The Cyber Resilience Act and Open-Source Software: A Fine Balancing Act

by **Mattis van 't Schip** *

Abstract: Open-source software, a type of software that can be publicly accessed, shared, and modified, is an integral part of modern digital infrastructure. Many products, from personal computers to internet-connected devices, run on open-source systems (e.g., Linux). Developers may work voluntarily or for limited compensation on such software. The character of this work, however, does not reduce the impact of cybersecurity incidents within these environments. Proprietary software, meaning software with restrictive license models, regularly implements open-source software: a vulnerability in the open-source software thus directly affects proprietary software too. Recent large-scale vulnerabilities (e.g., Log4j) highlighted this dual nature of open-source software: developers work on projects based on personal passion or ideologies, while the software is often equally as critical as software created and maintained by larger technology enterprises.

The Cyber Resilience Act, the recently proposed European cybersecurity legislation for products, aims to offer a legal response to cybersecurity problems in modern software and hardware. This paper addresses the role of open-source software cybersecurity in the Cyber Resilience Act with specific attention to the difficulties of reconciling cybersecurity responsibilities and open-source products. I show that the Cyber Resilience Act does achieve a balance between regulation for open-source software and advancing cybersecurity, but only through a narrowly applicable and, at times, complex legislative approach.

Keywords: Open-Source Software; Cybersecurity; Cyber Resilience Act

Recommended citation: Mattis van 't Schip, The Cyber Resilience Act and Open-Source Software: A Fine Balancing Act, 16 (2025) JIPITEC 73 para 1.

A. Introduction

1 Behind the facade of giant technology enterprises exists an ecosystem of 'open-source software'. The source code of this type of software is publicly accessible and developers write the code under licenses that allow for use, redistribution, modification, and sharing by third parties. 'Open source' does not merely mean public access to source code. The Open Source Initiative (OSI), a body responsible for the generally accepted definition of 'open source', indicates that the concept holds certain additional criteria.[1] For instance, open-source software licenses should not discriminate based on intended use.[2]

2 Many types of open-source software support today's largest software packages: Linux, an open-source operating system, powers many modern ICT products, from desktop computers to Internet of Things devices; millions of websites rely on Apache, an open-source web server. Open-source software is thus an important cornerstone of the modern digital infrastructure.[3]

3 The advantages of open-source software align with recent regulatory efforts in the EU that aim to curtail the market power of the major digital enterprises. For instance, the Digital Services Act regulates online

* Ph.D. Candidate at the Interdisciplinary Research Hub on Digitalization and Society (iHub), Radboud University. This research is funded through the NWO INTERSCT project [NWA.1160.18.301].

1 Open Source Initiative, 'The Open Source Definition' (22 March 2007) <https://opensource.org/osd> accessed 19 January 2024.

2 <https://opensource.org/licenses/>.

3 Chinmayi Sharma, 'Tragedy of the Digital Commons' (2023) 101 North Carolina Law Review 1129.

platforms (and especially the "very large" online platforms, i.e., the major social media platforms), while the Digital Markets Act imposes responsibilities on "gatekeepers" (e.g., Microsoft, Meta).[4] Open-source software can serve as a transparent, public alternative to these dominant platforms.

4 Like other types of software, open-source software comes with cybersecurity risks.[5] For example, Log4j, a piece of open-source software for logging purposes, suffered a critical vulnerability which allowed hackers to remotely access systems.[6] Some experts held that the vulnerability affected virtually every digital service globally.[7] The Log4j vulnerability was critical because open-source software is often incorporated in larger proprietary software packages; the vulnerability in Log4j thus directly affected numerous other products.[8]

5 In September 2022, the European Commission introduced a new legislative proposal for the cybersecurity of software and hardware products, the Cyber Resilience Act.[9] At the end of 2024, the Act came in effect.[10] The Act applies when manufacturers and/or software developers place software or hardware products on the market of the European Union "in the course of a commercial activity".[11] If they place these products on the market, software developers must implement certain cybersecurity requirements in their product and, in certain cases, follow strict assessment procedures.

6 Although this requirement potentially excludes open-source software, the 'commercial activity' condition offers few assurances, as evident from the legislative discussions surrounding its interpretation.[12] The commerciality of open-source software projects can range from monetising other services on the open-source software platform (e.g., Android) to occasional donations from end users (e.g., hobby projects).[13] The Commission proposal merely mentioned these examples, but did not offer a further clarification of what "supplying in the course of a commercial activity" entails.

7 The text adopted by the Parliament, instead, includes a rather comprehensive set of Recitals, which cover many open-source software development and financing methods. The compromise text therefore exempts nearly every known type of open-source software development from the scope of the Cyber Resilience Act. This exemption helps developers, who do not have to comply with legal burdens for software that they provide openly to the public.

4 Regulation (EU) 2022/2065 of the European Parliament and of the Council of 19 October 2022 on a Single Market For Digital Services and amending Directive 2000/31/EC [2022] OJ L277/1 (Digital Services Act); Regulation (EU) 2022/1925 of the European Parliament and of the Council of 14 September 2022 on contestable and fair markets in the digital sector and amending Directives (EU) 2019/1937 and (EU) 2020/1828 [2022] OJ L265/1 (Digital Markets Act).

5 Jaap-Henk Hoepman and Bart Jacobs, 'Increased Security through Open Source' (2007) 50 Communications of the ACM 79.

6 For an extensive overview, see Raphael Hiesgen and others, 'The Log4j Incident: A Comprehensive Measurement Study of a Critical Vulnerability' [2024] IEEE Transactions on Network and Service Management 1.

7 Sean Lyngaas, 'US Warns Hundreds of Millions of Devices at Risk from Newly Revealed Software Vulnerability' (*CNN*, 13 December 2021) <https://www.cnn.com/2021/12/13/politics/us-warning-software-vulnerability/index.html> accessed 19 January 2024; Ars Technica spoke of 'arguably the most severe vulnerability ever', see Dan Goodin, 'As Log4Shell Wreaks Havoc, Payroll Service Reports Ransomware Attack' (*Ars Technica*, 13 December 2021) <https://arstechnica.com/information-technology/2021/12/as-log4shell-wreaks-havoc-payroll-service-reports-ransomware-attack/> accessed 19 January 2024; Similarly, see the Guardian Associated Press, 'Recently Uncovered Software Flaw "Most Critical Vulnerability of the Last Decade"' *The Guardian* (11 December 2021) <https://www.theguardian.com/technology/2021/dec/10/software-flaw-most-critical-vulnerability-log-4-shell> accessed 19 January 2024.

