# [draft*] The Cyber Resilience Act: unintended harms to security and stability of Open Source Internet Infrastructure Software

We, the undersigned companies, are responsible for the development and maintenance of some of the most well-known and widely adopted Open Source Internet Infrastructure Software.

**NLnet Labs** (www.nlnetlabs.nl) is a small, independent public benefit organisation founded in 1999. Its mission is to write open-source software and contribute to open standards for the Domain Name System and (safe) inter-domain routing, thereby improving the robustness, security and reliability of the Internet. NLnet Labs maintains widely used implementations for DNS including NSD and Unbound, and the safety of inter-domain routing including Krill and Routinator.

**CZ.NIC** (www.nic.cz) is an association responsible for running the registry for the top level domain of the Czech Republic (.CZ). CZ.NIC also develops and maintains a set of open source software in the area of internet infrastructure like routing daemon BIRD and DNS servers Knot DNS and Knot Resolver.

**Internet Systems Consortium, Inc** (www.isc.org) a not-for-profit company (operating under US IRC section 501(c)3) based in the US, is dedicated to developing software and offering services in support of the Internet infrastructure. ISC is responsible for developing and distributing three widely-deployed open source Internet networking software systems: BIND9, ISC DHCP, and Kea DHCP, and operating one of the 13 root name server systems of the Internet.

**The Network Device Education Foundation** (NetDEF - www.netdef.org) is a not-for-profit company (operating under US IRC section 501(c)3) based in the US. NetDEF maintains, tests and develops the FRR (Free Range Routing) project, implementing OSPF, IS-IS, BGP, RIP, RIPng, and other protocols, under the umbrella of the Linux Foundation.

*** This draft remains subject to amendment and approval by the named parties.**

# Contents

# 1 Introduction: focus of this submission

This submission is focussed on one topic: how the Cyber Resilience Act (CRA)[1] could:

- create unintended but serious harm to the security and stability of Open Source Internet Infrastructure Software,
- and, as a result, increase the risk to the security and stability of the internet.

# 2 The organisations making this submission

We, the undersigned companies, are responsible for the curated development and maintenance of some of the most well-known and widely adopted Open Source Internet Infrastructure Software.[2] The Domain Name System (DNS) and Border Gateway Protocol (BGP) are two of the key technical systems underpinning the Internet infrastructure: the co-signers of this comment represent the major open source implementations of both DNS and BGP.

We share some important characteristics that, taken collectively, help to assure the security and stability of the Open Source Internet Infrastructure Software entrusted to our care.

- We are organised as not-for-profit entities;
- We were created for the purpose of using our engineering expertise to support the internet for the benefit of humanity – a mission we take seriously;
- We maintain a strong and healthy global network of sophisticated users who provide regular feedback and direction on development; and
- We maintain robust systems for receiving vulnerability reports and taking appropriate and proportionate action in response.

We have found a variety of means to secure stable and recurring finance that we apply to the development and maintenance of our Open Source Internet Infrastructure Software, while preserving the characteristics of that software that promote an environment of permissionless innovation for the benefit of society.

# 3 Open Source Internet Infrastructure Software

## 3.1 Definition of Open Source Internet Infrastructure Software

We define Open Source Internet Infrastructure Software as any software[3] that satisfies all of the following characteristics:

- it serves predominantly to implement one or more open standards that describe core operational functions of internet WAN or LAN infrastructure such as DNS (RFC

[1] 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 (15 September 2022).
[2] We define this term below in Section 3.1. Open Source Internet Infrastructure Software projects we manage include: DNS (BIND9, Unbound, NSD, OpenDNSSEC, Knot Resolver, Knot), Routing (BIRD, FRR), Routing safety (Routinator, Krill), DHCP (ISC DHCP, Kea).
[3] Our definition is limited to software, as such.

