): Juan Rico - ECL
Contributing Partner(s): RAL, ECL, EXP
Reviewer: ECL - Eclipse Foundation Europe

Document History

Version	Date	Issued By	Status	Comments
0.1	18-02-2026	ECL	Draft	Initial draft of D2.3
0.2	19-02-2026	EXP	Draft Update	Added aggregated insights from CRA Self-Assessment Platform (Section 6.1 and Annex 2) and alignment mapping with D2.2 maturity domains
0.3	20-02-2026	EXP	Draft Update	Added Annex 1 – SME Practical Implementation Toolkit (MVC checklist, templates, SBOM checklist, vulnerability mini-playbook, update policy template); Added SME Path section (5.5), Decision Matrix (5.5.1), expanded Due Diligence framework (5.4), and structured proportional maturity levels; Added Unified Benchmark Table for Secure FOSS Lifecycle (Section 3.2.7) and harmonised lifecycle + vulnerability handling sections;
1.0	31-03-2026	ECL	Release	Review document and release.

Executive Summary

This document, **D2.3 - CRA Adoption Best Practice Document**, guides Small and Medium-sized Enterprises (SMEs) on achieving compliance with the Cyber Resilience Act (CRA) when using Free and Open Source Software (FOSS).

It emphasizes the **risk-based approach** of the CRA, where risk is contextual to the FOSS component's criticality and usage. The guide promotes a proportional, risk-based philosophy for FOSS developers, focusing on:

- Integrating security across the lifecycle.
- Ensuring governance and transparency (e.g., security contacts).
- Building supply chain resilience (secure release practices).

The document also clarifies the roles of **Open Source Stewards** regarding CRA Article 25 attestations and provides specific **SME CRA compliance guidelines**, including advice on due diligence for consuming FOSS and security attestations.

In summary, the document translates the CRA's risk-based mandate into actionable best practices for the FOSS community and offers SMEs a clear path for secure and compliant FOSS integration.

Table of contents

1 Introduction - Secure FOSS development under the CRA 6
2 Principles of cyber resilience (context for the ones using the FOSS components) 7
2.1 Principles 7
2.1.1 Risk-based approach to cybersecurity 7
3 Secure FOSS Component Development under a Risk-Based Model 8
3.1 Risk-Based Orientation 8
3.2 Lifecycle Integration of Security 8
3.2.1 Design Phase 9
3.2.2 Implementation Phase 9
3.2.3 Testing & Validation 10
3.2.4 Release & Distribution 10
3.2.5 Maintenance 10
3.2.6 End-of-Life 10
3.3 Governance and Transparency as Risk Controls 11
3.4 Relevance to Supply Chain Security 11
3.4.1 Voluntary security attestations 12
3.5 Cybersecurity activities for full life cycle support 13
3.6 Vulnerability handling 13
3.6.1 Core Elements of Vulnerability Handling 14
3.6.2 Proportional Maturity levels 14
3.6.3 Relevance for SMEs Integrating FOSS 15
4 Open Source Stewards 16
4.1 Role and responsibilities 16
4.2 Open source Stewards as compliance enablers 17
4.2.1 Structured Security Policies 17
4.2.2 Coordinated Vulnerability Handling 17
4.2.3 Improved Transparency 17
4.2.4 Regulatory Interface Support 18
4.3 Cybersecurity attestations 18
5 SME CRA compliance guidelines 20
5.1 Context and needs 20
5.1.1 SME Personas and What to Focus On 20
5.2 CRA obligations 21
5.3 Open source compliance ecosystem 21
5.4 Due diligence for Open Source Components 21
5.4.1 Risk-based Assessment of FOSS components 22
5.4.1.1 Proportional Due Diligence 23
5.4.1.2 Ongoing Monitoring and Lifecycle Responsibility 24

5.4.2 Consumption of Security Attestations 24
5.4.3 Common Pitfalls in FOSS Integration 25
5.4.4 Red Flags to Watch for 25
5.4.5 Quick Due Diligence Checklist for SMEs 25
5.5 SME Path to CRA-Asigned FOSS Adoption 26
5.5.1 Decision Matrix for FOSS Components Due Diligence 27
6 Conclusions and recommendations 28
6.1 Observed SME Challenges – Aggregated Platform Insights 28
6.1.1 Platform Engagement and CRA Awareness 28
6.1.2 Overall Maturity Snapshot 29
6.1.3 Domain-Level Observations 29
6.1.4 Structural Gaps Identified Across SMEs 30
6.1.5 Implications for CRA-Aligned Adoption 30
6.1.6 Evidence-Informed Guidance 31
7 Annex 1 - SME Practical Implementation Toolkit 32
7.1 A1.1 Minimum Viable Compliance (MVC) for SMEs Using FOSS Components 32
7.2 A1.2 FOSS Component Register (Template) 33
7.3 A1.3 Open Source Component Record (Template) 34
7.4 A1.4 Vulnerability Handling Mini-Playbook (SME-sized) 34
7.5 A1.5 Dependency Update Policy (Short Form) 35
7.6 A1.6 SBOM Checklist for SMEs (What to capture, when) 36
8 Annex 2 – Aggregated Findings from the OCCTET CRA Self-Assessment Platform 38
8.1 A2.1 Platform Overview 38
8.2 A2.2 CRA Awareness Distribution 38
8.3 A2.3 Domain Maturity Scores (Average) 38
9 ACRONYMS AND ABBREVIATION 40
10 BIBLIOGRAPHY 41

1 Introduction - Secure FOSS development under the CRA

The Cyber Resilience Act (CRA) establishes a horizontal cybersecurity framework for products with digital elements placed on the EU market. A core principle of this regulation is the **risk-based approach to cybersecurity**, which mandates that manufacturers and other economic operators identify, assess, and mitigate cybersecurity risks throughout a product's lifecycle.

This principle is especially pertinent for **Free and Open Source Software (FOSS) components**, which are commonly used as building blocks in both commercial and non-commercial products. While the CRA does not regulate Open Source development per se, it directly impacts how FOSS components are **selected, integrated, maintained, and secured** when incorporated into products with digital elements.

This chapter explains the application of the risk-based approach to open source components and outlines the practical implications for secure FOSS component development.

2 Principles of cyber resilience (context for the ones using the FOSS components)

2.1 Principles

2.1.1 Risk-based approach to cybersecurity

Under the CRA, cybersecurity obligations are not uniform or prescriptive; instead, they are scaled based on:

- The **intended purpose** of the product
- The **foreseeable use and misuse**
- The **severity and likelihood of potential impacts**
- The **role of the software component within the product**

Manufacturers are therefore required to implement "appropriate and proportionate" security measures based on these identified risks, rather than striving for absolute security.
For **FOSS** components, this underscores that **risk is contextual**, not inherent. The same open source library might pose a low risk in one scenario and a high risk in another, entirely dependent on how and where it is deployed.
A risk-based approach to **FOSS** under the CRA necessitates looking beyond the fact that a component is open source and focusing on an assessment of:

- **Functional criticality:** Does the component handle vital functions like authentication, cryptography, update mechanisms, or network exposure?
- **Attack surface:** Is the component exposed to untrusted inputs or external interfaces?
- **Dependency depth and transitivity:** How many indirect dependencies are introduced, and how well are they understood?
- **Maintenance and responsiveness:** Is the project actively maintained? Are security issues promptly acknowledged and addressed?
- **Integration context:** How is the component configured, hardened, and isolated within the final product?

These factors, and not the development model itself, are what determine the required level of due diligence and security controls.

3 Secure FOSS Component Development under a Risk-Based Model

For developers and maintainers of FOSS components, the CRA's risk-based approach translates into **enabling downstream risk management**, rather than assuming direct regulatory responsibility.