8 Sharma (n 4) 1131–1133.

9 Proposal for a Regulation of the European Parliament and of the Council on horizontal cybersecurity requirements for products with digital elements and amending Regulation (EU) 2019/1020 COM(2022) 454 final [Cyber Resilience Act].

10 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 and amending Regulations (EU) No 168/2013 and (EU) No 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act) [2024] OJ L (to be published).

11 Art 3(22) CRA.

12 See the calls for support from the open source community when the Cyber Resilience Act proposal was published in, inter alia, Maarten Aertsen, 'Open-Source Software vs. the Proposed Cyber Resilience Act' (*The NLnet Labs Blog*, 14 November 2022) <https://blog.nlnetlabs.nl/open-source-software-vs-the-cyber-resilience-act/> accessed 20 December 2023; Deb Nicholson, 'Python Software Foundation News: The EU's Proposed CRA Law May Have Unintended Consequences for the Python Ecosystem' (*Python Software Foundation News*, 11 April 2023) <https://pyfound.blogspot.com/2023/04/the-eus-proposed-cra-law-may-have.html> accessed 20 December 2023; Simon Phipps, 'What Is the Cyber Resilience Act and Why It's Dangerous for Open Source' (*Voices of Open Source*, 24 January 2023) <https://blog.opensource.org/what-is-the-cyber-resilience-act-and-why-its-important-for-open-source/> accessed 20 December 2023.

13 David A Wheeler, 'F/LOSS Is Commercial Software' [2009] Open Source Business Resource <http://timreview.ca/article/229>.

At the same time, these broad exemptions could undermine the overall aim of the Cyber Resilience Act to improve the state of cybersecurity for software and hardware.

8 This paper analyses the difficulties of reconciling open-source software development with cybersecurity risk management responsibilities. The research question is: To what extent does the Cyber Resilience Act impose responsibilities on open-source software developers that achieve a balance between stimulating open-source software development and, simultaneously, mitigating cybersecurity problems within open-source software?

9 This paper proceeds as follows: Section B summarises the history and meaning of open-source software and its cybersecurity implications. Section C discusses the Cyber Resilience Act, with specific attention to the definition of supplying a product 'in the course of a commercial activity' for open-source software products. Section D highlights, using several examples, how difficult an assessment of 'supplying in the course of a commercial activity' is under the current legal terminology in the Recitals. Section E then looks at specific rules pointed at open-source software within the Cyber Resilience Act, such as the special regulatory regime for 'open-source software stewards'. Based on this legal framework, Section F questions whether the Cyber Resilience Act now achieves a balance between encouraging open-source software development and mitigating cybersecurity problems. Based on this balance, Section G looks at the future of open-source software under EU law. Section H concludes.

B. Open-Source Software

10 Open-source software originates from an academic environment. At MIT, Richard Stallman intended to design a free operating system that opposed the barriers developing against sharing software in the 1980's.[14] To support the GNU project, Stallman established the Free Software Foundation (FSF). The FSF focused on free access and usability of software ('a matter of liberty'[15]) instead of 'free of charge' software.[16] A decade later, the quickly growing community surrounding 'free software' moved towards a new label: 'open source'. The 'free' label was unattractive to many companies, which prevented larger enterprises from becoming involved in the development of 'free' software.[17] Therefore, under the Open Source Initiative, the community created a definition for 'open-source' software next to 'free' software.[18]

11 Open-source software is a type of software with source code that is publicly accessible. The use of open-source software comes with some requirements, which different developers have formalised in specific licenses.[19] Some developers, for instance, specify that users accept that they receive the software 'as-is', so that the developers cannot be held liable for damages caused by the software.[20] At the same time, the licenses also formalise that the developers cannot discriminate based on the envisioned use of the software: any type of user (e.g., large technology companies, hobby developers) can freely access and use the code how they desire (e.g., modification, sharing).[21]

12 This Section analyses open-source software and its unique characteristics in comparison to its counterpart, proprietary/closed-source software. In addition, the Section highlights the cybersecurity characteristics of both software development methods.

I. The Development and Ideologies of Open-Source Software

13 Open-source software is published on a diverse set of platforms by equally diverse developers. Developers participate to different degrees (e.g., occasional code change to full-time work), receive different types of remuneration (e.g., full salary, donations), and contribute based on diverse motivations (e.g., passion, peer recognition). This diversity laid the groundwork for 'open source' as a community of people involved with all types of projects that aim at providing open access to information and knowledge, such as open-source software and open access science.

14 The counterparts to open-source software exists in two forms: proprietary software (restrictive

14 Richard Stallman, 'Initial Announcement' (*GNU*, 27 September 1983) <https://www.gnu.org/gnu/initial-announcement.html> accessed 19 January 2024.

15 'What Is Free Software? - GNU Project - Free Software Foundation' <https://www.gnu.org/philosophy/free-sw.html.en> accessed 13 January 2025.

16 Moreno Muffatto, *Open Source: A Multidisciplinary Approach* (Imperial College Press 2006) 7.

17 'History of the OSI' (*Open Source Initiative*, 19 September 2006) <https://opensource.org/history/> accessed 19 January 2024.

18 Open Source Initiative (n 2); Muffatto (n 17) 14.

19 P McCoy Smith, 'Copyright, Contract, and Licensing in Open Source' in Amanda Brock (ed), *Open Source Law, Policy and Practice* (2nd edn, Oxford University Press 2022).

20 See, as an example, the 1-clause BSD license: <https://opensource.org/license/bsd-1-clause>.

21 Open Source Initiative (n 2).

licensing) and closed-source software (restricted access to source code). Proprietary software works with licenses that severely restrict the user in their use of the software (e.g., no modifying the source code). Proprietary software can thus also be open-source software, as software with publicly accessible source code but a restrictive license.[22] In addition, proprietary software exists as closed-source software, where the source code is not available *and* the license restricts the user. Open-source or closed-source software is thus a choice during the development phase of a software package, while proprietary software refers to the distribution phase.

15 The dichotomy between open-source and proprietary/closed-source software can be illustrated through the Linux and Microsoft Windows operating systems: Linux is an open-source operating system, with many different versions existing today, because the license allows modification of the code (e.g., Linux Mint, Ubuntu, Arch Linux).[23] Microsoft develops the proprietary and closed-source Windows operating system; its source code is not publicly available and its license restricts any modification to the Windows source code. Microsoft thus solely develops and controls the different Windows versions.

