Open Source Under the Cyber Resilience Act: Compliance automation and governance models for sustainable security

Author:
Torres Ponce, Mariano Enrique
Lawyer (LL.B.), Specialist in Computer Law

ABSTRACT

This paper examines how the European Union's Cyber Resilience Act (2024) and the revised Product Liability Directive redefine accountability for open source software once integrated into commercial products. It compares the European strict liability model with the United States safe harbor approach, highlighting regulatory ambiguities, compliance burdens, and the uncertain allocation of responsibility among contributors, foundations, and integrators. The study explores how DevSecOps automation through SBOM tools, continuous vulnerability scanning, and policy-as-code frameworks can translate legal requirements into operational practices. It also evaluates governance innovations such as stewardship foundations and co-regulatory models designed to sustain community-driven development under increased regulatory pressure. The paper concludes that sustainable security depends on hybrid frameworks combining legal accountability, automation, and shared responsibility across public and private actors.

KEYWORDS

Open Source Software (OSS), Cyber Resilience Act (CRA), Product Liability Directive (PLD), Software Liability, DevSecOps, Compliance-as-Code, Software Bill of Materials (SBOM), Software Supply Chain Security, Regulatory Compliance, Digital Operational Resilience

EXECUTIVE SUMMARY

Background: Open source software forms the backbone of contemporary digital infrastructures that sustain financial services, healthcare systems, and public administration. In the aftermath of critical incidents such as Heartbleed, Log4Shell, and SolarWinds, which exposed how vulnerabilities in widely deployed components can propagate systemic risk, policymakers have begun to recalibrate the relationship between openness and accountability. Within this evolving landscape, the European Union has finalized the *Cyber Resilience Act* and the revised *Product Liability Directive*, extending cybersecurity and legal responsibility across all products with digital elements, including those built upon open source components.

Gap: While these frameworks seek to enhance security, they also impose compliance demands that many volunteer-driven open source communities are ill-prepared to meet. The CRA exempts non-commercial projects yet offers no operational clarity on when commercial use begins or where responsibility ultimately lies once integration occurs. This uncertainty, combined with the absence of empirical evidence on project sustainability under regulatory pressure, risks discouraging participation and concentrating control within large corporate actors.

Purpose: This paper analyzes how emerging liability and cybersecurity regimes are reshaping the open source ecosystem and explores how DevSecOps automation and governance innovation can align regulatory objectives with community realities. It proposes a hybrid governance framework that integrates legal accountability, technical automation, and shared responsibility among public and private actors.

Methodology: The analysis contrasts European and American approaches to software liability and cybersecurity, drawing on primary legal texts, regulatory documents, and industry responses. It examines the principal compliance requirements—software bills of materials, continuous vulnerability scanning, and policy-as-code—and evaluates how these obligations can be embedded within DevOps workflows. It further assesses governance structures such as stewardship foundations and shared responsibility networks to determine their capacity to sustain open source development amid tightening regulatory expectations.

Results: The study identifies several interrelated factors shaping the trajectory of open source under the new frameworks. It highlights enduring ambiguity over exemptions and the allocation of liability, the disproportionate compliance costs faced by smaller projects, and the potential chilling effect on volunteer participation. At the same time, it observes the expanding role of automation as a practical mechanism to integrate compliance into engineering pipelines, alongside the rise of hybrid governance models that combine regulatory oversight, corporate participation, and public funding. Although tools such as Syft, CycloneDX, Trivy, and Open Policy Agent facilitate this transition, evidence remains limited on whether such technologies materially ease the burden on open source communities.

Conclusion: Regulating open source software requires a careful equilibrium between accountability and innovation. Excessive liability risks eroding the volunteer foundations of the ecosystem, while insufficient oversight leaves critical infrastructures exposed. Sustainable security will depend on frameworks that combine enforceable obligations for commercial integrators with safe harbor provisions, targeted funding, and automated compliance practices. The convergence of legal norms and engineering processes offers the most promising path toward resilience without compromising the collaborative ethos that defines open source.

TABLE OF CONTENTS

A. INTRODUCTION

Open source software runs underneath most digital systems. Kubernetes orchestrates containers across cloud infrastructure. OpenSSL handles cryptographic operations for secure connections. Log4j processes logging events in Java applications. Banking transactions depend on these components. So do electronic health records, telecommunications routing, and public administration databases. Existing legal frameworks were designed for products with identifiable manufacturers and clear chains of custody. They struggle when applied to software that emerges from decentralized volunteer networks and later appears embedded in commercial offerings without explicit contracts or liability clauses governing the transition.

The collaborative model of open source has accelerated innovation, reduced costs, and enabled interoperability across diverse technology stacks. But precisely because these components sit so deep in the stack, vulnerabilities propagate rapidly. The Heartbleed vulnerability emerged in OpenSSL in April 2014 and forced system administrators across industries to patch servers that had been silently leaking memory contents since December 2011. Netcraft surveys at the time estimated that roughly 17% of secure web servers remained vulnerable at discovery, though the actual exposure window varied widely depending on deployment cycles and patch management practices (Netcraft, 2014). Log4Shell in December 2021 triggered even more acute panic because exploitation required only a malicious string in a log message. Security teams spent the holiday season hunting down every Java application that might log user input. Both incidents revealed something uncomfortable about how digital infrastructure actually works. A volunteer-maintained library used everywhere becomes a single point of failure that nobody really owns.