More in depth, the BSI (German Federal Office for Information Security) published the "TR-03185-2 Secure Software Lifecycle for Open Source Software" [REF - https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03185/BSI-TR-03185-2.html] that provides a practical reference framework on how to approach open source software development from a security standpoint. This proposal outlines the following principles.

3.1 Risk-Based Orientation

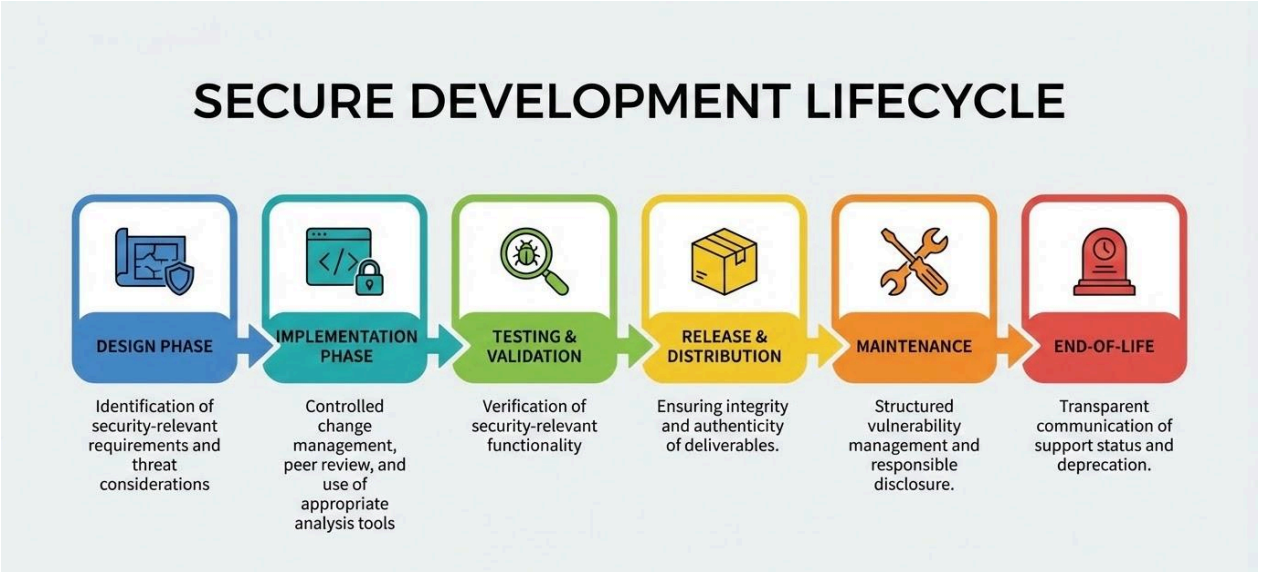
Risk-based philosophy is the core driving force of the CRA. Rather than prescribing rigid, one-size-fits-all controls, the analysis should include outcome-oriented criteria that can be implemented according to:

- The project's criticality and usage context.
- The potential impact of security failures.
- The maturity and resources of the community.
- The exposure of the software within supply chains.

This proportionality is essential for FOSS, where projects range from small volunteer-maintained libraries to widely deployed infrastructure components.

3.2 Lifecycle Integration of Security

Security by design requires that cybersecurity considerations are embedded throughout the entire software lifecycle. In a CRA-aligned, risk-based model, security is not treated as a separate activity but as a continuous and proportionate process spanning design, implementation, testing, release, maintenance, and end-of-life.



The Secure Development Lifecycle illustrated above reflects a structured progression in which each phase contributes to the overall resilience of a FOSS component and supports downstream compliance obligations.

3.2.1 Design Phase

In the design phase, security integration begins with:

- Identification of security-relevant requirements
- Consideration of foreseeable misuse
- Analysis of attack surfaces
- Architectural decisions that reduce exposure

For components with higher functional criticality (e.g., authentication, cryptography, update mechanisms), more structured threat analysis may be appropriate. At a minimum, security-relevant functionality should be documented clearly to enable downstream assessment.

3.2.2 Implementation Phase

During implementation, lifecycle security requires:

- Controlled change management
- Version-controlled repositories
- Peer review practices
- Secure coding discipline

Projects with increased maturity may integrate automated static analysis or dependency scanning tools. The objective is not prescriptive tooling, but consistent and traceable development practices.

3.2.3 Testing & Validation

Testing and validation confirm that security-relevant functionality operates as intended.

Typical measures include:

- Functional testing of authentication and access controls
- Input validation checks
- Basic security testing prior to release

Projects with higher exposure may integrate automated SAST or DAST tools within CI/CD pipelines.
Testing ensures that vulnerabilities are detected before distribution.

3.2.4 Release & Distribution

Secure release practices are critical supply chain controls.
Key practices include:

- Clear versioning
- Documented changelog
- Integrity protection of release artifacts
- Transparency of included dependencies (e.g., SBOM generation)

More advanced projects may implement signed releases or reproducible builds via immutable pipelines.
This phase strengthens downstream trust and traceability.

3.2.5 Maintenance

Security integration does not end at release. Maintenance requires:

- Monitoring newly disclosed vulnerabilities
- Structured vulnerability management
- Timely patching
- Clear Communication of updates

Ongoing maintenance directly supports CRA lifecycle obligations and reduces systematic supply chain risk.

3.2.6 End-of-Life

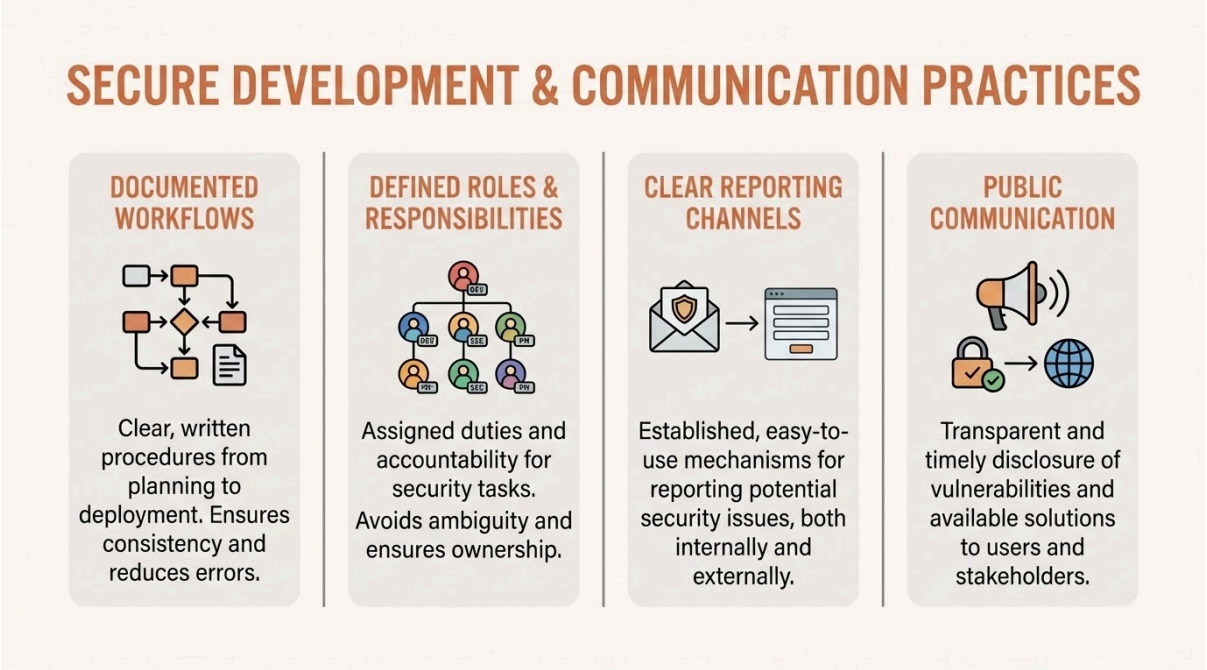
Transparent end of life communication is essential to prevent unmanaged risk.
Projects should:

- Clearly communicate support status
- Announce deprecation timelines
- Recommend migration paths where applicable

This enables downstream users, including SMEs, to plan secure transitions and avoid unsupported dependencies.