16 Open-source software exists in many forms. Linux is a prominent example because, as a popular operating system, it has millions of users. However, open-source software also exists on a smaller scale, for example as a small web app that maybe a hundred people may use. When the software license complies with the open source definition of the Open Source Initiative (OSI),[24] the open source community considers it open-source software.[25]

17 There is no singular form of organization behind open-source software development.[26] Since open-source software is usually – but certainly not always – free for users, open-source software developers often rely on smaller financial resources to build their software. Open-source developers often have other intrinsic and extrinsic motives. Intrinsic motives rely on "the tendency to seek out novelty and challenges" (e.g., improving knowledge of a certain programming language), while extrinsic motives focus on the outcome of certain conduct (e.g., improving reputation among peers in the development community).[27] Some developers therefore band together under a non-commercial entity and offer technical support to their largest users for a fee, while other developers work on projects completely voluntarily or based on small donations from end users.

18 In connection to the structure of different open-source software, the users of the software differ considerably, as anyone can access the software's source code. Major technology enterprises frequently use open-source software as a foundation on which they build their proprietary software packages; individuals might instead use open-source software because of its lower cost or as an alternative to the monopoly power of large technology enterprises.[28]

19 In line with these different structures and users of software, I identify three types of open-source software projects: 1) a standalone open-source project (e.g., a developer publishing some personal code); 2) open-source software incorporated into other proprietary and/or open-source software (e.g., Log4j);[29] 3) commercialized open-source software (e.g., where the organisation requires a fee for usage).[30] The difference between a standalone project (1) and an integrated project (2) largely relies on the use case of the software package, since some packages do not offer standalone functionalities.[31] Section C illustrates the meaning of this categorization within the legal framework of the Cyber Resilience Act.

20 Open-source developers often publish the source code of their software on online repositories (e.g., GitHub, SourceForge, personal websites). Other developers can access the code there, and download it for further use, or review the code and offer

22 For some examples, see <https://en.wikipedia.org/wiki/List_of_proprietary_source-available_software>.

23 For an extensive list of Linux distributions, see <https://en.wikipedia.org/wiki/List_of_Linux_distributions>.

24 <https://opensource.org/licenses/>.

25 This does not mean that there are no open-source software licenses outside the OSI's list, but merely that the OSI has not (yet) classified them as compliant with the open source definition.

26 Muffatto (n 17) ch 3.

27 Jürgen Bitzer, Wolfram Schrettl and Philipp JH Schröder, 'Intrinsic Motivation in Open Source Software Development' (2007) 35 Journal of Comparative Economics 160; Muffatto (n 17) 58–62.

28 Muffatto (n 17) 62–64.

29 For instance, on Microsoft's evolving stance towards open-source software, see Benjamin J Birkinbine, *Incorporating the Digital Commons: Corporate Involvement in Free and Open Source Software* (University of Westminster Press 2020) 49–72.

30 RedHat is the most prolific example of such projects, see also ibid 73–88; Although Red Hat recently changed its company policies, to the dismay of the open source community, Kevin Purdy, 'Red Hat's New Source Code Policy and the Intense Pushback, Explained' (*Ars Technica*, 30 June 2023) <https://arstechnica.com/information-technology/2023/06/red-hats-new-source-code-policy-and-the-intense-pushback-explained/> accessed 13 December 2023.

31 The Cyber Resilience Act also speaks of certain types of open-source software 'intended for integration by other manufacturers'. Recital 18 CRA.

feedback.[32]

21 Open-source software is part of the broader 'open source' movement, which is based on certain philosophical (e.g., about information and knowledge) or pragmatic beliefs (e.g., free alternatives for users) about the need for open-source software.[33] These beliefs explain the altruistic nature of open source and relate back to the Free Software Foundation: many developers offer their software to the public because they are part of a wider community movement which aims to keep knowledge, in a broad sense, publicly accessible and shareable.[34]

II. Open-Source Software and Cybersecurity

22 Open-source software represents a deliberate choice for transparency: the source code of the software is accessible and the developers are transparent about its inner workings. An alternative to such transparency is 'security through obscurity'.[35] This dichotomy between 'transparency' and 'obscurity' forms the foundation for many security-related discussions about open-source software.[36]

23 By hiding the inner workings of the software, closed-source software does not show its internal processes; attackers cannot view the source code to discover exploitable vulnerabilities.[37] In contrast, advocates for open-source software development believe transparency allows open-source software to be more secure.[38] In the following, I illustrate the security dynamics of open-source and closed-source software in two phases: 1) during the development of the software and 2) after publication of the software.

1. Development of Software

24 Proponents often use the transparent nature of open-source software as an argument that open-source software is more secure; if developers can peer review source code, they can identify and patch vulnerabilities and similar problems quickly.[39]

25 Raymond coined this view of security of open source code as 'Linus' Law': "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."[40] Thus, an open-source software package is more secure if – and only if – many developers view and co-operate on the source code as the project benefits from their diverse views.[41]

26 The prevention of backdoors is an example of the benefits of the 'many eyeballs' system. If attackers change the source code of open-source software to allow themselves backdoor access to the system, or if the backdoor existed from the start, other developers can easily notice such changes and prevent the attackers from exploiting the backdoor.[42] This is not the case for closed-source systems, where such backdoors are not immediately visible to others.

27 An opposing view to the 'many eyeballs' principle of Linus' Law is the view of 'too many cooks in the kitchen'.[43] In the latter view, the security of open-source software diminishes because too many developers are working on the software simultaneously and in fragmented ways.[44] A single developer may decide to contribute solely to their preferred elements of the project, without

32 GitHub had more than 400 million contributions to open-source projects in 2022. See <https://github.blog/news-insights/research/octoverse-2022-10-years-of-tracking-open-source/>.

33 Ian Walden, 'Open Source as Philosophy, Methodology, and Commerce: Using Law with Attitude' in Amanda Brock (ed), *Open Source Law, Policy and Practice* (2nd edn, Oxford University Press 2022).

34 Charlotte Hess and Elinor Ostrom (eds), *Understanding Knowledge as a Commons: From Theory to Practice* (MIT Press 2007).

35 Hoepman and Jacobs (n 6).

36 Charles-H Schulz, 'Open Source Software and Security: Practices, Governance, History, and Perceptions' in Amanda Brock (ed), *Open Source Law, Policy and Practice* (2nd edn, Oxford University Press 2022); Christian Payne, 'On the Security of Open Source Software' (2002) 12 Information Systems Journal 61.