1035), DHCP (RFC 2131), or routing infrastructure such as BGP (RFC 4271) or RPKI (RFC 6810);
- the software that embodies these standards is published under the terms of an open source licence as defined by the Open Source Initiative;[4]
- it is directed to, and intended solely for adoption and implementation by, technically sophisticated users (typically communications network service operators and equipment manufacturers); and
- it is not directed to, or intended for direct adoption or implementation by, consumers.[5]

This software is, by its very nature, mostly invisible to most internet users most of the time. They constitute important tools in the hands of sophisticated network operators and equipment manufacturers.

## 3.2 Factors that assure security and stability of Open Source Internet Infrastructure Software

Many Open Source Internet Infrastructure Software projects enjoy a strong history of security and stability. Others do not. This naturally raises the question of what factors contribute to security and stability.

### 3.2.1 Diversity of solutions

Maintaining the diversity of Open Source Internet Infrastructure Software solutions is a key element for assuring stability and security. Much akin to biodiversity, diversity of software and operations is a critical ingredient to resilience. As an example, network operators often adopt multiple independent software tools to avoid single points of failure in their ability to provide reliable internet infrastructure.[6]

### 3.2.2 Stable governance of the open source project

Open source software projects can be conducted under many different systems of management and control. Projects without stable governance have a very low chance of long-term survival, much less realise the maintenance required to keep its software secure and adapt it to changing requirements.

Security and stability rests, in part, on a clear system of management controls over how code is introduced to the definitive corpus of the software. Good stewardship costs time and effort and therefore money.

---

[4] OSI definition available at: https://opensource.org/osd.
[5] We use the term “consumer” in the way it is ordinarily used in European legislation to define consumer protection rights. I.e., in the context of the supply of a given product or service, a consumer is normally a person who obtains that product or service for purposes outside the scope of their business, trade, profession, or craft.
[6] The operators of the DNS root server system, for example, promote multiple independent open source implementations of DNS server software “to increase the diversity of software in the root name server system, the lack of which is widely considered to be a potential vulnerability.” (Email from Daniel Karrenberg, RIPE NCC and K-root operator, addressed to dns-wg, Feb 14, 2003. Available at: https://www.ripe.net/ripe/mail/archives/dns-wg/2003-February/000891.html)

### 3.2.3 An ethos of releasing software only when it is ready

The development path for this type of software does not follow the common model of software innovation that has dominated much of commercial software since the late 1970s: early product release encouraged by a profit-driven motive to secure market share. Because we are not driven by a need to monetize our software, we can take the time to do it properly.

The ethos surrounding development and maintenance of this software must instead be founded on a careful and considered accretion of functions and reduction of anomalies. To be successful (and trusted by their sophisticated user community), the developers and maintainers of this software must take a long-term view of their development activity and avoid creating incentives to rush the publication of revised features and functions.

### 3.2.4 Making source code available to a large community of sophisticated users

One of the hoped-for benefits of distributing source code is to enable more people to examine the code. Though more users does not mean more examiners, Internet protocols and their implementation in Open Source Internet Infrastructure Software have historically attracted the attention of (academic) researchers. By making the source code of Open Source Internet Infrastructure Software freely available for inspection it is delivered into the hands of people who are most likely (other than the developers) to identify the source of a problem.

### 3.2.5 Robust system to enable reporting and addressing vulnerabilities

Although open source software is naturally more amenable to finding and analysing vulnerabilities, maintaining a robust vulnerability reporting program remains an important element in maintaining security and the trust of the user community.

Addressing vulnerabilities in the Internet Infrastructure space can be challenging. It is important to maintain the relationship between the software (under the control of the manager/curator) and the standards which it implements (which are a living expression of the internet engineering community). As a result, it occasionally becomes expedient or necessary for multiple entities (both users and developers) to convene emergency work parties to address vulnerabilities in the underlying protocols, or multiple implementations. This sort of collaboration, once again, is significantly aided by the open source nature of the software.

## 3.3 The special challenge of recurring finance

Most of the factors described above that promote security and stability depend upon professional, timely, and recurring engineering effort and oversight. History teaches that this cannot be accomplished long-term by the efforts of volunteers alone, no matter how talented. This, in turn, means that somebody must pay for the effort.