3.3 Governance and Transparency as Risk Controls

In open source ecosystems, governance mechanisms often serve as primary risk mitigation tools. Commonly adopted practices in open source projects recognise that transparency, documentation, and community processes are not merely administrative features but core security enablers. Four main elements remain as key as presented in the graph



These measures reduce systemic risk by improving accountability and enabling downstream users to assess the trustworthiness of a project.

3.4 Relevance to Supply Chain Security

Open source components are deeply embedded in modern software supply chains. A risk-based approach to FOSS development must therefore consider not only project-level risks but also systemic risks propagated through reuse and redistribution.

Key aspects to consider supports supply chain resilience by encouraging:

- Secure release practices.
- Traceability of changes.
- Transparent maintenance status.
- Clear security contact point.

Such practices improve the ability of downstream integrators, including small and medium-sized enterprises, to perform risk assessments and meet regulatory or contractual obligations.

3.4.1 Voluntary security attestations

The modern digital economy is built upon a complex, interconnected software supply chain where the security of the end product is inseparable from the integrity of its upstream components. As the **European Cyber Resilience Act (CRA)** moves toward implementation, the industry faces a critical challenge: how to verify the security of the millions of open source dependencies that form the backbone of global infrastructure without stifling the collaborative model that produced them.

The most viable solution lies in **Article 25**, which outlines a framework for voluntary security attestations. By shifting from a model of reactive patching to one of proactive, standardized transparency, Article 25 offers a transformative opportunity to strengthen the entire supply chain. This approach allows non-profit stewards to provide tiered, risk-based assurances in exchange for remuneration that covers transparent operating costs—effectively turning regulatory compliance into a sustainable engine for supply chain resilience.

Three key pillars sustain supply chain resilience

1. **Facilitating Efficient Due Diligence via Trusted Artifacts.** The CRA requires manufacturers to perform rigorous due diligence on every integrated component—including free and open source software. By standardizing attestations as machine-readable, verifiable artifacts, we can eliminate the "denial-of-service attack" on maintainers caused by duplicative, manual compliance requests. This provides manufacturers with consistent, trustworthy information they can integrate directly into their automated risk assessments, streamlining the flow of secure code across the market.
2. **Incentivizing Security Maintenance Across the Supply Chain.** Security is not a static state but a continuous process of maintenance. Attestations create a tangible bridge between regulatory requirements and the economic support necessary to sustain that maintenance. This framework provides a structured pathway for manufacturers to financially support the stewards and maintainers of critical projects whose security work is essential to the manufacturer's own compliance. By aligning remuneration with a project's transparent operating costs, we transform compliance into a driver for security-focused investment.
3. **Strengthening Community-Led Governance and Resilience.** The strength of the software supply chain lies in its diversity. We advocate for an attestation model that is strictly voluntary, proportionate, and risk-based, ensuring that the unique governance models of open source are respected. This approach empowers communities to define their own security postures while offering a "tiered" assurance model. It prevents a "one-size-fits-all" mandate, fostering a collaborative ecosystem where security transparency is a community-led value rather than a legal liability.

3.5 Cybersecurity activities for full life cycle support

Full life cycle support fundamentally reshapes how organizations must approach open source dependencies. A commitment to maintain and secure a product throughout its operational lifetime implies continuous accountability for all incorporated components, including third-party open source libraries. Because open source components evolve independently, their security posture, maintenance status, and governance maturity can change over time. Vulnerabilities may emerge years after integration; maintainers may step back; release cadences may slow. Full life cycle support therefore requires structured dependency visibility, continuous monitoring, version control discipline, and documented update policies. Without these mechanisms, organizations risk inheriting unmanaged technical debt and latent supply chain vulnerabilities.

To address these challenges, several concrete measures should be implemented:

1. Manufacturers should maintain a continuously updated Software Bill of Materials (SBOM) to ensure transparency and traceability of dependencies across product versions.
2. Dependency risk assessment should be institutionalized through criteria such as project activity level, vulnerability response time, governance transparency, and release discipline.
3. Manufacturers should adopt continuous vulnerability monitoring.

Beyond internal controls, sustainable lifecycle management increasingly requires upstream engagement. Where dependencies are mission-critical, organizations should consider contributing fixes, sponsoring maintainers, or participating in project governance to reduce systemic risk.

In sum, full life cycle support demands moving from passive consumption of open source to active, risk-based monitoring and participation. By combining transparency (SBOMs), structured risk assessment, continuous monitoring, and upstream participation, manufacturers can transform open source dependencies from unmanaged exposure into a strategically governed asset that supports long-term security, compliance, and trust.

3.6 Vulnerability handling

Vulnerability handling is a central element of lifecycle security and a key component of the CRA's risk-based framework.
Under the Cyber Resilience Act, manufacturers are required to identify, manage, and remediate vulnerabilities in a timely and structured manner throughout the lifecycle of the product. While open source projects are not directly regulated as such, their practices significantly influence the ability of downstream manufacturers and SMEs to comply with these obligations.

In a risk-based model, vulnerability handling is not limited to reactive patching. It encompasses:

- Accessible reporting mechanisms
- Structured triage and risk assessment
- Transparent remediation processes
- Transparent disclosure practices
- Continuous monitoring of known vulnerabilities

For FOSS components, vulnerability management must balance proportionality with responsibility. Smaller projects may not have the resources of large commercial vendors, but they should still implement baseline practices that enable downstream integrators to assess and manage risk effectively.

3.6.1 Core Elements of Vulnerability Handling

A structured vulnerability handling framework should include the following elements:

1. Reporting and Intake
    a. A publicly available security contact or reporting channel
    b. Clear instructions for responsible disclosure
    c. Defined intake and acknowledgement procedure

    This ensures that security researchers and users can report issues in a controlled and transparent manner.
2. Triage and Risk Assessment
    a. Evaluation of severity and potential impact
    b. Identification of affected versions
    c. Assessment of exploitability and exposure

    Severity classification may rely on widely accepted frameworks (e.g., CVSS), while maintaining proportionality to the project's size and context.
3. Remediation and Release Management
    a. Development of corrective patches
    b. Secure release process
    c. Clear versioning and changelog documentation

    Where feasible, updates should be accompanied by clear communication on mitigation steps and affected configurations.
4. Disclosure and Transparency
    a. Public advisory publication when appropriate
    b. Clear communication to downstream users
    c. Coordinated disclosure where multiple stakeholders are involved

    These measures enhance trust and support supply chain transparency.

3.6.2 Proportional Maturity levels

To reflect the CRA's proportionality principle, vulnerability handling practices can be structured across three maturity levels.

- **Level 1 – Minimum Viable Practice**
  - Public vulnerability reporting channel
  - Manual triage of reported issues
  - Patch publication and changelog update

  This level enables basic traceability and downstream risk management.
- **Level 2 – Structured Process**
  - Documented vulnerability management policy
  - Defined triage workflow and internal tracking
  - Public advisory publication
  - Defined target response time

  This level improves predictability and transparency.
- **Level 3 – Advanced and Supply Chain Integrated**
  - Coordinated Vulnerability Disclosure (CVD) policy
  - CVE assignment where applicable
  - Continuous vulnerability monitoring tools
  - SLA-based remediation tracking
  - Integration of vulnerability scanning in CI/CD pipelines

  This level strengthens systemic resilience across the software supply chain.