37 Ross Anderson, 'Open and Closed Systems Are Equivalent (That Is, in an Ideal World)' in Joseph Feller and others (eds), *Perspectives on free and open source software* (MIT Press 2005).

38 Eric Raymond, 'The Cathedral and the Bazaar' (1999) 12 Knowledge, Technology & Policy 23.

39 ibid.

40 'Linus' refers to the founder of the Linux operating system, Linus Torvalds. Raymond also more informally coins Linus' Law as "Given enough eyeballs, all bugs are shallow", see ibid 29.

41 Raymond (n 38).

42 Payne (n 36) 66–67.

43 Andrew Meneely and Laurie Williams, 'Secure Open Source Collaboration: An Empirical Study of Linus' Law', *Proceedings of the 16th ACM conference on Computer and communications security* (ACM 2009) 453; Ann Barcomb and others, 'Managing Episodic Volunteers in Free/Libre/Open Source Software Communities' (2022) 48 IEEE Transactions on Software Engineering 260.

44 Martin Pinzger, Nachiappan Nagappan and Brendan Murphy, 'Can Developer-Module Networks Predict Failures?', *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering* (ACM 2008).

contributing to the overall project goals. This 'unfocused contribution' forms a security risk.[45] Unfocused contributions disrupt the concept of Linus' Law in large-scale open-source projects, as the additional 'eyeballs' do not necessarily improve the project.[46] Therefore, the idea that open-source software is more secure simply because a diverse set of developers can access the source code is not clearly proven.

2. Post-Release Vulnerabilities

28 Linus' Law mainly relates to the development phase of open-source software projects. However, security problems can also develop in the post-release phase, after publication of the software or a new version release.

29 In a comprehensive study, Schryen found that there was no statistical significance in terms of the severity of vulnerabilities between open-source and closed-source software equivalents.[47] He also found that the type of patching behaviour, in terms of speed and type of vulnerabilities, differed significantly between different open-source and closed-source vendors. This difference existed across open-source and closed-source vendors: the mode of open-source or closed-source development seemed, therefore, not to influence patching behaviour.[48]

30 Ransbotham analyses how threat actors exploit vulnerabilities differently between open-source and closed-source projects based on two years of log data from intrusion detection systems.[49] He holds that vulnerabilities of open-source software projects have a generally greater risk of exploitation and receive more exploitation attempts. These differences can be partially attributed to the difference in transparency between open- and closed-source software. If a vulnerability is discovered internally in a closed-source environment, the developers have some additional time to work on fixing the vulnerability before they make the changes public. In open-source projects, changes in the source code – and thus possible vulnerabilities – are immediately publicly accessible.[50]

31 In general, there are thus small differences between open-source and closed-source software security, both in the development and post-release phase. Vulnerabilities exist in and impact both types of software.

C. The Cyber Resilience Act and Open-Source Software Cybersecurity

32 European law did not consider cybersecurity rules for open-source software until 2022. This lack of regulation changed when the European Commission proposed the 'Cyber Resilience Act', which contained specific rules for open-source software cybersecurity.[51] The Cyber Resilience Act was adopted at the end of November 2024 and comes into effect on 10 December 2024.[52]

I. The Cyber Resilience Act in Short

33 The Cyber Resilience Act imposes 1) *cybersecurity requirements* on 2) *manufacturers* of 3) *products with digital elements* that they 4) *place on the Union's market* in the course of a 5) *commercial activity*.[53] Below, I briefly review these elements in light of the applicability of the Act to open-source software.[54]

1. Cybersecurity Requirements

34 The cybersecurity requirements for products with digital elements form the focal point of the Cyber Resilience Act. These requirements include security throughout the lifecycle of the product (security-by-design), releasing the product without known exploitable vulnerabilities, and protection of the integrity and authenticity of data.[55] Next to these requirements, the Act contains traditional product requirements (e.g., providing documentation) and security-specific duties (e.g., providing security

45 Meneely and Williams (n 43) 456.

46 Meneely and Williams (n 43); Andrew Meneely and Laurie Williams, 'Strengthening the Empirical Analysis of the Relationship between Linus' Law and Software Security', *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (ACM 2010).

47 Guido Schryen, 'Is Open Source Security a Myth?' (2011) 54 Communications of the ACM 130, 136–137.

48 ibid 139.

49 Sam Ransbotham, 'An Empirical Analysis of Exploitation Attempts Based on Vulnerabilities in Open Source Software' [2010] Workshop on the Economics of Information Security 1.

50 ibid 5.

51 Cyber Resilience Act proposal (n 10).

52 Cyber Resilience Act (n 11).

53 Art 1 CRA.

54 See also Liane Colonna, 'The End of Open Source? Regulating Open Source under the Cyber Resilience Act and the New Product Liability Directive' (2025) 56 Computer Law & Security Review 106105.

55 Annex I Part 1 CRA.

updates).[56]

35 There are two main methods for developers to show their compliance in the proposal: 1) performing a self-assessment; and 2) receiving a third-party audit.[57] In general, the choice for a specific route depends on the type of product. The Cyber Resilience Act categorises products with certain privileges in networks or computer systems (e.g., password managers, operating systems) as 'important' products.[58] Important products must, if they cannot follow certain European technical standards, perform a third-party audit to prove their compliance with the Act's requirements.[59] Open-source software is exempted from a third-party audit, even if they are considered 'important products', as long as they provide technical documentation to the public.[60]

36 The provision and the supporting Recital do not indicate a reason for this exemption. However, many open-source software packages have certain elevated privileges and would therefore be important products (e.g., Log4j). In that context, the Parliament and Council most likely wanted to prevent a 'chilling effect' on open-source software development in the face of possibly costly third-party audits.

37 A further category exists for 'critical' products with digital elements, with even stricter conformity requirements.[61] The Act currently lists three critical products: hardware devices with security boxes; smart meter gateways; and smartcards.[62]

2. Manufacturers

38 A manufacturer is a "natural or legal person who develops or manufactures products with digital elements".[63] Both traditional hardware manufacturers and software developers are 'manufacturers' under the Cyber Resilience Act. In case manufacturers do not strictly produce the product themselves, but place their trademark on products produced by another actor, they remain the manufacturer of the final product.[64]

39 As highlighted above, not all open-source software forms a standalone package. Some of the most prominent open-source software packages derive their popularity from integration by proprietary software developers. Google, for instance, uses numerous pieces of open-source software, such as databases,[65] for their own software packages (e.g., Google Maps). Google, in this example, creates and markets their end product and is thus the manufacturer for the end product under the Cyber Resilience Act.[66] The proprietary developers must thus also ensure that they securely integrate the open-source database system – the open-source developer is not responsible for compliance in this case.[67] I delve into this separation further in Section E.II.