By definition, the predominant core of Open Source Internet Infrastructure Software is not sold - it is given to the world for free. Similarly, maintenance is not sold - it is given away to the world for free. The challenge becomes how to secure financing without breaking the benefits of the open source model. Grant seeking activity alone is not sufficiently predictable.

Those who provide grants are typically more interested in funding new features and less interested in funding routine maintenance activity that is critical to promote security and stability.

Many entities that manage Open Source Internet Infrastructure Software projects have used a number of creative methods to secure recurring revenue. One increasingly popular model is to offer affiliated technical support or consultancy services to implementer/operators. Another emerging practice is to offer units of additional software with adjacent value-add functionality on a commercial subscription basis.[7] This preserves the open source freedoms of the core software that implements the open standard. A third practice is to cross-subsidize the development and maintenance by locating the development team within another organisation holding closely aligned goals and ethos that generates recurring revenue (such as a membership organisation).

# 4 How the CRA could jeopardise the security and stability of Open Source Internet Infrastructure Software

While we applaud the efforts of the European Commission to enhance the cyber security of products with digital components, we fear that the CRA could create a series of unintended adverse consequences to the security and stability of Open Source Internet Infrastructure Software - and by extension to the Internet.

**Problem 1: the CRA as applied.** We feel that the regulation as applied would impose disproportionate regulatory compliance burdens on developers and curators of “critical products” that will strain their existing capacity while failing to enhance the security or stability of this type of software. We address these concerns in Section 5.

**Problem 2: Scope of partial exemption for free and open source software.** We are concerned that the limited scope of the exemption concerning free and open source software would create unwelcome disincentives for some Open Source Internet Infrastructure Software developers and curators to professionalise their development activity. We are also concerned that the limited scope of exemption would create unwelcome incentives for some Open Source Internet Infrastructure Software developers and curators to move away from their not-for-profit open source model and towards a for-profit closed source model. We address these concerns in Section 6.

# 5 Concerns about the CRA as applied to organisations that manage Open Source Internet Infrastructure Software

In this section, we highlight a series of challenges presented by the application of the CRA to organisations that manage what is already secure and stable Open Source Internet Infrastructure Software. These challenges arise from uncertainty of interpretation and/or

[7] This is often described as an “open core” model. In this submission, however, we assume that (1) the functionality of the relevant open standard remains in the open core, and (2) the predominant purpose of the relevant software as a whole remains focussed on implementing the relevant open standard.

disproportionate regulatory burden that could destabilise rather than enhance security and stability of the software.

## 5.1 Proving compliance for our processes of secure development diverts resources from a practice we have run for decades and are fully self-motivated and incentivized to perform and perfect

We are not opposed to the codification of what are essentially high level industry best practices in EU law. Our organisations have decades of experience running vulnerability handling and remediation processes. Most of the Annex I *vulnerability handling requirements* have been our standard practice.[8] The same goes for the application of the Annex I *security requirements*, with one notable exception (covered as a concern later). Some of the stated requirements originated as best practices in the wider free and open source movement.

Our concern is with the cost of proving compliance. We worry that the burden of the conformity assessment procedures will divert resources from our actual practice of these *essential security requirements* that we already are fully self-motivated and incentivized to perform and perfect.

> **RECOMMENDATION:** We would welcome alternative regulatory approaches to encourage third parties to inspect our work, and benefit from the fact that our software and its development history are fully available for inspection. We would rather not divert our own limited resources that are already spent in a manner fully aligned with the security goals of the CRA.

## 5.2 Third party audits for “critical products” will be a costly burden unlikely to improve the quality of our software

If any of our Open Source Internet Infrastructure software is deemed a “critical product”[9] that we are “making available on the market”, we are very likely to incur costly third party process audits as part of the required conformity assessments. Third party process audits are a burden for us unlikely to improve the quality of our software and its security properties.