3.6.3 Relevance for SMEs Integrating FOSS

For SMEs integrating FOSS components into products with digital elements, vulnerability handling practices of upstream projects directly affect their own compliance posture.

SMEs should therefore assess:

- Whether a project has a public vulnerability disclosure channel
- How quickly issues are acknowledged and resolved
- Whether advisories and patches are transparently communicated
- Whether the project demonstrates maintenance activities

Where a component is functionally critical or publicly exposed, SMEs should consider applying enhanced due diligence measures, including continuous monitoring and, where available, reliance on voluntary security attestations under Article 25.

4 Open Source Stewards

The particularities of open source are well considered by the CRA, and it introduces the open source stewards as a new actor. The Open Regulatory Compliance community has developed a white paper that reflects the expected role of stewards under the CRA [REF - https://orcwg.org/files/cra/resources/white-paper-on-open-source-software-stewards-and-cra.pdf]. This specific category is designed for entities that:

- Systematically and sustainably support the development of specific open source software.
- Play a governance or coordination role.
- Do not place the software on the market as a commercial product.

Typical examples include foundations, non-profit organisations, and other coordinating bodies that:

- Oversee governance structures.
- Maintain infrastructure.
- Coordinate security processes.
- Support long-term maintenance.

The key legal distinction is that stewards are **not manufacturers**, provided they do not commercialise the software in a way that would qualify as placing a product with digital elements on the EU market.

This distinction is critical for maintaining the open source development model while introducing structured accountability.

4.1 Role and responsibilities

The CRA imposes a tailored set of obligations on open source stewards. These are proportionate and reflect their non-commercial role.

Stewards must:

- Establish and document a verifiable cybersecurity policy.
- Support secure development practices.
- Coordinate vulnerability handling processes.
- Document, address, and facilitate remediation of vulnerabilities.
- Share relevant vulnerability information with affected projects.
- Cooperate with national authorities and Market Surveillance Authorities (MSAs).

Importantly:

- Stewards are not subject to the same administrative fines as manufacturers.
- Their obligations focus on process, coordination, and transparency rather than product conformity.

The legislative intent is clear: strengthen the resilience of open source infrastructure without transforming stewards into regulated product manufacturers.

4.2 Open source Stewards as compliance enablers

Open source stewards can act as an intermediary governance layer between community development and regulated product manufacturers.

For SMEs, this creates several advantages:

4.2.1 Structured Security Policies

Stewards formalise security policies that SMEs can reference when documenting:

- Secure development practices.
- Upstream governance models.
- Vulnerability coordination processes.

This reduces the need for SMEs to individually assess each community from scratch.

4.2.2 Coordinated Vulnerability Handling

Stewards often manage:

- Security mailing lists.
- Disclosure channels.
- Patch coordination.
- Public advisories.

SMEs can integrate these processes into their own vulnerability management frameworks.

4.2.3 Improved Transparency

Stewards typically provide documentation regarding:

- Governance structure.
- Release management.
- Maintenance commitments.
- Security posture.

This documentation supports CRA technical files and risk assessments.

4.2.4 Regulatory Interface Support

When MSAs request information, stewards can:

- Provide upstream documentation.
- Clarify vulnerability handling processes.
- Support transparency around development practices.

While responsibility remains with the manufacturer, structured stewardship reduces uncertainty.

4.3 Cybersecurity attestations

The Cyber Resilience Act recognises the unique role of open source ecosystems within the European digital landscape. While FOSS development is not directly regulated as commercial manufacturing, the CRA introduces the concept of voluntary security attestations under Article 25, enabling structured transparency across the supply chain.

The introduction of voluntary security attestations under the CRA represents a significant opportunity to bridge the gap between regulatory expectations and practical implementation realities. Unlike traditional conformity assessment models that rely on external audits or one-size-fits-all certifications, CRA attestations are envisaged as a flexible, risk-based mechanism whereby organisations can provide contextualised evidence about their processes, practices, and product characteristics in a structured way. By anchoring compliance effort in measurable artefacts and documented practices, attestations offer manufacturers a pragmatic route to demonstrate due diligence and conformity with security requirements across the product lifecycle.

For open source communities and stewards, CRA attestations create an opportunity to contribute directly to regulatory readiness without assuming the full burdens of being classified as a manufacturer. Attestations allow stewards to communicate risk-based assurances about the governance, security practices, and maintenance commitments of projects, helping downstream integrators and manufacturers to make informed decisions. When these assurances are aligned with recognised best practices and transparent operating costs, they can form a reusable and scalable compliance asset across the broader ecosystem. This not only reduces duplication of effort for individual manufacturers but also reinforces the role of stewardship as a trusted bridge between collaborative development and regulatory compliance frameworks.

For SMEs, which may lack an extensive internal compliance infrastructure, the attestation model offers particular value. By relying on steward-produced attestations and integrating them into their own compliance artefacts, SMEs can reduce the cost, complexity, and uncertainty associated with implementing CRA requirements. Attestations promote clarity

and predictability, helping organisations prioritise resources where risk is highest and enabling a shared vocabulary for compliance evidencing. When coupled with robust tooling, machine-readable artefacts, and community coordination, this approach can lower barriers to compliance, support more resilient supply chains, and encourage cooperative governance models that benefit the European digital ecosystem as a whole.

5 SME CRA compliance guidelines

5.1 Context and needs

5.1.1 SME Personas and What to Focus On

SMEs approach CRA adoption from different roles and levels of technical maturity. To make this guide practical, the following “personas” illustrate common responsibilities and what each role should prioritise when using and integrating FOSS components.

Persona A – Non-technical manager (CEO/COO/Operations Manager)
Focus: making sure the business can demonstrate a reasonable, repeatable compliance approach without needing deep technical detail.

- Ensure roles are assigned (who owns updates, who monitors vulnerabilities, who approves dependencies).
- Ensure basic evidence exists (SBOM availability, security contact, update policy, vulnerability reporting channel).
- Approve time and budget for “minimum viable compliance” activities.
- Require simple documentation: “what we use, why we use it, how we keep it updated”.

Persona B – Technical lead (CTO/Lead Developer/DevOps Lead)
Focus: implementing practical controls that reduce risk and can be maintained over time.

- Integrate security into the lifecycle (review, testing, secure releases, updates).
- Establish dependency visibility (SBOM, dependency scanning, tracking transitive dependencies).
- Implement vulnerability handling (intake, triage, patching process, release notes).
- Ensure traceability (versioning discipline, changelog, protected branches, access controls).

Persona C – Product owner / Compliance coordinator (Product Manager / QA / Security Champion)
Focus: connecting product decisions, customer expectations, and compliance evidence.

- Clarify product scope and intended use (where FOSS is used, where exposure exists).
- Apply a risk-based view: which components are critical vs low-risk.
- Maintain lightweight evidence packs (risk decisions, component register, update history).
- Coordinate across teams so monitoring and updates don’t fall between roles.