40 The Cyber Resilience Act includes a set of rules for importers and distributors too. These rules ensure that manufacturers cannot evade compliance by letting importers and distributors bring the product to the Union market.[68] An importer brings products with digital elements to the Union market of "a natural or legal person established outside the Union."[69] A distributor is an actor that is not a manufacturer or importer, but who still places the product on the market.[70] Importers and distributors have separate responsibilities to ensure that the products they place on the Union market comply with the requirements of the Cyber Resilience Act.[71]

3. Product with Digital Elements

41 The provisions of the Cyber Resilience Act apply to 'products with digital elements', meaning "any software or hardware product".[72] Open-source software is thus a 'product with digital elements' if: 1) the open-source project develops *software* or *hardware*; and 2) that software or hardware is a *product* under the Cyber Resilience Act.

56 Art 13 CRA.
57 Art 32(1) CRA.
58 Art 7(1) CRA & Annex III CRA. The Commission proposal used the term 'critical' products, which is now an even more critical class above important products.
59 Art 32(2) CRA.
60 Art 32(5) & Recital 91 CRA.
61 Art 8 & Art 32(4) CRA.
62 Annex IV CRA.
63 Art 3(13) CRA.
64 Art 3(13) CRA.
65 For instance, Google moved their database systems to the open-source MariaDB, see Jack Clark, 'Google Swaps out MySQL, Moves to MariaDB' *The Register* (12 September 2013) <https://www.theregister.com/2013/09/12/google_mariadb_mysql_migration/> accessed 14 August 2024.
66 Art 3(13) & Art 13(5) CRA.
67 Izquierdo Grau analyses this division between standalone open-source and integrated open-source in the context of the recent Product Liability Directive proposal, see Guillem Izquierdo Grau, 'An Appraisal of the Proposal for a Directive on Liability for Defective Products' (2023) 12 Journal of European Consumer and Market Law 198.
68 Art 19 & 20 CRA.
69 Art 3(16) CRA.
70 Art 3(17) CRA.
71 Art 19(2) & 20(2) CRA.
72 Art 3(1) CRA.

42 The Cyber Resilience Act defines open-source software as "software the source code of which is openly shared and [...] made available under a free and open-source license."[73] From this definition, however, it is not immediately clear that open-source software is also a software *product.*

43 The Cyber Resilience Act itself does not define what a 'product' is. The EU's Blue Guide, the Commission's interpretation guide for product rules, offers some additional guidance for definitions related to European product legislation.[74] The Guide defines a product in relation to its placing on the market: "Union harmonisation legislation applies to products which are intended to be placed (and/or put into service) on the market."[75] This element of 'placing onto the market' is thus an important qualifier for open-source software as a software *product* under the Cyber Resilience Act.

4. Placing on the Market

44 The Cyber Resilience Act defines that a product is placed on the market when it is "made available" on the Union market, meaning "the supply of a product [...] for distribution or use [in the Union] *in the course of a commercial activity*, whether in return for payment or free of charge."[76] These definitions highlight that open-source software can thus be offered on the market – and therefore be a product under the Cyber Resilience Act – even if the software is offered for free.

45 Additionally, open-source software is "placed on the market" in the sense of the Cyber Resilience Act if the developer supplies the product "in the course of a commercial activity". Although this is an additional requirement, its abstract character caused much discussion after the Commission's proposal.[77]

5. Commercial Activity

46 The provisions of the Cyber Resilience Act do not clearly define 'supplying a product in the course of a commercial activity'. Recital 18 of the Cyber Resilience Act states that "only free and open-source software made available on the market, and therefore supplied for distribution or use in the course of a commercial activity should be covered by this Regulation." Although the Recitals are not legal provisions, they offer an interpretation of what 'supplying in the course of a commercial activity' means in the context of open-source software.[78]

47 The Recitals note several examples of open-source software supplied in the course of a commercial activity. Open-source software is supplied in the course of a commercial activity if the developer 1) charges a price for a product; 2) charges a price for technical support services that does not serve the recuperation of actual costs; 3) provides a software platform where the manufacturer monetises other services; or 4) if the software requires as a condition for use the processing of personal data, unless for certain legitimate purposes (e.g., security).[79] The legislators seemingly had particular open-source projects in mind when drafting these examples. For instance, the provision of a software platform where the manufacturer monetises other services can relate to Android: the core of Google's mobile operating system is open source, but Google integrates the Google Play Store, Google Drive, and other similar services into Android when providing the platform to smart phones.[80]

48 This list is not exhaustive, as the Recital notes that supply within the course of a commercial activity "might be characterised" by the options mentioned above.[81] Other activities and conditions can also bring the open-source software project in the context of a commercial activity, placing additional emphasis on the question when an activity is 'commercial' under the Cyber Resilience Act.

49 Many hobby developers add donation options to their open-source software (e.g., Patreon, PayPal). Developers often make such donation requests to cover the project's maintenance costs (e.g., website

73 Art 3(48) CRA.

74 Commission notice – The 'Blue Guide' on the implementation of EU product rules [2022] OJ C247/1.

75 Blue Guide (n 75), 17.

76 Art 3(22). Emphasis mine.

77 Aertsen (n 13); Webmink In Draft, 'Fixing The CRA For Open Source' (*Webmink In Draft*, 20 February 2023) <https://the.webm.ink/fixing-the-cra-for-open-source> accessed 21 February 2023; Nicholson (n 13).

78 See Llio Humphreys and others, 'Mapping Recitals to Normative Provisions in EU Legislation to Assist Legal Interpretation', *JURIX* (2015) 42–44 and cases cited therein.

79 Recital 15 CRA.

80 Ron Amadeo, 'Google's Iron Grip on Android: Controlling Open Source by Any Means Necessary' (*Ars Technica*, 21 July 2018) <https://arstechnica.com/gadgets/2018/07/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/> accessed 19 January 2024.