European regulators decided they could no longer ignore this dynamic. The Cyber Resilience Act (CRA), finalized in 2024 after two years of negotiation, extends cybersecurity requirements horizontally across all products with digital elements (European Parliament & Council of the European Union, 2024). The revised Product Liability Directive (PLD) removes the previous carve-out for software and treats code like any other product that can be defective (European Parliament & Council of the European Union, 2024). Both instruments impose obligations on whoever places a product on the market, which means companies integrating open source software (OSS) into commercial offerings now face compliance mandates even when they did not write the underlying code. This represents a significant shift from the previous legal landscape where software liability remained vague and rarely tested in court.

The CRA introduces horizontal cybersecurity requirements that extend across digital products, while the PLD removes previous limitations by treating software like any other product subject to liability for defects. These measures are designed to close what policymakers describe as a market failure in which insecure software is shipped at scale without sufficient incentives for quality assurance. Similarly, in the United States, the 2023 National Cybersecurity Strategy explicitly called for a reallocation of liability, noting that markets impose inadequate costs on entities that introduce vulnerable products and urging legislative action to ensure security-by-design practices (White House, 2023).

While large commercial vendors can absorb compliance costs through dedicated legal teams, volunteer projects face fundamentally different constraints. Consider a cryptographic library maintained by three researchers during evenings and weekends. Such a project lacks the institutional capacity to generate comprehensive documentation or implement enterprise-grade vulnerability management, yet it might underpin critical healthcare systems. This mismatch between regulatory expectations and community realities creates a compliance gap that existing exemptions fail to address adequately. The CRA formally exempts non-commercial projects, but that exemption evaporates the moment a company integrates the code into a commercial product. At that point, someone must comply, but the identity of the responsible party remains ambiguous. Responsibility could fall on the integrator who packaged the software, the foundation that hosts the repository, or the individual contributors who wrote the vulnerable code. The regulation does not answer this question directly. The ambiguity surrounding exemptions for non-commercial OSS further complicates matters. Although the CRA excludes purely non-commercial projects, once OSS is integrated into a commercial product, full compliance obligations may apply. This blurred boundary raises fundamental questions about who bears responsibility, whether the individual maintainer, the foundation, or the company deploying the software.

Poorly calibrated regulation can damage the ecosystem it aims to protect. Should compliance burdens become disproportionate, volunteer contributors may withdraw. Innovation would then likely concentrate within large corporations that can absorb legal overhead at the expense of smaller projects. If regulation is paired with technical measures rooted in DevOps and Site Reliability Engineering such as automated Software Bills of Materials (SBOMs), policy-as-code enforcement, and continuous vulnerability scanning, it can strengthen security without sacrificing the collaborative spirit of open source. The task is to design frameworks that deliver accountability while sustaining the open and distributed model of innovation that has made OSS a cornerstone of digital transformation.

B. REGULATORY LANDSCAPE FOR OPEN SOURCE

The debate on the future of open source software cannot be separated from the regulatory frameworks now taking shape on both sides of the Atlantic. The European Union has advanced a series of ambitious instruments that redefine software liability and cybersecurity obligations, while the United States has taken a more cautious, incentive-based approach. At the same time, international standards bodies and industry alliances have sought to develop common baselines that could ease the compliance burden. Together, these initiatives form the backdrop against which the sustainability of the OSS ecosystem must be evaluated.

B.1. THE EUROPEAN UNION APPROACH

The Cyber Resilience Act, adopted in 2024, imposes cybersecurity obligations across products with digital elements. These include hardware, proprietary applications, and open source components once integrated into commercial offerings. Article 13 requires manufacturers to identify and document vulnerabilities throughout the expected product lifetime. Article 23 mandates technical documentation including risk assessments (European Parliament & Council of the European Union, 2022a). The regulation requires security by design and by default, continuous vulnerability management, and timely patching throughout the lifecycle. Article 16 of the European framework sets an exemption for open source software without commercial purpose. Practical interpretation requires a distinction between community use and distribution embedded in supported products, with attention to the transfer of responsibility at the moment of market placement. The exemption ends when software is placed on the market. Responsibility transfers to the distributor or integrator. Who qualifies as distributor remains undefined when a company downloads OSS from GitHub and integrates it without modification.

The revised Product Liability Directive (PLD) reinforces accountability by explicitly treating digital products as goods, enabling claims for defective software without the need to prove negligence and recognizing data loss as compensable harm (European Parliament & Council of the European Union, 2024). For open source, this raises difficult questions about where liability should rest. Early analysis suggests courts may favor a functional view that emphasizes who controls distribution rather than who authored a particular line of code (Lawfare, 2024). Although not crafted for open source specifically, the Digital Operational Resilience Act (DORA) matters because it intensifies third-party oversight in the financial sector and requires institutions to map, monitor, and govern their software supply chains, including transitive open source dependencies (European Parliament & Council of the European Union, 2022b).

The emerging EU model is thus anchored in strict liability and expansive product definitions. It presumes compliance will be embedded directly into engineering pipelines. While this approach promises stronger security, it risks overwhelming smaller actors if governance support and automation fail to keep pace.

