Vrijschrift.org is a Dutch NGO that strives to defend freedom of expression, free software and to expand the public domain as well as knowledge as a commons.

Introduction
The introduction of liability rules for security flaws in software has been long overdue. Vrijschrift.org welcomes initiatives of this nature, including this draft proposal of the European Commission for a Cyber Resilience Acte (hereafter: CRA). This is something that urgently needs to be done, and needs to be done well, so in that sense it is very good that this is finally on the legislative agenda. As for the nature of the regulation: this proposals is modeled around existing European models for product conformity, of which the CE markings on a lot of products are the most noticeable in practice. This is also where our concerns lie, since software is not necessarily in the same league as USB-chargers are. So some concrete suggestions on our concerns and on how to improve this proposal.

Software as a form of expression
First of all, the EU legislative bodies should acknowledge that software, especially software whose licenses explicitly acknowledge the right to inspect and improve the source code, is a form of expression. This means that when the EU regulates software, it needs to find a better balance of interests when they compete than is the case with mere product conformity regulation. It does not make for good legislative techniques to do so in the recitals and not expand on it in the substantive articles. It has always been a core tenet of European fundamental rights law that interference with fundamental rights such as freedom of expression, such interference should be clearly delineated by law. To put it simply: restrictions of freedom of expression must be knowable and predictable to citizens. Recitals in legislation are not good enough to achieve that, and neither is handing down the power to make delegated acts to the Commission to further interfere with this particular freedom without striking some explicit balance between the fundamental right and the urgency of the social goal for doing so.

Unclear definitions in the CRA
Secondly, when defining a notion of commercial activity regarding open source software, the CRA provides examples of the creation and/or use of open source software that are typically understood as non-commercial but would fall under this definition of commercial activity. For example, if a small civil society organisation uses open source software to allow people to RSVP for one of their events (a remote data processing solution, under the CRA), it would fall under the CRA's definition of commercial activity. It is quite obvious that that organisation must safeguard the personal data of the event attendees as prescribed by GDPR. Under the proposed text of the CRA, however, the NGO would be obliged to obtain a CE marking (mandatory conformity marking for regulating the goods sold within the European Economic Area) for the software being used, which would make using such software virtually impossible in practice.

Not to mention that the poorly placed definition of commercial activity in the recitals contains circular reasoning of the type that, if it were to happen in software, would be considered a hidden defect. As a rule, CE markings are not an instrument that is fit for purpose in the context of stand-alone software. And even in the context of embedded software it is fraught with issues that typically do not occur with other aspects of physical products. Mainly because well-maintained software involves updates and patches that have no equivalent in the physical world and make for a much more dynamic situation than CE marking (self-)certification processes allow for.

An alternative approach would be to introduce product liability for security flaws in software when no patches are made available by the publisher for free or other costs attached (such as loss of functionality) and there is no source code available nor the right to change the source code. This would put the open source software ecosystem in the clear.

Coherence with other legislative proposals
Lastly, the CRA adds a new patch to a quilt of regulations that cover the concept of product liability for software in general. While it is understandable that software in automotive applications and in medical equipment live under different regulatory schemes (both are specifically exempted from the proposed CRA), those same pieces of code could also be used across various other sectors in ways not foreseeable by the publisher.

There are a great many tools, libraries and pieces of software made available under open source licenses, often by small businesses or small groups of developers that will in practice be covered by the CRA as well as various sectoral regulations, the upcoming general product liability regulation and possibly also the AI Act and the AI Liability Act.

This creates legal uncertainty for developers and is bound to result in a chilling effect on the very ecosystems that have often acted more responsibly, by and large, in dealing with security defects in their software than proprietary actors.

A better approach would be to integrate the matter of security flaws in software in a coherent framework that takes matters of source availability and the right to alter software into account as well as other types of defects. Doing this piecemeal and per industry creates avoidable barriers for entry in an industry that always has had the benefit of having little in the way of barriers of entry.