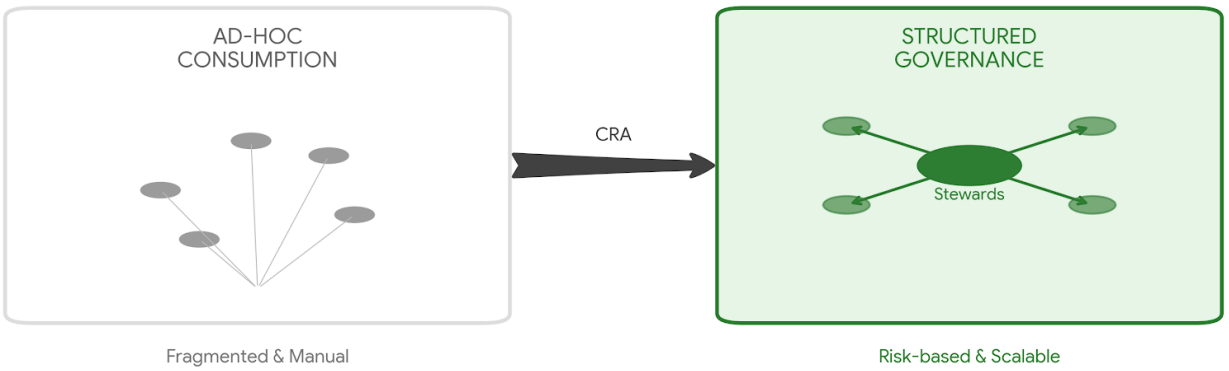
These personas are not mutually exclusive. In many SMEs, one person may cover multiple roles. The key is ensuring that ownership and accountability are explicitly assigned.

5.2 CRA obligations

5.3 Open source compliance ecosystem

The CRA signals a broader transformation: compliance is no longer limited to internal product design. It extends across the software supply chain.

For SMEs, this implies a shift from ad-hoc open source consumption to structured, risk-based ecosystem governance.



Open source stewards become key actors in this governance architecture, and the consumption of non-stewarded components will be subject of intense scrutiny to gauge properly the risks associated with it.

Rather than individually auditing every upstream project, SMEs can:

- Leverage steward-aligned processes.
- Rely on structured security frameworks.
- Document engagement as part of due diligence.
- Demonstrate proportionate, risk-based management.

This approach aligns with the CRA's intent: improving cybersecurity without imposing a disproportionate burden.

5.4 Due diligence for Open Source Components

The integration of Free and Open Source Software (FOSS) components into products with digital elements introduces shared responsibility across the software supply chain.

Under the CRA's risk-based framework, manufacturers — including SMEs — are expected to apply appropriate and proportionate due diligence when selecting, integrating, and maintaining third-party software components.

Due diligence is not intended to discourage the use of open source software. On the contrary, FOSS remains a foundational enabler of innovation and competitiveness. However, open source software must be integrated responsibly.

Due diligence ensures that:

- You understand what you are integrating
- You know the level of risk it introduces
- You can justify your integration decision
- You are able to monitor and maintain it over time

Importantly, due diligence is not a one-time check before release. It is a lifecycle-oriented activity that continues throughout the product's expected lifetime. This aligns directly with the CRA's requirements for secure development and vulnerability management.

5.4.1 Risk-based Assessment of FOSS components

Before integrating a FOSS component, SMEs should assess three key dimensions.

These questions are designed to be understandable for both technical and non-technical decision-makers.

Functional Criticality
Ask:

- Does this component handle authentication, encryption, access control, or updates?
- Is it central to the product's core functionality?
- If this component fails, would our product stop working?
- Could it compromise data confidentiality, integrity, or availability?

The more critical the function, the more careful the evaluation should be.
A logging library is not the same as an authentication module.

Exposure Level
Ask:

- Is this component accessible from the internet?
- Does it process input from external users?
- Is it exposed through APIs?
- Or is it fully internal and isolated?

Higher exposure increases the attack surface.
An internal utility component may require lighter review than a publicly exposed service component.

Impact of Failure
Ask:

- Could exploitation lead to personal data breaches?
- Would service disruption impact customers?
- Would this trigger regulatory notification obligations?
- Could it damage our reputation?

Impact determines how much effort is justified in risk management.
Important: These Factors Work Together

These dimensions must be considered together. For example:

- A moderately critical component that is publicly exposed may require enhanced controls.
- A highly critical component used only internally may justify structured but proportionate review.

The goal is proportionality, not uniform control.

5.4.1.1 Proportional Due Diligence

Consistent with the CRA's proportionality principle, due diligence practices may be structured across increasing levels of maturity.

Level 1 – Foundational Due Diligence
Appropriate for components with low criticality and limited exposure.
Measures include:

- Verification of project activity and recent maintenance
- Review of publicly disclosed vulnerabilities
- License compatibility verification
- Inclusion of the component in the Software Bill of Materials (SBOM)

At this level, the objective is transparency and traceability. For many SMEs, this represents a realistic and achievable baseline aligned with proportional compliance expectations.

Level 2 – Structured Due Diligence
Appropriate for components with moderate criticality or exposure.
Additional measures may include:

- Review of governance transparency and maintainer structure
- Assessment of vulnerability reporting and response practices
- Evaluation of release discipline and versioning clarity
- Automated dependency scanning tools

Structured due diligence introduces predictability. It reduces reliance on ad-hoc decisions and supports documented internal risk assessments.

Level 3 – Enhanced Due Diligence
Appropriate for highly critical or publicly exposed components.
Enhanced measures may include:

- Continuous vulnerability monitoring
- Review of coordinated disclosure practices
- Evaluation of long-term maintenance commitment
- Consideration of voluntary security attestations under Article 25
- Engagement with maintainers where appropriate

This level is particularly relevant for components that:

- Handle sensitive data
- Are publicly exposed
- Support essential product functionality
- Operate in regulated or high-impact environments

Enhanced due diligence strengthens resilience across the broader software supply chain. Templates supporting Level 1–3 are provided in Annex 1.

5.4.1.2 Ongoing Monitoring and Lifecycle Responsibility

Due diligence does not end at the moment of integration. Vulnerabilities may be discovered years later. Maintainers may change. Releases may slow down.

Lifecycle responsibility requires continuous awareness of evolving vulnerabilities and project maintenance status.

SMEs should:

- Monitor vulnerability disclosures affecting included dependencies
- Track new releases and security updates
- Maintain updated SBOM records
- Reassess risk when exposure or architecture changes

Reassessment is particularly important when:

- New product features are introduced
- Public exposure increases
- Regulatory requirements change
- Upstream maintenance activity declines

This continuous approach aligns directly with the CRA's lifecycle security obligations.

5.4.2 Consumption of Security Attestations

Where available, voluntary security attestations under Article 25 may support structured due diligence. Attestations can:

- Provide standardised and comparable information on governance and security practices
- Reduce repetitive compliance requests directed at maintainers
- Facilitate integration into internal risk assessment processes

However, reliance on attestations does not eliminate integrator responsibility. SMEs must evaluate the relevance of attested information in relation to their own integration context, exposure level, and product architecture.
Attestations enhance transparency but do not replace contextual risk assessment.

5.4.3 Common Pitfalls in FOSS Integration

SMEs frequently encounter recurring challenges when integrating FOSS components.
Common pitfalls include:

- Assuming that widespread adoption automatically implies security
- Failing to identify indirect (transitive) dependencies
- Conducting one-time validation without ongoing monitoring
- Relying on upstream patching without tracking update adoption
- Integrating components without documenting selection rationale

Recognising these patterns enables SMEs to shift from reactive integration toward structured risk management.

5.4.4 Red Flags to Watch for

When evaluating a FOSS component, certain indicators may suggest elevated risk.
Examples include:

- No visible security reporting channel
- Long periods without updates or maintenance
- Unresolved critical vulnerabilities without communication
- Inconsistent versioning or release documentation
- Lack of clarity regarding maintainer responsibility
- Large dependency trees with limited transparency

The presence of a red flag does not automatically disqualify a component. However, multiple indicators may justify enhanced due diligence or reconsideration of integration strategy.