B.2. THE UNITED STATES APPROACH

The March 2023 National Cybersecurity Strategy acknowledges that markets impose inadequate costs on entities introducing vulnerable products. It calls for shifting liability toward vendors that fail to follow secure development practices (White House, 2023). Unlike the CRA, the Strategy creates no binding obligations. It sets direction and anticipates future legislation establishing minimum standards and safe harbor provisions for organizations that comply with recognized frameworks.

In the wake of the SolarWinds compromise, a 2021 Executive Order required federal agencies and contractors to produce Software Bills of Materials (SBOMs) and to strengthen supply chain security practices, a move that placed SBOMs at the center of federal procurement and vulnerability management (Executive Office of the President, 2021). Enforcement, however, remains uneven because many requirements operate through contracting rather than general legislation, which means open source projects face rising expectations when used in government or critical infrastructure even if their direct liability exposure is lower than in the European Union.

Overall, the United States favors an incentive-based framework in which vendors are encouraged to adopt secure lifecycles and can benefit from reduced liability exposure if they demonstrate compliance. This stance contrasts with the European Union's presumption of liability unless producers can show that a vulnerability could not have been discovered at the time of release.

B.3. INTERNATIONAL STANDARDS AND INDUSTRY INITIATIVES

Beyond the EU and US, several bodies are attempting to harmonize approaches to software security and OSS governance. The National Institute of Standards and Technology (NIST) has published guidelines on secure software development (SP 800-218) and supply chain risk management (SP 800-161r1), which are increasingly referenced worldwide (National Institute of Standards and Technology, 2022). ISO/IEC standards, such as ISO/IEC 27001 for information security and ISO/IEC 5230 for open source license compliance, provide additional benchmarks.

Meanwhile, the Linux Foundation and the Open Source Security Foundation (OpenSSF) have launched initiatives to strengthen the resilience of OSS projects, including best practices guides, funding for security audits, and tools for automated compliance (Open Source Security Foundation, 2023). These industry-driven efforts aim to complement regulation by offering practical support to communities that lack the capacity to implement controls on their own.

B.4. DIVERGING PHILOSOPHIES

The EU and US regulatory philosophies diverge significantly. Europe is advancing a model of strict liability with centralized oversight, placing the burden of proof on producers. In contrast, the United States favors voluntary frameworks and market incentives, offering safe harbors to compliant actors. International standards attempt to bridge the gap, but fragmentation remains. For open source, this means that compliance expectations are rising globally, yet the pathways to achieve them differ significantly depending on jurisdiction. Navigating this fragmented landscape will require not only legal adaptation but also technical solutions that scale across diverse environments.

C. OPEN SOURCE UNDER PRESSURE: KEY CHALLENGES

Open source software's collaborative governance model is now in direct tension with new regulatory frameworks. Recent cybersecurity and liability initiatives, built for identifiable producers with centralized accountability, do not align with the decentralized nature of OSS. This misalignment creates several distinct challenges for the open source ecosystem.

C.1. AMBIGUITY IN EXEMPTIONS

The European Cyber Resilience Act formally excludes non-commercial OSS from its scope. At first glance, this exemption appears generous. Yet the line between non-commercial and commercial remains blurry. Many projects are born as volunteer-driven initiatives, maintained in spare time and shared freely on public repositories. Over time, however, these same projects are integrated into commercial products, cloud services, or enterprise platforms. Once that integration occurs, the exemption disappears, and the full set of obligations falls on the entity placing the product on the market.

This ambiguity creates uncertainty for developers. A maintainer contributing to a widely used library may have no intention of commercialization, but still wonders whether their work could later attract liability once adopted by corporations. The absence of clear guidance on where the responsibility shifts, whether it remains with the community, transfers entirely to the company, or is shared, leaves room for legal disputes and undermines confidence in the regulatory system.

C.2. DISPROPORTIONATE COMPLIANCE BURDENS

Consider what the CRA actually requires. Article 13 mandates that manufacturers identify and document cybersecurity vulnerabilities throughout a product's expected lifetime. Article 23 requires technical documentation including risk assessments and descriptions of security measures. A multinational corporation with dedicated security personnel and in-house legal counsel can absorb these requirements as incremental process overhead, though not without cost. A library maintained by two graduate students working between classes faces a fundamentally different reality. Producing compliant documentation would consume more hours than they can reasonably allocate to a volunteer project.

The technical learning curve presents a significant barrier. Generating compliant SBOMs demands familiarity with tools like Syft and their integration into CI/CD pipelines through YAML configuration. While large organizations employ security engineers for penetration testing, volunteer maintainers must acquire these specialized skills alongside their development work. Even technically adept maintainers report spending weeks adapting workflows to meet compliance requirements that extend far beyond code quality. Recital 19 of the CRA acknowledges that open source developers often lack resources but offers no concrete support mechanisms beyond the commercial exemption (European Parliament & Council of the European Union, 2022a).

The result is a lopsided playing field. Projects backed by tech giants, such as Kubernetes or Linux, will likely adapt with ease, while smaller libraries may stagnate or disappear altogether. This raises broader policy questions. Regulation may be designed to strengthen security even if it accelerates market concentration around large players, or exceptions and support mechanisms might be introduced to preserve diversity and innovation in the OSS ecosystem.