The exception explicitly designed to avoid involvement of third party auditors for Class I critical products in Art 24(2) is unlikely to help us or others in the Free and Open Source movements in practice. We are at present underrepresented in the EU policy[10] and

[8] The exceptions are the newer SBoM requirement in Annex I, Section 2(1) and the prescriptions on the timing and availability of vulnerability descriptions (Annex I, Section 2(4)) and security patches (Annex I, Section 2(8). We discuss our concerns with the latter two in Section 5.4.

[9] We note in passing that this is not clear to us. The definitions in Annex III are difficult to apply with respect to some items of Open Source Internet Infrastructure Software. We note generally that Annex III would benefit from significant additional guidance on interpretation.

[10] The main body of CRA’s impact assessment report contains only one mention of open source, which (ironically) does not take account of the CRA’s impact on open source. (SWD(2022) 282 final) Annex III of that assessment makes the sweeping (and debateable) observation that “According to the literature, it is in principle possible to segment the open source software (OSS) market into commercial open source and non-commercial open source.” “The literature” in question, however, appears to be a single paper written by a researcher at a well-known vendor of closed-source ERP

standardisation processes.[11,12] This underrepresentation in the EU standards processes will negatively affect the availability of harmonised standards, common specifications or certification schemes that are applicable to Open Source Internet Infrastructure Software. In turn, these will therefore be unavailable for use in self-certification. The only alternative left available are the conformity assessment procedures that involve paying for third party process auditors.

Business process audits are a very poor match for open-source projects not run by large commercial enterprises for three reasons. First, audits (and auditors) are very much geared towards traditional businesses and business processes, which do not necessarily align well with how open-source projects operate. Second, legal, compliance and auditing are skill-sets not necessarily present in groups of open-source developers who are otherwise well equipped to develop secure and stable software. Third, audits can be prohibitively expensive for organisations. Compliance costs can be prohibitive for projects that (partially) rely on volunteers and/or are (partially) sustained by donations, paid features or in-kind support from the users of their software.

> **RECOMMENDATION**: We have already outlined above the key factors that enhance security and stability of our software. We suggest that in the circumstances we describe, we should be assured of some method that allows us to self-assess compliance regardless of the availability of EU standards applicable to Open Source Internet Infrastructure Software.

## 5.3 A requirement to fix all “known exploitable vulnerabilities” without regard to severity would skew engineering effort into tasks with increasingly diminished security impact

Annex I, Section 1(2) prohibits the delivery of software with “known exploitable vulnerabilities.” The term “known exploitable vulnerabilities,” however, does not appear to be defined.[13] As a result, this appears to encompass all security vulnerabilities that are both known and can be exploited (ie. leveraged by a threat actor) regardless of severity.[14]

---

software. This does not, in our opinion, constitute sufficient representation of those who are most familiar with the challenges of developing and maintaining open source software, and the benefits that it provides to society.

[11] Standardisation for Internet protocols, their implementation in software and their operations by practitioners happen predominantly in multi-stakeholder forums such as the Internet Engineering Task Force, the RIR communities and Network Operator Groups. We are very active in that space.

[12] Sidenote: the open hardware movement is even more underrepresented than we are.

[13] In contrast, the proposal itself defines the term “actively exploited vulnerability” which we understand as the class of arbitrary code execution security vulnerabilities which is a specific, more narrowly defined (and more severe) subclass of all security vulnerabilities.
Article 3(39): “*‘actively exploited vulnerability’ means a vulnerability for which there is reliable evidence that execution of malicious code was performed by an actor on a system without permission of the system owner”*

[14] Such a reading would be consistent with the meaning of “exploited” in the definition of “vulnerability” in NIS2 Article 6(15): “*‘vulnerability’ means a weakness, susceptibility or flaw of ICT products or ICT services that can be exploited by a cyber threat”*

The result is a prohibition on delivery of software with any known vulnerabilities capable of exploitation under any conditions (no matter how unlikely), and without any regard to the real-world impact.

This sort of unconditional requirement, untethered from any real-world risk assessment, would force developers to spend engineering effort without regard for effectiveness of that effort. This seems directly opposite of the CRA's stated goal of having "manufactures take security seriously".