5.4.5 Quick Due Diligence Checklist for SMEs

To support practical implementation, the following simplified checklist may serve as a starting point.

Basic Visibility

- ☐ Is the project actively maintained?
- ☐ Is version history transparent?
- ☐ Is a security contact publicly available?

Vulnerability Awareness

- ☐ Are vulnerabilities documented?
- ☐ Are fixes released in a structured manner?
- ☐ Is there evidence of responsiveness?

Supply Chain Transparency

- ☐ Is the component included in the SBOM?

- ☐ Are dependencies visible?
- ☐ Is versioning consistent?

Lifecycle Considerations

- ☐ Is there evidence of ongoing maintenance?
- ☐ Are updates communicated clearly?
- ☐ Is end-of-life status defined?

This checklist does not replace structured risk assessment but provides an accessible and proportionate entry point for SMEs with limited resources.

5.5 SME Path to CRA-Asigned FOSS Adoption

The CRA is risk-based and lifecycle-oriented. SMEs can adopt a practical approach by following a simple “path” that can be scaled depending on product criticality and resources.

Step 1 – Start (Set ownership and baseline visibility)

- Assign responsibility for: dependency selection, security updates, vulnerability monitoring, and documentation.
- Create a minimal inventory of key FOSS components used in the product.

Step 2 – Identify scope (Where FOSS matters in your product)

- Identify which parts of the product are internet-facing, handle data, or are security-sensitive.
- Clarify which components are critical to product functionality.

Step 3 – Assess risk (Use proportional thinking)

- Evaluate each component using: criticality, exposure, and impact.
- Decide what “level of due diligence” is reasonable (foundational / structured / enhanced).

Step 4 – Apply proportional controls (Do what is appropriate)

- For low-risk components: basic checks + SBOM + monitoring.
- For higher-risk components: structured governance checks, stronger release discipline, more monitoring.

Step 5 – Monitor (Make it continuous, not one-time)

- Track vulnerabilities and upstream updates.
- Keep SBOM updated for each release (or at least major releases).
- Ensure updates are adopted in a timely way.

Step 6 – Reassess (When things change)

- The product becomes more exposed (new integrations, internet exposure),
- Critical functionality changes,
- Upstream maintenance declines,
- New vulnerabilities appear that affect core dependencies.

This path is designed to help SMEs demonstrate a structured and defensible approach, even with limited resources.

5.5.1 Decision Matrix for FOSS Components Due Diligence

To support consistent and proportionate decision-making, SMEs may use the following simplified decision matrix.
The matrix combines two primary dimensions:

- Functional Criticality (Low / Medium / High)
- Exposure Level (Internal / Limited / Public)

The result indicates the recommended Due Diligence Level (L1 / L2 / L3) as defined in Section 5.4.1.1.

Functional Criticality ↓ / Exposure →	**Internal (isolated use)**	**Limited (controlled access)**	**Public (internet-facing)**
Low Criticality	L1 – Foundational	L1 – Foundational	L2 – Structured
Medium Criticality	L1 – Foundational	L2 – Structured	L3 – Enhanced
High Criticality	L2 – Structured	L3 – Enhanced	L3 – Enhanced

How to Use This Matrix

1. Determine the functional criticality of the component:
    a. Does it support core functionality?
    b. Does it handle authentication, encryption, or updates?
    c. Would failure significantly impact the product?
2. Determine the exposure level:
    a. Is it internal only?
    b. Is it accessible through authenticated APIs?
    c. Is it publicly exposed?
3. Locate the intersection in the matrix to identify the recommended due diligence level.

Important Considerations

- This matrix provides guidance, not rigid classification.
- If the impact of failure is exceptionally high (e.g., safety, regulated environment), SMEs may choose to apply a higher due diligence level.
- The matrix supports proportionality, avoiding both over-engineering and under-assessment.

6 Conclusions and recommendations

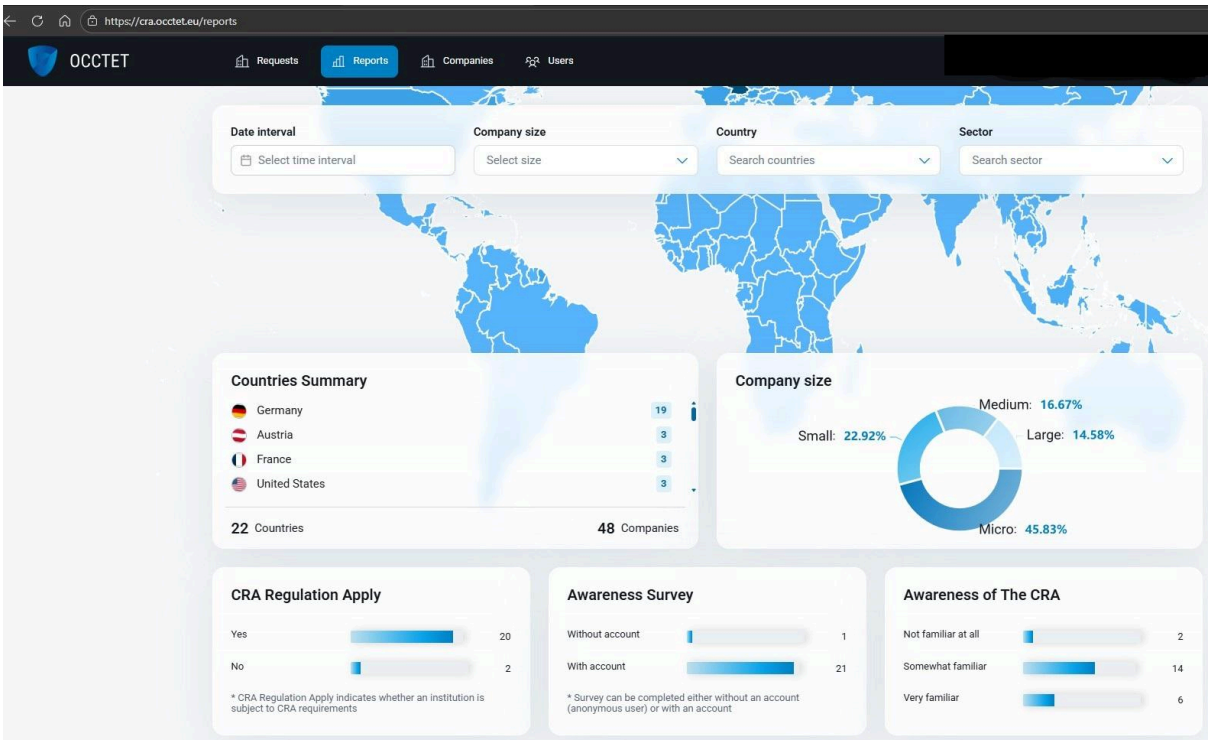
6.1 Observed SME Challenges – Aggregated Platform Insights

To complement the best practice guidance provided in this document, anonymised and aggregated findings from the OCCTET CRA Self-Assessment Platform (https://cra.occtet.eu/) provide insight into the current maturity landscape of SMEs with regard to provide insight into the current maturity landscape of organizations voluntarily assessing their preparedness for CRA-aligned cybersecurity and lifecycle practices.

The data presented below reflects voluntary assessments conducted across 22 countries and 48 registered companies. All findings are fully anonymised and aggregated. No company-level or personal data is included.

The purpose of this section is not to evaluate individual organisations, but to understand common patterns and recurring challenges. These insights directly informed the structure and priorities of this deliverable.