81 Recital 15 CRA.

costs).[82] At the same time, research shows that, in certain large-scale open source projects, code contributions by companies can be ten times larger than contributions by volunteers.[83] Such large-scale contributions might lead to the conclusion that the entire open-source project falls into a 'commercial activity', as commercial parties maintain nearly the entire project. A strict dichotomy between open-source software and commerciality does not exist.[84] There are diverse ways in which an open-source project can obtain financial and/or organisational support.[85]

50 The Commission proposal lacked insight into these diverse methods of commerciality, as the text only gave examples of open-source software supplied during a commercial activity.[86] The Council and Parliament, in response, significantly expanded the Recitals, especially regarding open-source software. As a result, the legislators exempted many types of open-source software from the scope of the Act. For example, the amended Recitals state that asking for donations does not constitute supply in the course of a commercial activities, as long as the developers do not seek to gain profits from those donations.[87] Furthermore, the Recitals state that an open-source project is not supplied in the course of a commercial activity merely due to development support from commercial entities.[88] In sum, the role of open-source software within the Cyber Resilience Act largely depends on whether the software is supplied in the course of a commercial activity.

D. Assessing the commerciality of a project

51 The commerciality of open-source software largely determines whether the software falls under the scope of the Cyber Resilience Act. Therefore, the exact meaning of 'supplying in the course of a commercial activity' merits further examination.

52 Most activities are commercial if developers use them to earn a profit, i.e. the income from these actions exceed maintenance costs. For example, the Cyber Resilience Act lists charging a price for the software or for technical support, when this exceeds maintenance costs, as indicative of supplying the software in the course of a commercial activity.[89]

53 In contrast, certain projects are not supplied during a commercial activity. Again, developers of such projects mostly do not earn income that exceeds their maintenance costs, such as receiving small donations.

82 Cassandra Overney and others, 'How to Not Get Rich: An Empirical Study of Donations in Open Source', *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (ACM 2020).

83 Yuxia Zhang and others, 'Companies' Participation in OSS Development–An Empirical Study of OpenStack' (2021) 47 IEEE Transactions on Software Engineering 2242, 2249.

84 Wheeler (n 14).

85 ibid.

86 Aertsen (n 13).

87 Recital 15 CRA.

88 Recital 18 CRA.

89 Recital 15 CRA.

Table 1: The scope of the Cyber Resilience Act for open-source software[90]

Indicative of a supplying the software *in* the course of a commercial activity	• An intention to monetise beyond the recuperation of actual costs • Charging a price for the product • Charging a price for technical support • Personal data processing as a condition for use of the software (except for certain justified purposes) • Accepting donations exceeding the costs of developing and maintaining the software, without the intention to make a profit.
Indicative of a supplying the software *outside* the course of a commercial activity	• Monetisation only to recuperate costs of maintenance, instead of making a profit (e.g., by public administration entities) • Supply of software intended to be integrated by other manufacturers, without monetisation of original software • Products which receive financial support or developmental support from manufacturers • The mere presence of regular releases • Development by non-profit organisations, if they use their earnings after cost for non-profit objectives • Contributions to open-source software when not involved in project leadership/ownership • Mere distribution on repositories
Special regulatory regime	Open-source software stewards, legal persons who "provide support on a sustained basis" for the development of open-source software and play a "main role in ensuring the viability" of open-source software

90 Recital 16-20 CRA.

54 Table 1 shows how the Recitals include and exempt numerous open-source software projects from the scope of the Cyber Resilience Act. Based on this overview, a few questions remain.

55 The list of commercial activities in the Recitals is non-exhaustive; the Recital states that a commercial activity "*might be* characterised" by the options mentioned.[91] In the future, courts may thus amend the list and determine that other activities are also commercial.

56 An assessment of other activities, however, is difficult, as the Recitals further state that "the mere circumstances under which the product has been developed, or how the development has been financed, should [...] not be taken into account" when assessing the commercial nature of the software.[92] This limitation seems to directly contradict the Recitals themselves. As shown in Table 1, the Recitals explicitly exempt certain types of development (e.g., development by commercial entities) and financial models (e.g., receiving donations) from the scope of 'supplying a product in the course of a commercial activity'. A court can thus seemingly not assess the commerciality of a project as the Recitals currently do.

57 Additionally, the Recitals contain an unclear role for the *intention* of gaining a profit. In the context of donations, the Recitals state that accepting donations "exceeding the costs [of] design, development and provision of a product" means that the software is supplied in the course of a commercial activity.[93] In contrast, when developers accept donations "without the *intention* of making a profit", they do not supply the product in the course of a commercial

91 Recital 15 CRA. Emphasis mine.

92 Recital 18 CRA.

93 Recital 15 CRA.

activity.[94] It is unclear from the Recitals when intent is measured: at the start of the project or when the developers introduce certain financing methods. A developer may not intend to make a profit initially, but, as the project grows, may consider it reasonable. Likewise, the developer may not intend to make a profit, but may receive such large donations from enthusiastic users that they completely exceed all maintenance costs. Such situations, which are not clearly determined in the Recitals, remain complex.

58 The Commission may still resolve some of the Recital's complexities. Pursuant to Article 26 of the Act, the Commission may publish guidance to support the application of the Cyber Resilience Act. The scope of the Act for free and open-source software is of particular importance when they provide such guidance.[95]

59 In sum, the Recitals, in general, indicate clearly when open-source software is commercial. If developers publish open-source software for which consumers pay a commercial price or other consideration (e.g., personal data), they supply the software in the course of a commercial activity. If, in contrast, developers merely maintain or support open-source software, they do not supply the software during a commercial activity. Simultaneously, when moving beyond a general assessment, the Recitals do contain certain conflicting statements. These statements might hinder clear answers to future questions surrounding the position of open-source software under the Cyber Resilience Act.

E. Specific provisions for open-source software within the Cyber Resilience Act

60 The Cyber Resilience Act does not only regulate open-source software developers to improve the cybersecurity of open-source software. The Act also prescribes specific rules for 'open-source software stewards', proprietary software developers, and other parties with the aim of improving the overall cybersecurity of the open-source software ecosystem.

I. Open-source software stewards

61 In the open-source software community, there are certain organisations that support the development of open-source software as part of their overall mission statement. In some cases, these organisations also develop core open-source software. An example of such an organisation is the Python Software Foundation, which aims to advance the Python programming language and its community. The foundation organises conferences, offers grants to developers, and "produces the core Python distribution".[96] Python is a core programming language for software worldwide; it ranks second, after JavaScript, in a recent study from Github on the open-source software hosted on their platform.[97] The Python Software Foundation thus offers core support to the open source community, both through development and support.

62 The Cyber Resilience Act addresses organisations such as the Python Software Foundation as 'open-source software stewards'.[98] A steward is a legal person that provides systematic support for the development of open-source software, which is *intended for commercial activities*, as part of their overall objectives.[99] Importantly, the definition states that a steward is *not* a manufacturer.