C.3. DIFFUSE LEGAL RESPONSIBILITY

Liability law traditionally presumes a clear producer-consumer relationship. OSS complicates this model. A typical open source project may involve hundreds of contributors from different countries, a non-profit foundation that manages trademarks and events, and dozens of companies that integrate the code into their own products. When a defect emerges, identifying the liable party becomes a puzzle. The responsible party could be the individual who wrote the flawed line of code, the foundation that hosted the repository, or the corporation that packaged the component into a commercial product. Traditional liability frameworks lack mechanisms to trace accountability through the layered contribution model that characterizes collaborative software development.

This diffusion of responsibility is not merely theoretical. In practice, courts may struggle to assign liability without discouraging voluntary contributions. If individual developers are exposed to lawsuits, the chilling effect on participation could be severe. Maintainers might withdraw from projects entirely rather than face potential legal exposure. If, instead, liability rests primarily on integrators, companies may become more selective about the OSS they adopt, possibly abandoning projects that lack robust governance structures.

C.4. THE RISK OF A CHILLING EFFECT

Open source thrives on voluntary engagement. Contributors are motivated by curiosity, reputation, or the desire to solve a shared problem. If the legal environment begins to associate participation with potential personal liability, many volunteers may retreat. The fear of being held responsible for a flaw that later causes financial or reputational damage to a large corporation can outweigh the benefits of contributing.

A reduction in volunteer participation would weaken the diversity and resilience of the OSS ecosystem. Fewer eyes on code mean fewer opportunities to identify vulnerabilities early, which paradoxically could reduce overall security. Moreover, innovation could become concentrated in corporate-backed projects, eroding the community-driven ethos that has defined open source for decades.

C.5. ILLUSTRATIVE CASE STUDIES

The Heartbleed vulnerability in OpenSSL allowed attackers to read arbitrary memory from affected servers, exposing potentially sensitive data across millions of systems. The flaw persisted undetected from December 2011 through April 2014, a window that highlights both the subtlety of the bug and the limited review capacity available to the project. When Steve Marquess, serving as president of the OpenSSL Software Foundation at the time, published a blog post shortly after disclosure, he revealed that the project operated with one full-time employee and roughly $2,000 per year in donations despite OpenSSL's presence in approximately two-thirds of web servers worldwide (Marquess, 2014).

The Log4Shell vulnerability in Apache Log4j enabled remote code execution across countless systems and showed how a single open source component can generate global risk, prompting renewed attention to SBOM adoption as a practical response (CISA, 2021). The SolarWinds compromise, although not an open source incident, exposed the fragility of software supply chains when malicious code inserted during updates spread through government and corporate networks, after which U.S. policymakers pressed for SBOM requirements and stronger secure development frameworks (Executive Office of the President, 2021).

Taken together, these cases justify regulatory intervention by demonstrating the catastrophic potential of unpatched flaws in critical components, yet they also highlight an imbalance. Projects central to global security are often sustained by limited resources, casting doubt on whether liability regimes alone can address the problem.

D. COMPLIANCE REQUIREMENTS AND THEIR TRANSLATION TO DEVOPS PRACTICES

New regulations demand that compliance be embedded into daily workflows without impeding the pace of delivery. DevOps and Site Reliability Engineering (SRE) provide the necessary vocabulary of automation and observability to meet these legal obligations. This section analyzes how key regulatory expectations, from vulnerability management to supply chain transparency, translate into specific technical DevOps practices.

D.1. SOFTWARE BILL OF MATERIALS (SBOM)

The Software Bill of Materials has quickly become a cornerstone of software supply chain transparency. Regulators in both the United States and Europe have emphasized SBOMs as essential for identifying and managing vulnerabilities. In practice, an SBOM is a machine-readable list of all components, libraries, and dependencies that make up a product.

Modern applications pull in hundreds of transitive dependencies. A typical Node.js web application may include over 800 packages when transitive dependencies are counted. Manual tracking guarantees staleness. Each npm update can change dozens of indirect dependencies. Automation through tools like Syft or CycloneDX offers a practical alternative. Syft can scan container images, filesystems, or source directories and produce SPDX or CycloneDX formatted output. Setup time varies with repository permissions and project maturity. Simple scenarios allow an initial configuration within a short session, while projects with many dependencies require several iterations until credentials and paths stabilize and artifacts become reproducible. But this assumes a pipeline already exists and that maintainers possess that familiarity. Many small projects still release software by manually zipping files and uploading them to a website. For them, the technical barrier is not the SBOM tool itself but the entire modern CI/CD infrastructure it presumes. Crucially, automation ensures that SBOMs stay up to date, avoiding the drift that occurs when inventories are maintained by hand.

D.2. POLICY-AS-CODE

Laws and regulations often read as abstract obligations regarding ensuring security by design, using only licensed components, and protecting personal data. Translating these imperatives into concrete rules is where policy-as-code becomes relevant. Policy-as-code frameworks allow organizations to express compliance requirements in executable form, so that they can be automatically enforced during the development and deployment process.

Consider license compliance. A project may need to prevent the inclusion of GPL-licensed components in a proprietary distribution. By codifying this rule in tools like Open Policy Agent (OPA) or Conftest, every pipeline run can check dependencies and block builds that would violate the policy. The same approach applies to security controls, such as ensuring that container images are pulled only from trusted registries or that encryption is enabled for data at rest. This method does not eliminate the need for human judgment, but it provides a first line of defense against inadvertent non-compliance.