6.1.1 Platform Engagement and CRA Awareness



The platform engagement shows encouraging participation:

- Registered companies: 48
- Countries represented: 22
- Organisations declaring they are subject to CRA requirements: 20

- Organisations declaring they are not subject: 2

CRA awareness levels indicate that most SMEs are not starting from zero:

- Not familiar at all: 2
- Somewhat familiar: 14
- Very familiar: 6

This suggests that awareness of the CRA is growing. However, awareness alone does not automatically result in structured implementation. Many SMEs understand that security and compliance matter, but are still in the process of translating that understanding into clear, documented practices.

6.1.2 Overall Maturity Snapshot

The aggregated average maturity score across domains is:1.66 (on a 0–3 scale).

This score reflects early-to-intermediate maturity across most SMEs, with stronger implementation in technical security controls than in governance, documentation, and lifecycle processes.

In practical terms, this indicates that many SMEs:

- Many SMEs already implement certain technical protections.
- Fewer have fully structured governance, documentation, and lifecycle processes.
- Security efforts often exist, but are not always formalised or consistently recorded.

This is a common and understandable situation for organisations operating with limited resources and fast development cycles.

6.1.3 Domain-Level Observations

Higher relative maturity areas:

- Confidentiality – 2.16

- Resilience – 2.06
- Network Security – 1.90

Lower relative maturity areas:

- Risk Management & Governance – 1.11
- Logging & Detection – 1.34
- Release Requirements – 1.48
- Secure Software Development Lifecycle – 1.54
- Vulnerability Management & Disclosure – 1.63

This pattern shows that technical controls are often more developed than process-oriented or governance-related practices.

For example:

1. Technical controls are more mature than structured governance:Encryption may be implemented, but ownership for dependency updates may not be formally assigned..
2. Patches may be applied, but release traceability may not be systematically documented.
3. Security improvements may be made, but risk decisions may not be recorded.

The CRA places emphasis not only on technical protection, but on lifecycle accountability and demonstrable structure. The gap between “doing security” and “structuring security” is therefore a key area of focus.

6.1.4 Structural Gaps Identified Across SMEs

Based on aggregated scoring patterns, the following recurring structural gaps emerge:

- Limited documented risk management framework
- No clearly assigned ownership for dependency updates
- Absence of formalised release documentation discipline
- Incomplete SBOM or dependency visibility
- Reactive rather than lifecycle-oriented vulnerability management

These gaps do not indicate a lack of commitment to security. Rather, they reflect the operational reality of SMEs where security responsibilities are often distributed across small teams. The challenge is not introducing complexity, but introducing clarity.

6.1.5 Implications for CRA-Aligned Adoption

The aggregated findings reinforce several conclusions:

1. SMEs benefit most from proportional, structured, and lightweight guidance.
2. Governance and lifecycle clarity are more critical gaps than pure technical controls.
3. Documentation, traceability, and repeatability are key enablers of CRA alignment.
4. Simple ownership assignment and visibility mechanisms can significantly improve maturity.

For this reason, this document focuses on practical tools and proportional benchmarks rather than heavy compliance models.

The Decision Matrix, Unified Lifecycle Benchmarks, and SME Toolkit (Annex 1) are specifically designed to help SMEs formalise existing practices without creating unnecessary overhead.

6.1.6 Evidence-Informed Guidance

The recommendations in this deliverable are directly informed by the observed maturity patterns:

- The lifecycle benchmarks respond to gaps in structured SDLC and release discipline.
- The vulnerability handling framework addresses informal triage and remediation workflows.
- The SBOM and dependency templates respond to limited supply chain visibility.
- The SME Path supports organisations moving from awareness toward structured implementation.

In this way, this d is not solely normative guidance, but evidence-informed best practice grounded in real SME self-assessment data collected within the OCCTET framework.

7 Annex 1 - SME Practical Implementation Toolkit

This Annex provides reusable templates and practical guidance to help SMEs implement a proportionate and lifecycle-oriented approach to integrating Free and Open Source Software (FOSS) components under the Cyber Resilience Act (CRA).

The tools below are intentionally lightweight. They are designed to be scalable depending on product criticality, exposure, and available resources.

7.1 A1.1 Minimum Viable Compliance (MVC) for SMEs Using FOSS Components

This one-page checklist describes a baseline set of practices that most SMEs can implement to support CRA-aligned integration of open source components. It is designed for SMEs with limited time and resources and should be strengthened proportionally based on risk.

A. Roles and Ownership (Who does what?)
✅ Assign an owner for:

- Dependency selection and approval
- Vulnerability monitoring
- Security updates / patching
- Release documentation (what changed, what was updated)

Clear ownership prevents security responsibilities from becoming implicit or unassigned.

B. Component Visibility (Know what you use)
✅ Maintain a simple “FOSS Component Register” including:

- Component name + version
- Where it is used (module / product area)
- Criticality (low/medium/high)
- Exposure (internal/limited/public)
- Update owner

✅ Generate an SBOM (at least for major releases) or maintain an equivalent dependency list.
Visibility is the foundation of risk-based management.

C. Basic Due Diligence (Before integration)
✅ For each important component, confirm:

- It is actively maintained (recent activity/release signals)
- Known vulnerabilities are reviewed
- License compatibility is checked
- A security contact or disclosure path exists (when applicable)

D. Vulnerability Handling (What you do when an issue appears)

✅ Have a basic internal process that answers:
- How do we receive vulnerability alerts?
- Who triages?
- How do we decide urgency?
- How do we patch and release?
- How do we communicate changes (release notes)?

E. Updates and Lifecycle (Keep it secure over time)
✅ Define a simple update policy:
- How often do we review updates? (e.g., monthly + urgent patches asap)
- How do we handle end-of-life components?
- Who approves dependency upgrades?

✅ Track at least:
- Date of last dependency review
- Date of last security update / patch release

F. Evidence (Be able to show you did it)
✅ Keep lightweight evidence that can be shown if needed:
- Component register + SBOM
- Update policy (1 page is enough)
- Vulnerability handling notes (process + at least one example log entry)
- Release notes referencing dependency/security updates

MVC principle: Start small, be consistent, and scale controls based on risk.

7.2 A1.2 FOSS Component Register (Template)

SMEs should maintain a simple register of the open source components integrated into their product.

This register supports traceability, due diligence, and lifecycle monitoring.
The table below provides a lightweight template that can be maintained in Excel, SharePoint, Git, or any internal tool.

Example (Filled Row):

Component Name	Version	Used in	Criticality	Exposure	License	Maintainer Activity	Last Vulnerability Review	Update Owner	Notes
ExampleAuthLib	2.4.1	Authentication	High	Public	MIT	Active (mon	12 Feb 2026	CTO	Security-sensitive. CVE-2025-XXXX patched in v2.4.1.

		modu le				thly relea ses)			Continuous monitoring enabled.

7.3 A1.3 Open Source Component Record (Template)

This record can be used for medium and high criticality components (or any component that is publicly exposed). It complements the FOSS Component Register by documenting why the component was selected and what monitoring is in place.

Open Source Component Record

- Component name:
- Version / range used:
- Repository / reference link (optional):
- Used in (module / feature):
- Functional criticality (Low / Medium / High):
- Exposure (Internal / Limited / Public):
- License:

Selection rationale (why this component?)

- Why was it chosen (e.g., feature fit, stability, performance, ecosystem)?
- Alternatives considered (if any):

Basic due diligence performed (tick or describe briefly)

- ☐ Maintainer activity reviewed
- ☐ Vulnerability history reviewed
- ☐ Security contact / disclosure method available (if applicable)
- ☐ License compatibility confirmed
- ☐ Included in SBOM

Known risks / notes

- Known limitations / concerns:
- Dependency depth concerns (if relevant):

Monitoring and ownership

- Update owner / responsible person:
- Vulnerability monitoring source(s): (e.g., advisories, CVE feeds, repository alerts)
- Review frequency: (e.g., monthly + urgent alerts as needed)
- Last review date:
- Next planned review date:

7.4 A1.4 Vulnerability Handling Mini-Playbook (SME-sized)

This playbook provides a structured but lightweight vulnerability workflow.