63 Open-source software stewards receive a special position within the supervision scheme of the Cyber Resilience Act. Stewards are subject to a "light-touch and tailor-made regulatory regime".[100] The idea behind this scheme seems to be that open-source software stewards are vital to the continuation of the open-source ecosystem; the legislators believe they have a "main role in ensuring the viability of [open-source software]".[101]

64 Open-source software stewards have several obligations.[102] First, stewards must put in place cybersecurity policies for secure development of open-source software and vulnerability handling by the developers of that software.[103] The Python Foundation, for instance, has a vulnerability handling system where users can contact the 'Python Security Response Team' for support.[104] Stewards cannot be fined for non-compliance with these obligations,[105] but they can be required to take certain corrective

94 Recital 15 CRA.
95 Art 26(2)(a) CRA.
96 <https://www.python.org/psf/mission/>.
97 Kyle Daigle and GitHub Staff, 'Octoverse: The State of Open Source and Rise of AI in 2023' (*The GitHub Blog*, 8 November 2023) <https://github.blog/news-insights/research/the-state-of-open-source-and-ai/> accessed 13 August 2024.
98 Recital 19 CRA: 'open-source software stewards include certain foundations[.]'
99 Art 3(14) CRA.
100 Recital 19 CRA.
101 Recital 19 CRA.
102 Art 24 CRA.
103 Art 24(1) CRA.
104 <https://www.python.org/dev/security/>.
105 Art 64(10)(b) CRA.

actions.[106] This exemption also means that stewards cannot affix a CE-mark to their product.[107]

65 Stewards must also co-operate with market surveillance authorities to mitigate vulnerabilities in open-source software packages.[108] Market surveillance authorities are responsible for taking corrective measures when developers do not comply with the rules of the Act. This co-operation seems to be the essence of the steward role: providing communication between the open-source community and authorities in cases such as Log4j. In that line, it is logical that open-source software stewards provide support for software *with commercial intent*, meaning integration into proprietary products or services.[109] Through commercial integration, these software packages – and their vulnerabilities – have considerable influence on the global software ecosystem.

66 Finally, there are obligations for stewards that are also involved with development of open-source software.[110] They must also comply with certain notification obligations for developers, particularly the notification of actively exploited vulnerabilities.[111] However, as open-source software stewards are *not* manufacturers per the definition in Article 3(14), they do not have equal obligations to traditional manufacturers. Article 24(3) only lists notification obligations.

67 It is imaginable that a strict delineation between open-source software stewards and manufacturers is not feasible in practice. Stewards, such as the Python Foundation, also develop software. Are such stewards then manufacturers for that software independently – assuming the software is supplied in the course of a commercial activity – or are they stewards – and thus *not* manufacturers – for both providing support and developing products? As described above, in the former they must comply with the Act's many obligations for manufacturers, while in the latter they only carry the notification obligations of Article 24(3).

68 As with the Recitals above, the Commission may provide some answers to the role of open-source software stewards when it publishes guidance on the application of the Cyber Resilience Act.[112] Moreover, regulators could eventually solve such conflicts through the 'tailor-made' regulatory regime for open-source software stewards.

106 Art 52(3) CRA.
107 Recital 19 CRA.
108 Art 24(2) CRA.
109 Recital 19 CRA.
110 Art 24(3) CRA.
111 Art 24(3) & 14(1) CRA.
112 Art 26(2)(a) CRA.

II. Proprietary manufacturers using open-source software

69 The Cyber Resilience Act applies to manufacturers of software and hardware products. This scope means that proprietary manufacturers are also responsible for improving open-source software cybersecurity, through several ways.

70 First, the Cyber Resilience Act inherently applies the broad applicability of the Act means that proprietary software – and proprietary software developers – must adhere to certain cybersecurity requirements. Since open-source software is virtually always part of proprietary software, the requirements for the proprietary software package inherently involve the underlying open-source software.

71 This connection between the cybersecurity of the proprietary package and the open-source software is made explicit in the Act. The Cyber Resilience Act requires manufacturers to exercise due diligence when integrating third-party components, including open-source components, into their own product.[113] This obligation seems to stem from cases such as the Log4j vulnerability, in which a vulnerability in an open-source component puts the entire (proprietary) software package at risk.

72 When exercising this due diligence, manufacturers may discover certain vulnerabilities. If a manufacturer identifies a vulnerability within an open-source component of their own software, they must, under the Act, report it to the open-source developers.[114] The manufacturers must also remediate the vulnerability according to the vulnerability handling requirements of the Act.[115] If, as part of this remedy, the manufacturers modify the code or hardware to address the vulnerability, they must also share this code with the open-source developer.

73 Other parties may help identify and remediate vulnerabilities in open-source software through voluntary security attestation programmes.[116] The Commission can set-up such a programme through delegated acts. These programmes strive to improve the overall cybersecurity of open-source software which is exempted from the scope of the Cyber Resilience Act.[117] The exact content of a security attestation programme, i.e. if the Commission

113 Art 13(5) CRA.
114 Art 13(6) CRA.
115 Art 13(6) & Annex I Part 2 CRA.
116 Art 25 CRA mentions 'developers or users' of open-source software and 'other third parties'. See also Recital 21.
117 Recital 21 CRA speaks of open-source software 'not subject to the essential requirements' of the Act.

provides financial or organisational support, is not clear from the provisions.

74 The due diligence obligation and the voluntary security attestation programmes help to expand the parties which support the cybersecurity of open-source software packages.

F. Cybersecurity and open-source software: a problem solved?

75 There is a fine balance between enhancing open-source software cybersecurity and regulating the open=source ecosystem which may rely on ad-hoc and voluntary work. The Cyber Resilience Act shows how delicate this balance is, with its many exemptions and categorisations of open-source software, to ensure that only software supplied within the course of a commercial activity is regulated. The question is then whether these considerations achieve a balance between mitigating cybersecurity risks of open-source software and introducing feasible legal obligations for the sector.

76 A project like Log4j, for instance, does not fall under the scope of the Cyber Resilience Act. The project does not charge a price for the software nor conducts any activities explicitly listed as commercial in the Cyber Resilience Act. The project is merely supported by certain donators and commercial entities, which are both explicitly exempted as commercial activities.[118] Most likely, Log4j itself would, therefore, not fall within the scope of the Cyber Resilience Act. The only cybersecurity obligations related to Log4j exist for entities who integrate Log4j into their own proprietary software.

77 On a general level, the Cyber Resilience Act is a step in the right direction for cybersecurity, regardless of the rules imposed on open-source software. Many cybersecurity requirements introduced by the Act were not present in existing legislation.[119] The Act thus, at minimum, might improve the cybersecurity of proprietary software, even if it would not cover open-source software.