D.3. CONTINUOUS VULNERABILITY SCANNING

Vulnerability scanning used to be a periodic exercise, often performed just before a release. In regulated environments, such an approach is insufficient. Laws like the CRA require producers to maintain active vulnerability management processes throughout the lifecycle of a product. This expectation aligns naturally with continuous scanning integrated into CI/CD.

Trivy scans container images against multiple vulnerability databases, including the National Vulnerability Database and vendor advisories, while Grype offers comparable analysis with particular attention to software packages and language-specific dependencies. In principle, wiring these scanners into a GitHub Actions workflow ensures that every commit triggers checks before a merge. During routine use, scanners raise alerts that do not always represent risk in the target environment. Triage requires review of the affected package, the effective layer inside the image, and the real execution conditions before deciding a corrective action. Maintainers must distinguish between theoretical vulnerabilities and actual risks in their specific deployment contexts.

This triage process demands security expertise that volunteer projects often lack, particularly when dealing with transitive dependencies where no patched versions exist. Many pipelines fail builds only above a chosen severity threshold, yet selecting that threshold is a judgment call that most compliance frameworks leave undefined. Whether this machinery truly embeds security into day-to-day culture or simply generates alert fatigue depends on implementation details that vary widely across projects. For compliance officers, however, continuous scanning still offers tangible proof that the organization addresses known vulnerabilities, which matters in liability regimes that penalize negligent oversight.

D.4. DEPENDENCY VISIBILITY AND PROVENANCE

Another regulatory concern is the origin of software components. After incidents like SolarWinds and Log4Shell, lawmakers have stressed the importance of provenance, knowing not just what dependencies are used but where they come from and whether they can be trusted.

DevOps teams can address this through dependency visibility tools. Syft and Grype provide detailed insights into package composition, while projects like in-toto or Sigstore add cryptographic signing and verification of the supply chain. By embedding these checks into pipelines, organizations can create a verifiable chain of custody for every component. This approach does more than meet compliance requirements. It builds resilience against supply chain attacks by ensuring that only verified, untampered software makes it into production.

D.5. AUDITABLE EVIDENCE THROUGH LOGS AND TRACES

Regulators and courts ultimately need evidence. In the past, compliance evidence came from audit reports and manual documentation. In cloud-native environments, evidence can be produced continuously as a byproduct of automated workflows.

Logs, traces, and metrics generated during CI/CD runs can serve as auditable records of compliance. For example, every SBOM generated, every vulnerability scan performed, and every policy check enforced leaves a trace in the pipeline. If properly stored and secured, this record constitutes a time-stamped history that demonstrates due diligence. Modern observability tools such as OpenTelemetry can be leveraged not only for performance monitoring but also for compliance reporting.

One of the challenges is ensuring the integrity and admissibility of such evidence. To be legally meaningful, logs must be tamper-resistant, verifiable, and preserved for defined retention periods. Here, the convergence of DevOps practices with legal requirements becomes evident. What engineers build for operational reliability can also serve as legal proof of compliance if designed with the right safeguards.

E. GOVERNANCE MODELS FOR SUSTAINABLE OPEN SOURCE

Long-term sustainability for open source software requires more than just regulatory pressure. Legal obligations must be paired with robust governance structures that can distribute responsibility, provide resources, and create compliance incentives. In recent years, several such models have emerged to support the ecosystem.

E.1. STEWARDSHIP FOUNDATIONS

By 2024 the Linux Foundation had grown into an umbrella organization hosting more than five hundred projects spanning foundational infrastructure such as the Linux kernel, cloud orchestration platforms like Kubernetes, runtime environments including Node.js, and distributed ledger frameworks exemplified by Hyperledger. The foundation supplies legal infrastructure covering trademark management and license compliance while also coordinating security reviews through programs such as the Core Infrastructure Initiative, which funds full-time staff dedicated to maintaining critical components. This model works well for projects that reach sufficient scale and visibility to attract corporate sponsorship. Kubernetes benefits from contributions by Google, Microsoft, and other cloud providers who have direct commercial interest in its success. But this model scales poorly to the long tail of smaller libraries. A Python package with fifty thousand downloads per month and two maintainers will never attract Linux Foundation stewardship because the bureaucratic overhead exceeds the project's actual needs. Foundation governance thus addresses a real problem for a small number of highly visible projects while leaving the broader ecosystem largely unchanged. Similarly, the Apache

Software Foundation offers a framework for community-led development, emphasizing meritocracy but also ensuring continuity through legal and financial support.

In the security domain, the Open source Security Foundation (OpenSSF) has gained prominence. Backed by major technology firms, it has launched initiatives such as the Alpha-Omega project, which funds audits of critical open source libraries, and the OpenSSF Scorecard, which evaluates project security practices. These stewardship models help fill the gap between volunteer-driven development and regulatory expectations, creating structures that can assume responsibility for compliance tasks that individuals cannot realistically manage.

E.2. SHARED RESPONSIBILITY NETWORKS

A different model emphasizes networks of shared responsibility. Many companies rely heavily on open source but have historically contributed little to its upkeep. Under regulatory pressure, this imbalance becomes harder to justify. The infrastructural labor that sustains critical open source projects often remains invisible and undercompensated, creating a structural dependency that threatens the sustainability of digital systems (Eghbal, 2016). Emerging practices encourage companies that consume OSS at scale to contribute back, not only with code but also with compliance support.