> **RECOMMENDATION**: We suggest revising Annex I, Section 1(2) so that the security requirement concerning known vulnerabilities rests upon a risk based assessment by the relevant economic operator.

## 5.4 Taking away our ability to provide security patches ahead of public availability to operators of critical internet infrastructure

Annex I, Section 2(8) requires that dissemination of security patches happens without delay. We observe that the users of our Open Source Internet Infrastructure Software operate services with widely varying levels of societal impact of outages. Taking away our ability to provide security patched ahead of their public availability to operators that provide the Internet's most critical functions[15] based on our professional judgement of societal impact may prove detrimental to the requirement's goals.

> **RECOMMENDATION**: We suggest revising Annex I, Section 2(8) to allow for staggered release of security patches when not doing so would put Internet users at greater risk.

## 5.5 Overly broad and novel reporting obligations (Article 11)

We agree that product vulnerability reporting is an important aspect of maintaining software security and stability. This is why we operate robust systems for receiving vulnerability reports and addressing vulnerabilities in a timely fashion.

Many of the additional obligations that would be imposed by Article 11 upon persons in their capacity as a manufacturer, however, seem unfounded and counterproductive when applied to manufacturers of Open Source Internet Infrastructure Software.

### 5.5.1 Obligations to report incidents related to third party use of our software conflicts with our role in vulnerability remediation

We are concerned that the new and overly broad obligations introduced by Article 11 would move software manufacturers from the realm of product vulnerability management (a natural activity for software developers and maintainers) into the process of security incident management (a role which, of necessity, involves those who use software to provide

[15] Critical functions such as the DNS Root Servers, TLD registries or RPKI trust anchors.

services). This expanded role would create tensions that hinder the cause of remediating security problems.

Article 11(1) imposes an obligation on manufacturers to notify public authorities quickly (and within 24 hours) of any "actively exploited" vulnerability. While vulnerabilities can be assessed by examining the product itself, the status of "active exploitation" depends on knowledge of operational use.[16] It specifically depends on having "reliable evidence" that an exploit occurred "without permission of the system owner."

We imagine that some software manufacturers might infer active exploitation from software telemetry reports[17] (especially software intended for adoption and use by consumers and other non-sophisticated end users). Manufacturers of Open Source Internet Infrastructure Software, however, do not typically introduce software telemetry functions from which such exploitation could be inferred[18] and are most likely to learn about active exploits from one of two possible sources: (1) published reports (which are readily available to the public and available for review by authorities); and (2) confidential disclosures from sophisticated end users who are requesting assistance to remediate an active incident which they are managing.

This reporting requirement would place the manufacturer into the insidious position of being required to report information related to active incidents managed by third parties, which could conflict with that third party's own reporting procedures.[19] We fear that this obligation could discourage end users of Open Source Internet Infrastructure Software from working with the very people who may be most able to assist them in resolving their incident and prevent others from falling victim to exploitation.

**RECOMMENDATION:** Revise Article 11 so that a manufacturer's reporting obligations concerning vulnerabilities in its own product does not extend to reporting details of third party incidents.

**RECOMMENDATION**: Revise Article 11 so that a manufacturer can defer reporting to avoid conflict with third party user reporting obligations.

**RECOMMENDATION:** Revise Article 11 to clarify that a manufacturer's obligation to report vulnerabilities to ENISA (or any other public agency) can be fulfilled by publishing vulnerability reports in the normal fashion such as the CVE reporting system.

[16] This is supported by the definition of "actively exploited vulnerability" found in Article 3(39): "a vulnerability for which there is reliable evidence that execution of malicious code was performed by an actor on a system without permission of the system owner" (emphasis added).
[17] Which seem to fall into the definition of "remote data processing solutions" in Article 3(2).
[18] These would be rejected and/or disabled by sophisticated users of this software as a security threat or intrusion into confidential network operations data.
[19] For example, an obligation to report incidents under Article 23 of NIS2.