1) Intake (How we receive alerts)

Sources monitored:

- ☐ Project advisories

- ☐ CVE databases
- ☐ Dependency scanning tool
- ☐ Customer reports
- ☐ Internal testing

Alert owner:
Backup contact:

2) Triage (Decide severity and urgency)
For each vulnerability, record:

- Affected component and version(s):
- Is it used in our product? (Yes/No)
- Exposure context: Internal / Limited / Public
- Functional criticality: Low / Medium / High
- Exploitability indicators (if known):
- Proposed response priority: Low / Medium / High

Decision rule (simple):

- High criticality + Public exposure → treat as urgent
- Medium risk → planned fix
- Low risk → track and fix in next planned cycle
-

3) Remediation (Fix and verify)

- Action taken (tick):
  - ☐ Upgrade to fixed version
  - ☐ Apply patch
  - ☐ Mitigation / configuration change
  - ☐ Temporary workaround
- Owner:
- Target date: Validation performed (e.g., testing / regression / deployment verification):

4) Release and communication

- Release notes updated (Yes/No)
- SBOM updated (Yes/No
- Customer/internal communication required? (Yes/No)
  - If yes: channel and message owner:

5) Close-out and learning
Closure date:
What worked/what to improve next time (1-2 bullets):

7.5 A1.5 Dependency Update Policy (Short Form)

Scope
Applies to third-party dependencies, including FOSS components.

Principal

Dependencies are maintained in a risk-based and lifecycle oriented manner.Update actions are proportional to criticality, exposure and impact.

Update cadence

- Routine dependency review: (e.g., monthly/quarterly)
- Security updates: applied based on risk classification
    - High risk: as soon as possible
    - Medium risk: planned fix within next release cycle
    - Low Risk:tracked and resolved during normal maintenance

Decision ownership

- Dependency upgrade approval owner:
- Emergency security fic approval owner:

End-of-life dependencies

If a dependency becomes unsupported or end-of-life:

- SME records the risk
- Evaluates alternative
- Plans migration or compensating controls
- Document the decision and timeline

7.6 A1.6 SBOM Checklist for SMEs (What to capture, when)

An SBOM supports transparency and traceability for products using FOSS components. SMEs can adopt SBOM practices progressively.

What to capture (minimum)

- Product name + version (release identifier)
- Component name + version
- Direct and transitive dependencies (as available)
- License information (where feasible)
- Unique identifiers (when available, e.g., package URL / hashes)

When to generate or update the SBOM

- Minimum:
    - For major releases (e.g., version 1.0, 1.1, 2.0)
- Recommended:
    - For every release that changes dependencies
- Always update when:
    - A security patch requires a dependency upgrade
    - A critical/high exposure component changes version
    - A vulnerability remediation is implemented

SBOM Maturity Levels (Proportional Implementation)

Level 1 – Foundational

- Structured dependency list
- Major release updates
- Linked to release documentation

Level 2 - Structured

- Automated SBOM generation (where possible)
- Direct + transitive dependencies
- License capture
- SBOM stored with release artifacts

Level 3 - Advanced

- CI/CD-integrated SBOM generation
- Continuous dependency monitoring
- Hash verification
- Historical SBOM version tracking
- Rapid impact analysis for CVEs

8 Annex 2 – Aggregated Findings from the OCCTET CRA Self-Assessment Platform

This annex summarises anonymised, aggregated results from voluntary SME assessments conducted through the OCCTET CRA Self-Assessment Platform.
No identifiable information is included.

8.1 A2.1 Platform Overview

Indicator	Value
Register Companies	48
Countries Represented	22
CRA Applicability (Yes)	20
CRA Applicability (NO)	2
Overall Average Maturity Score	1.66

8.2 A2.2 CRA Awareness Distribution

Awareness Level	Count
Not Familiar	2
Somewhat familiar	14
Very Familiar	6

8.3 A2.3 Domain Maturity Scores (Average)

Domain	Average Score
Confidentiality	2.16
Resilience	2.06
Network Security	1.90
Protection from Unauthorized Access	1.76
Integrity	1.65
Vulnerability Management & Disclosure	1.63

Secure Software Development Lifecycle	1.54
Release Requirements	1.48
Logging & Detection	1.34
Risk Management & Governance	1.11

Observed Trends

- Governance and documentation practices are the least mature areas.
- Release discipline and structured SDLC processes require strengthening.
- Technical controls are more mature than lifecycle governance.

9 ACRONYMS AND ABBREVIATION

CRA — Cyber Resilience Act
ENISA — European Union Agency for Cybersecurity
EU — European Union
GDPR — General Data Protection Regulation
FOSS — Free and Open-Source Software
AI — Artificial Intelligence
IPR — Intellectual Property Rights
KPI — Key Performance Indicator
PII — Personally Identifiable Information
RBAC — Role-Based Access Control
SBOM — Software Bill of Materials
SAST — Static Application Security Testing
DAST — Dynamic Application Security Testing
MFA — Multi-Factor Authentication
SOC — Security Operations Centre
IR — Incident Response
API — Application Programming Interface
TLS — Transport Layer Security
ISO — International Organization for Standardization
IEC — International Electrotechnical Commission
ETSI — European Telecommunications Standards Institute
SUS — System Usability Scale
TRL — Technology Readiness Level
WP — Work Package
DoA — Description of Action
BSI - German Federal Office for Information Security

10 BIBLIOGRAPHY

1. **TR-03185-2 Secure Software Lifecycle for Open Source Software** [REF - https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03185/BSI-TR-03185-2.html
2. **White Paper on open source software steward and cra:** https://orcwg.org/files/cra/resources/white-paper-on-open-source-software-stewards-and-cra.pdf
3. **Regulation (EU) 2024/2847 of the European Parliament and of the Council** of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act), Official Journal of the European Union, L, 2024.
4. **European Union Agency for Cybersecurity (ENISA**), Good Practices for Security of Products with Digital Elements, ENISA, 2023.
5. **European Union Agency for Cybersecurity (ENISA**), Guidelines for Coordinated Vulnerability Disclosure, ENISA, 2022.
6. **ISO/IEC 27001:2022**, Information security, cybersecurity and privacy protection — Information security management systems — Requirements, International Organization for Standardization / International Electrotechnical Commission, Geneva.
7. **ISO/IEC 62443-4-1:2018**, Security for industrial automation and control systems — Secure product development lifecycle requirements, ISO/IEC.
8. **ETSI EN 303 645 V2.1.1** (2020), Cyber Security for Consumer Internet of Things: Baseline Requirements, European Telecommunications Standards Institute.
9. **Directive (EU) 2022/2555** (NIS2 Directive) of the European Parliament and of the Council on measures for a high common level of cybersecurity across the Union, Official Journal of the European Union, L 333, 27.12.2022.
10. **European Commission**, The Cyber Resilience Act — Questions & Answers, European Commission, 2024.
11. **European Commission**, DIGITAL Europe Programme – Model Grant Agreement, European Commission, 2024.
12. **OCCTET Project Consortium**, Description of Action (DoA), Grant Agreement No. 101190474, 2024.
13. **OCCTET Project Consortium**, CRA SME requirements and self-assessment checklists (D1.2), 2025.

Note: All regulatory and standards references correspond to the versions in force at the time of deliverable submission (October 2025). Future CRA guidance updates will be reflected in subsequent project releases.