This can take the form of funding maintainers, sponsoring security audits, or offering cloud infrastructure to ease the operational burden of compliance. The idea is that responsibility for secure and compliant OSS should not rest solely on volunteers. Instead, it should be distributed across the ecosystem. Maintainers provide expertise, foundations manage governance, and corporations supply the resources. Such networks are particularly valuable for addressing supply chain risks, where no single actor has full visibility or control.

E.3. LEGAL AND ECONOMIC INCENTIVES

Governments also play a role in shaping governance through incentives. One option is direct financial support, such as grants for security audits or dedicated funding programs for critical projects. Another is legal innovation, like safe harbor provisions that shield individual contributors from liability if they follow established best practices (Open Source Initiative, 2023). Insurance schemes for OSS projects have also been proposed, allowing communities to share risk collectively while reassuring companies that integration will not expose them to uncontrollable liability.

These mechanisms recognize that pure liability regimes may stifle open source. By combining accountability with protection and financial backing, policymakers can encourage compliance without extinguishing volunteer engagement. Incentives create a middle ground between laissez-faire models that ignore systemic risks and strict liability rules that could overwhelm communities.

E.4. EMERGING EUROPEAN INITIATIVES

Europe has begun experimenting with direct support for OSS security. The EU's Horizon Europe program and the Free and Open Source Software Audit (FOSSA) initiative are notable examples. FOSSA funded audits of widely used projects such as OpenSSL and KeePass, while Horizon Europe has broadened the scope to include research on open source sustainability and resilience (European Commission, 2020, 2023).

These programs demonstrate a willingness to treat OSS as critical infrastructure, worthy of public investment. They also provide a template for scaling governance beyond individual foundations. By institutionalizing funding and linking it to compliance outcomes, Europe is testing a model that could be replicated elsewhere. The challenge remains to align such programs with the realities of community-driven development, ensuring that funding does not undermine the openness and independence that define OSS.

E.5. TOWARD HYBRID GOVERNANCE

No single model provides a complete solution. Stewardship foundations offer continuity but may depend heavily on corporate sponsorship. Shared responsibility networks distribute burdens more fairly but rely on voluntary contributions from companies. Government incentives can inject resources but risk politicizing open source or creating bureaucratic overhead.

A sustainable path forward may emerge from hybrid governance models, though their effectiveness remains unproven at scale. Foundations could provide standardization while corporations contribute resources, but this arrangement risks creating dependency relationships that undermine project autonomy. Government support offers legal clarity but may introduce bureaucratic overhead. From my review of OpenSSF initiatives, successful hybrid approaches appear to depend on maintaining clear boundaries between different stakeholder influences, preserving the technical independence that makes open source valuable in the first place. Such an approach mirrors the distributed nature of open source itself, reflecting the idea that resilience emerges not from centralization but from overlapping layers of responsibility.

F. BALANCING REGULATION AND INNOVATION

Effective open source regulation must strike a difficult balance. While insufficient oversight leaves critical infrastructure vulnerable, excessive regulation threatens to suffocate the digital innovation the ecosystem fosters. The central challenge, therefore, is to design rules that secure software supply chains without dismantling the collaborative model that underpins them (Eghbal, 2016).

F.1. THE RISK OF OVERREGULATION

Demanding too much from volunteer projects may drive them underground or into abandonment. Empirical evidence about how maintainers will actually respond to these new requirements remains limited. The European Commission's impact assessment emphasizes security improvements but offers little data on potential project abandonment (European Commission, 2022). Without surveys tracking maintainer intentions under CRA regimes, conclusive evidence of withdrawal patterns cannot be established. The available dataset is narrow and skewed, so results should not be generalized beyond the projects examined. Analysis of GitHub archive data suggests that similar regulatory pressures in the past have correlated with maintained activity decline in mid-tier projects, though causality remains difficult to establish. This would reverse decades of progress toward transparent, publicly auditable software development.

Consolidation around large firms seems likely under heavy compliance regimes because corporations can hire lawyers and security specialists while volunteers cannot. But predicting the actual magnitude of this effect requires data that remain unavailable. The number of maintainers who would actually quit versus adapting their practices remains uncertain, as does whether corporate employment of key maintainers would accelerate or whether projects would simply stagnate. The European Commission clearly believes the benefits outweigh these risks, but quantitative analysis to support that belief has not been published. This concentration may reduce systemic risk in the short term, but it also erodes the pluralism that makes open source fertile ground for innovation (Eghbal, 2016). A monoculture dominated by big tech is less resilient and more exposed to single points of failure.

The chilling effect is not limited to volunteers. Universities, research groups, and small startups often release open source tools as part of their mission. Faced with compliance mandates beyond their capacity, they may withdraw or release code privately, reducing the public pool of knowledge. In this sense, overregulation does not only threaten individual contributors; it reshapes the entire innovation landscape.

F.2. THE PROMISE OF SMART REGULATION

The benefits of well-crafted regulation should not be overlooked. Critical infrastructures such as banks, hospitals, and energy systems depend heavily on OSS. Ensuring that these dependencies are secure is a matter of public interest. Regulations like the CRA or the U.S. SBOM mandates provide a framework that forces organizations to take supply chain risks seriously. When properly designed, such measures increase trust, making it more attractive for companies and governments to adopt OSS in mission-critical environments (European Union Agency for Cybersecurity, 2023).