### 5.5.2 Obligation to report manufacturer's security incidents is not conditioned on risk-assessment; over-reporting is burdensome and of little value

Article 11(2) would obligate all manufactures to notify ENISA about "any incident having impact on the security of [their] product". This appears to include incidents suffered and managed by the manufacturer itself (such as an intrusion into the manufacturer's own technological infrastructure).

The trigger for reporting, however, does not seem to take account of relevant risk created by the incident. This appears to impose an obligation to report all first party incidents, no matter how trivial. Further, there is nothing that appears to limit this concept of a reportable "incident" to a breach of technological controls. On its face, it would seem to apply to events that place operational controls at risk (such as unanticipated staff changes).

This would appear to bring all software product manufacturers into a first party incident reporting regime that is more aggressive than dictated by NIS2. Such an obligation would produce an unnecessary reporting burden and a body of reports of questionable value.

> **RECOMMENDATION:** Revise Article 11(2) to clarify that manufacturers are only obligated to report first party incidents that present a material risk to the security of their software product.

## 5.6 Other issues

In addition to the concerns expressed above, there are a number of additional issues within the CRA that would benefit from clarification.

### 5.6.1 Uncertainty related to substantial modification of software triggering conformity reassessment (Recital 23)

It is unclear to us what type of software release constitutes a "substantial modification" (Article 3(31)) that requires compliance verification or conformity re-assessment (Recital 23). Even the stable versions of our software receive frequent maintenance updates. We are concerned that delaying their delivery will not benefit security and stability.

> **RECOMMENDATION:** Revise Recital 23 and/or Article 3(31) to clarify that "substantial modification" is a risk-based measurement. I.e., a modification should be considered "substantial" only if it creates a material risk of jeopardising the validity of the underlying conformity assessment.

> **RECOMMENDATION:** Revise Recital 23 to clarify that routine software maintenance activity would not normally constitute "substantial modification" triggering the need to re-verify product compliance.

### 5.6.2 Time-limited availability of software for testing purposes is incompatible with common open source software practice (Article 4(3))

The requirement that "software is only made available for a limited period required for testing purposes" in Article 4(3) is incompatible with the practice common in the Free and Open Source movement to make the version control system containing the full development history of software publicly accessible, and provide (packaged) software builds for pre-release versions for testing purposes.[20]

> **RECOMMENDATION:** Revise Article 4(3) to exempt open source software from being required to be "only made available for a limited period required for testing purposes".

# 6 Risks arising from the overly narrow open source exemption (Recital 10)

The CRA would exempt some, but not all, open source software development and supply activity from regulatory coverage.[21] We acknowledge and appreciate that the European Commission created an exception at all. We now focus upon the specifics of the exemption and its implications.

## 6.1 Overly expansive interpretation of "commercial activity" leads to overly narrow scope of exemption

To qualify for the open source exemption currently, the relevant development or supply activity must fall "outside the course of a commercial activity." To the untrained eye this might appear to exempt development and supply of all Open Source Internet Infrastructure Software because this software is given away and not "sold". Sadly, this is incorrect.

The regulation clarifies that the concept of "commercial activity" is incredibly wide. It includes activity as limited as "charging a price for technical support services." Thus this exemption, which we feel is important, would not apply to the project maintainers who have created a limited but stable method to finance the very activities that make their projects more secure and stable while also retaining the open source characteristics of the software that underpin trust and confidence in the software.[22]

---

[20] For example, this web site contains an archive of 20 years worth of open source releases from ISC https://ftp.isc.org/isc/.

[21] Recital 10: "In order not to hamper innovation or research, free and open-source software developed or supplied outside the course of a commercial activity should not be covered by this Regulation. This is in particular the case for software, including its source code and modified versions, that is openly shared and freely accessible, usable, modifiable and redistributable. In the context of software, a commercial activity might be characterized not only by charging a price for a product, but also by charging a price for technical support services, by providing a software platform through which the manufacturer monetises other services, or by the use of personal data for reasons other than exclusively for improving the security, compatibility or interoperability of the software." (emphasis added)

[22] See discussion above at Sections 3.2 & 3.3.