78 In the specific context of open-source software, the Act aims to balance between improving cybersecurity of open-source software while not discouraging open-source software development. Broadly speaking, the Act only covers 'commercial' open-source software. Many types of open-source software are non-commercial, as evident by the Recitals, which means that most open-source software is not regulated by the Cyber Resilience Act. The balance seems, thus, to fall in favour of alleviating regulatory pressure on open-source software developers, instead of (fully) improving open-source software cybersecurity. However, the cybersecurity side is also supported by the responsibilities imposed on integrators of open-source software and the voluntary security attestation programmes.

79 In sum, the Cyber Resilience Act aims to make open-source software more secure than it is currently, without imposing responsibilities on developers that may discourage further open-source software development. The legislation certainly emphasizes not discouraging the development, but responsibilities on both developers and users of open-source software will likely help improve its cybersecurity.

G. The future of open-source software under EU law

80 The Cyber Resilience Act is the first piece of legislation that aims to strike a balance between responsibilities for open-source software and supporting its ecosystem.[120] This means that the legislative choices made in the Act will have consequences for the future of open-source software under EU law. However, the Cyber Resilience Act includes many of its considerations for open-source software in the Recitals. This legislative choice has two consequences: 1) there is no clear embedded legal framework for open-source software in the Cyber Resilience Act, due to the applicability of the Recitals and 2) many of the considerations are specific to the current landscape of open-source software and therefore overly restrictive when considering future developments.

81 Recitals only have legal power insofar as the Court of Justice of the European Union and supervisory authorities use them to interpret the provisions of the Cyber Resilience Act. In 1998, the Court held that "the preamble to a Community act has no binding legal force and cannot be relied on as a ground for derogating from the actual provisions of the act in question."[121] Recitals, therefore, can be useful for interpretation of ambiguous legal provisions (e.g., supplying in the course of a commercial activity)

118 Based on the assumption that the donations do not exceed the project's maintenance costs. For further information, see <https://logging.apache.org/log4j/2.x/support.html>.

119 Pier Giorgio Chiara, 'The Cyber Resilience Act: The EU Commission's Proposal for a Horizontal Regulation on Cybersecurity for Products with Digital Elements: An Introduction' [2022] International Cybersecurity Law Review.

120 Colonna (n 54).

121 Case C-162/97 *Nilsson and others* ECLI:EU:C:1998:554, [1998] ECR I-7477, para 54.

but are not separate legal provisions on which the Court will rely.

82 In addition, the Recitals are very specific and pinpoint different commercial modes within the current landscape of open-source software. Future developments may fall outside the scope of the current Recitals. For instance, a developer could place advertisements in their software, based on user consent to see them. These advertisements allow the developer to continue working full-time on the project and similar projects. Would this choice constitute an "intention to monetise",[122] which places the project inside the course of a commercial activity? Or is this just a circumstance under which "the development has been financed",[123] although the developer also uses the money to work on other projects? European consumer law tackles this problem for 'information society services' by stating that they are "provided for remuneration".[124] 'Remuneration' is a broad concept which involves advertisement income, but also the request for personal data by the service, as in the Cyber Resilience Act.[125] In comparison, the Cyber Resilience Act's notion of a commercial activity then seems overly restrictive, while a concept such as 'for remuneration' more easily adapts to future developments.

83 It seems that the Cyber Resilience Act's approach of placing virtually all considerations for open-source software in the Recitals might make the Act particularly vulnerable to future developments. This focus on the existing landscape, combined with the difficult method for assessing commerciality as described in Section D, may impair the applicability of the Cyber Resilience Act in the future. An embedded legal framework for open-source products in product legislation, which could also adapt to future developments, remains missing.[126]

122 Recital 15 CRA.

123 Recital 18 CRA.

124 Art 1(b) Directive (EU) 2015/1535 of the European Parliament and of the Council of 9 September 2015 laying down a procedure for the provision of information in the field of technical regulations and of rules of Information Society services.

125 Recital 18 Directive 2000/31/EC of the European Parliament and of the Council of 8 June 2000 on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market ('Directive on electronic commerce').

126 See also Colonna on the role of open-source software in the new Product Liability Directive and the AI Act, Colonna (n 54).

H. Conclusion

84 This paper analysed the position of open-source software in the Cyber Resilience Act. The paper answered the following question: To what extent does the Cyber Resilience Act impose responsibilities on open-source software developers that achieve a balance between stimulating open-source software development and, simultaneously, mitigating cybersecurity problems within open-source software?

85 Open-source software stems from a unique development culture aimed at distributing knowledge freely. Simultaneously, the software is crucial for the modern digital infrastructure. As with any software, there are certain cybersecurity risks inherent in open-source software. The European Union aims to mitigate some of those risks through the Cyber Resilience Act.

86 The Cyber Resilience Act aims to regulate cybersecurity risks without discouraging open-source software development. The Act achieves this balance by covering only open-source software 'supplied in the course of a commercial activity'. The Act also introduces several other mechanisms to support the cybersecurity of open-source software. First, the Act prescribes a special regulatory regime to open-source software stewards, legal persons who support and advance the open-source software ecosystem. Second, proprietary manufacturers may only integrate open-source software components in a diligent manner. Therefore, they must also fix vulnerabilities discovered in open-source components and share such fixes with the developers of the component. Through voluntary security attestation programmes, the Act also supports other parties interested in advancing open-source software cybersecurity.

87 At the same time, the Recitals contain complex legal terminology. The Recitals mention many modes of financing and development of open-source software and if those modes are 'supplying in the course of a commercial activity'. However, the Recitals also note that an assessment of a project based merely on financing or development modes is not sufficient. It is currently unclear how this situation should be resolved in practice when an open-source project is neither an explicitly included nor excluded commercial activity.

88 The Cyber Resilience Act, however, does certainly advance cybersecurity of open-source software compared to the current regulatory landscape. Through rules for proprietary integration, proprietary software developers are also responsible for the cybersecurity of open-source software. Such rules mean that, even when a project is exempted

from the CRA's scope, it will receive cybersecurity support through the Act's obligations on other parties.

89 The future position of open-source software under EU law remains somewhat unclear after the Cyber Resilience Act, especially since so many of its considerations for open-source software occur in the Recitals. In sum, the Cyber Resilience Act achieves a balance between encouraging open-source software development and mitigating cybersecurity risks within open-source software, but some key challenges remain for the future.