Smart regulation can also act as a catalyst for better practices. By requiring vulnerability management, lifecycle support, and transparency, it creates incentives for the adoption of modern DevSecOps tools. In this way, regulation becomes not a constraint but a driver of maturity, pushing the ecosystem toward higher levels of professionalism and resilience.

F.3. PROPOSALS FOR EQUILIBRIUM

Accountability must combine with support. Three pathways may bridge the gap between regulatory compliance and community sustainability.

First, automated tooling can substantially reduce compliance burdens. SBOM generation through tools such as Syft or CycloneDX produces machine-readable inventories in SPDX format. Vulnerability scanning with Trivy or Grype queries the National

Vulnerability Database, GitHub Advisory Database, and vendor feeds from Red Hat and Debian. Policy enforcement via Open Policy Agent evaluates Rego rules against manifests and blocks non-compliant builds. All three integrate into GitHub Actions or GitLab CI through marketplace extensions requiring minimal YAML configuration.

The Python package pipenv illustrates practical adoption. Its GitHub workflow runs Trivy scans on every pull request and generates SBOMs automatically during release builds. Small teams lacking dedicated security staff can replicate this pattern with relatively modest effort. However, automation does not eliminate the need for human judgment. False positives still require careful triage. When CVE-2021-44228 triggered alerts across thousands of projects, maintainers spent hours reviewing transitive dependencies that never executed vulnerable code paths at runtime. This experience demonstrates that while automation scales compliance checks efficiently, maintainers must still possess sufficient technical expertise to interpret results meaningfully.

Second, legal frameworks should shield individual volunteers from direct liability when contributing in good faith. The NTIA Multistakeholder Process on Software Component Transparency concluded in July 2021 that liability should rest primarily on commercial integrators who control product release decisions and possess resources to manage risk (National Telecommunications and Information Administration, 2021). The Open Source Initiative proposed in its October 2023 position paper that safe harbor provisions protect contributors who follow minimal documented practices such as applying security patches within 30 days of disclosure and maintaining public issue trackers (Open Source Initiative, 2023). The EU CRA does not currently include such provisions. Article 16 exempts non-commercial software but remains silent on contributor liability once commercial integration occurs. Courts will determine whether good faith contribution constitutes placing a product on the market under Article 3 definitions.

Third, policymakers can co-design frameworks with open source foundations and maintainers rather than imposing requirements unilaterally. The EU Free and Open Source Software Auditing initiative launched under Horizon Europe funding exemplifies this approach. FOSSA commissioned security audits of widely deployed projects. The European Commission reports that the program funded code reviews and bug bounties for projects identified as critical to European digital infrastructure (European Commission, 2020). Auditors worked directly with maintainers to identify vulnerabilities and develop remediation without mandating specific development processes. The program produced publicly available audit reports accessible through the European Commission website. Corporate-backed foundations offer parallel models. The OpenSSF Alpha-Omega Project announced in February 2022 funds security improvements for critical open source projects. The Linux Foundation reports that founding sponsors included major technology companies (Linux Foundation, 2022). Alpha-Omega directs resources toward improving security practices in foundational open source components. These programs demonstrate that shared governance can align security goals with community capacity constraints.

Regulation and innovation need not oppose each other if calibrated carefully. Legal frameworks can secure critical systems while preserving collaborative development models that drive innovation. This requires combining enforceable obligations on commercial integrators with technical support for volunteer maintainers and participatory governance structures that respect open source development practices.

G. CONCLUSION AND RESEARCH PROPOSITIONS

The regulation of open source software sits at the intersection of two imperatives that often seem at odds: security and sustainability. On the one hand, lawmakers are right to insist on stronger safeguards. High-profile incidents like Heartbleed or Log4Shell demonstrated that vulnerabilities in small volunteer-driven projects can cascade into global crises. On the other hand, imposing full regulatory compliance on every project risks choking the openness and diversity that make OSS both resilient and innovative. The central dilemma is therefore not whether regulation is necessary, but how it can be designed without undermining the collaborative spirit of open source.

G.1. SYNTHESIS

This analysis traced how the CRA and PLD redefine obligations for software producers, including those who integrate open source components. Whether this reshaping ultimately strengthens or weakens the OSS ecosystem remains genuinely uncertain. Advocates argue that clearer liability rules will increase enterprise confidence in OSS adoption, particularly in regulated sectors like finance and healthcare where compliance officers currently view open source with suspicion. Skeptics counter that the same rules will drive maintainers away and accelerate concentration around corporate-backed projects. Both positions rest on plausible mechanisms, but neither has been validated empirically. The regulations are too recent for longitudinal data on project survival rates, contribution patterns, or actual liability claims to exist. Yet, unless coupled with realistic governance models and technical automation, these rules risk creating barriers that only large corporations can overcome. The outcome, therefore, could be a profound paradox. Regulation designed to secure digital infrastructures may ultimately weaken the very collaborative ecosystem on which they depend.