## 6.2 Impact of the narrow scope of the exemption

We fear that the overly narrow definition of this important exemption will produce a series of incentives that will jeopardise security and stability of Open Source Internet Infrastructure Software.

### 6.2.1 Disincentive to professionalise development and curation

There are a wide variety of projects that seek to manage the development and curation of Open Source Internet Infrastructure Software. Some are better managed than others. One recurring problem in this field concerns the growing burden suffered by all-volunteer development teams. Their efforts, begun with the best of intentions, become an unfunded mandate.

Teams in this position should be given every incentive to seek sustainable sources of financial support. The narrow scope of the exemption in Recital 10, however, does the opposite. It tells developers in this position that the easiest way to avoid regulatory burden is to continue to close their eyes to the need for financial stability. Worse, it cautions such developers against selling any technical support services that might somehow be connected to the software they developed for fear that this will cost them their regulatory exemption.

The narrow scope of this exemption would penalise the very groups who have found a way to develop modest recurring (albeit nonprofit) revenue sources[23] such as the supply of technical support services or small value-add software additions. Both groups develop software that underpins the operation of the internet. Imposing regulatory burdens on one group, but not the other, creates the wrong incentives for the long-term health of this software.

### 6.2.2 Incentive to move away from open source nonprofit model

So what is a responsible developer of Open Source Internet Infrastructure Software to do? Faced with increasing regulatory burden, the development team will need to find some way to create significant additional revenue to fund the compliance burden.

Raising the price of technical support agreements or adjacent software functions is one solution, although it is not clear how far this can be extended. While the technical staff of large user organisations may appreciate the benefit of a commercial technical support arrangement, it often takes tremendous effort to persuade the finance director of such entities to pay for such services with respect to software described to them as “free.”

Responsible developers face increased regulatory pressure and expense to do something to secure their future. Some may attempt to migrate away from the open source model as their only certain way to make enough money to pay for the regulatory mandates. Sadly, moving away from open source will most likely harm security and stability.[24]

[23] We emphasise that these revenue streams are relatively modest compared with the revenues that can be generated through rent seeking sales of software licences or the sale of maintenance services for a closed source product.
[24] See above at Section 3.2.3.

### 6.2.3 Risk of harming product diversity and reducing innovation

Faced with increasing regulatory compliance burdens, some project maintainers may abandon projects (to avoid the compliance burden entirely) or seek to merge with others who are similarly situated (to share the compliance burden). Both approaches create the risk of loss of product diversity, one of the key elements that enhances security and stability in this space.[25] Open Source software helps drive the development of new technical standards (to secure the Internet infrastructure). As new standards are proposed, the Internet Engineering Task Force looks for examples of open source implementations as evidence that the standard is ready for adoption.

## 6.3 Resolving the scope of the open source exemption

We suggest two alternative solutions to address the scope of this regulatory exemption. Either would remedy the problem we describe while preserving the important policy goals of the exemption.

**RECOMMENDATION**

**ALTERNATIVE 1**: We suggest exempting all open source development and supply activity from the scope of the CRA without exception. This is the most simple solution to the problems described in this section.

**ALTERNATIVE 2**: We suggest clarifying the interpretation of “commercial activity” in a way that more appropriately fits the context of Open Source Internet Infrastructure Software. This would acknowledge that not all efforts to produce recurring revenue (such as paid-for “technical support services”) should be characterised as “commercial activity”.

**ALTERNATIVE 3:** Make the exemption from coverage subject to additional conditions that help to promote security and stability of Open Source Internet Infrastructure Software[26] such as:

(a) the predominant purpose of the subject matter software is to implement widely recognized open standards (such as the RFC series);

(b) the development or supply of the subject matter software is conducted on a not-for-profit basis; or

(c) the developer, supplier or maintainer of the subject matter software operates an appropriate system for reporting and resolving vulnerabilities.

[25] See above at Section 3.2.1.
[26] See above at Section 3.2