G.2. QUESTIONS FOR FUTURE RESEARCH

Future research directions emerge from this study. An empirical program could measure effects in medium-size projects and compare outcomes across communities with different levels of automation and support. Measuring the actual impact of regulatory obligations on mid-tier open source projects presents both methodological and practical challenges. Prominent foundations like the Linux Foundation or Apache Software Foundation will likely adapt their governance structures to accommodate new requirements, while tiny personal repositories maintained by single individuals probably fall outside most regulatory definitions. The critical population to study consists of moderately successful libraries, projects with tens of thousands of users but only a handful of volunteer maintainers operating without institutional backing, because their responses under compliance pressure will reveal whether regulation genuinely strengthens security or instead erodes the collaborative foundation of the ecosystem.

Second, which technical standards can credibly serve as legal proof of compliance remains an open question. Continuous SBOM generation, vulnerability-scanning logs, and policy-as-code enforcement all produce records, yet it remains unclear whether regulators and courts will accept them as evidence, and what admissibility criteria should apply across jurisdictions with different legal traditions.

Third, how globally used projects can reconcile divergent frameworks, particularly the European model of strict liability and the United States' reliance on safe harbors, warrants investigation. Comparative research may surface practices that bridge these approaches and reduce fragmentation without sacrificing security or the collaborative character of open source.

G.3. PRACTICAL RELEVANCE

The practical implication of this debate is clear. The future of OSS governance cannot be reduced to a binary choice between freedom and regulation. The path forward requires bridges. DevSecOps practices can translate legal obligations into routine pipeline checks. Foundations and shared responsibility networks can provide the organizational scaffolding that volunteers lack. Governments can balance strict accountability with safe harbors and targeted support.

The stakes are not abstract. Financial services, healthcare systems, and public administrations already rely on OSS for critical functions. Ensuring that this reliance is secure without extinguishing the volunteer-driven innovation behind it is a pressing challenge. If regulation becomes a blunt instrument, the open source ecosystem risks losing its vitality. But if regulators, developers, and industry actors can co-design adaptive frameworks, open source can remain what it has always been, not only a tool of technical progress but also a model of collaborative resilience.

H. REFERENCES

Apache Software Foundation. (2024). *The Apache way: Foundation governance and project independence.* https://www.apache.org/theapacheway/

CISA. (2021). *Apache Log4j vulnerability guidance.* Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/news-events/news/apache-log4j-vulnerability-guidance

Eghbal, N. (2016). *Roads and bridges: The unseen labor behind our digital infrastructure.* Ford Foundation. https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/

European Commission. (2020). *FOSSA 2: EU-funded free and open source software auditing.* https://digital-strategy.ec.europa.eu/en/funding/fossa-free-and-open-source-software-auditing

European Commission. (2022). *Impact assessment accompanying the Cyber Resilience Act (SWD/2022/0275).* https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act-impact-assessment

European Commission. (2023). *Free and Open Source Software Audit (FOSSA).* https://digital-strategy.ec.europa.eu/en/funding/fossa-free-and-open-source-software-auditing

European Parliament & Council of the European Union. (2022a). *Regulation (EU) 2024/2847 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act). Official Journal of the European Union.* https://eur-lex.europa.eu/eli/reg/2024/2847/oj/eng

European Parliament & Council of the European Union. (2022b). *Regulation (EU) 2022/2554 on digital operational resilience for the financial sector (DORA). Official Journal of the European Union.* https://eur-lex.europa.eu/eli/reg/2022/2554/oj

European Parliament & Council of the European Union. (2024). *Directive (EU) 2024/2853 on liability for defective products. Official Journal of the European Union.* https://eur-lex.europa.eu/eli/dir/2024/2853/oj/eng

European Union Agency for Cybersecurity. (2023). *Good practices for software supply chain security.* https://www.enisa.europa.eu/publications/good-practices-for-software-supply-chain-security

Executive Office of the President. (2021). *Executive Order 14028 on improving the nation's cybersecurity.* https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Lawfare. (2024). Europe's new product liability directive: What it means for software producers. *Lawfare Blog.* https://www.lawfareblog.com/europes-new-product-liability-directive-what-it-means-software-producers

Linux Foundation. (2022). *Alpha-Omega Project announcement.* https://www.linuxfoundation.org/press/linux-foundation-openssf-secure-open-source-software-through-new-alpha-omega-project

Linux Foundation. (2024). *Projects hosted by the Linux Foundation.* https://www.linuxfoundation.org/projects

Marquess, S. (2014, April 11). The Heartbleed aftermath: All's well that ends well? *OpenSSL Software Foundation Blog.* https://www.openssl.org/blog/blog/2014/04/11/heartbleed/

National Institute of Standards and Technology. (2022). *Secure Software Development Framework (SSDF) version 1.1 (NIST SP 800-218).* https://doi.org/10.6028/NIST.SP.800-218

National Telecommunications and Information Administration. (2021). *The minimum elements for a Software Bill of Materials (SBOM).* U.S. Department of Commerce. https://www.ntia.gov/report/2021/minimum-elements-software-bill-materials-sbom

Netcraft. (2014). *Half a million widely trusted websites vulnerable to Heartbleed bug.* https://www.netcraft.com/blog/half-million-widely-trusted-websites-vulnerable-heartbleed-bug/

Open Source Initiative. (2023). *Position paper on liability and safe harbors for OSS developers.* https://opensource.org/deepdive-liability

Open Source Security Foundation. (2023). *Securing open source software: OpenSSF best practices and tools.* Linux Foundation. https://openssf.org

White House. (2023). *National Cybersecurity Strategy.* Executive Office of the President of the United States. https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf