In Open We Trust?

Ashwin Ramaswami

Published on: Jun 01, 2023

URL: https://commonplace.knowledgefutures.org/pub/ktd793z4

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)
"Who can afford to do professional work for nothing? What hobbyist can put three-man years into programming, finding all bugs, documenting his product, and distribute it for free?"


In 1976, an irate Bill Gates wrote the above passage to discourage personal computer hobbyists from making copies of his company’s Altair BASIC software without paying for it. Forty-seven years later, it is Microsoft that has ironically become a major proponent of the notion that software can be distributed for free through licenses that grant users extensive freedoms to use, modify, or distribute it: free and open source software (FOSS).

Microsoft’s transformation is but indicative of a seismic shift in the software landscape where major corporations have come to embrace FOSS, government agencies are developing software in the open by default, and social code platforms such as GitHub and Hugging Face permissively disseminate datasets, source code, and AI models.

Today, free and open source software has proven a marvelous achievement. Open source licenses have challenged the dominant intellectual property regime of exclusion by allowing developers to freely contribute to a software commons and together create public goods of value. Developing software in the open enhances opportunities to improve transparency, security, and innovation, and even establish a competitive advantage. Much of the digital infrastructure we all rely on, from Linux to Apache to Android, is FOSS and as vital to our critical infrastructure as roads and bridges or water systems.

In the span of decades, open source has gone from a seemingly quixotic or “hobbyist” movement to a major force of innovation and collaboration that is embodied both by our digital infrastructure and the people and communities who build it. Yet while open source software makes up the foundational infrastructure of the Internet, large-scale security incidents have pointed to the importance of sustaining the open source ecosystem. As we encounter and adopt new technologies, from cloud computing to generative AI, both technologists and policymakers inevitably have to consider the benefits and harms of making such technologies open and widely available. Ultimately, we need software and infrastructure that we can trust to be safe, functional, secure, and sustainable. The continued success of FOSS thus comes down to one question: How do we build trust in open source?

**Underlying Asymmetries**

Building trust in the open source ecosystem involves addressing three asymmetries around how FOSS is built, used, and understood. First, an imbalance in the nature of production versus consumption of FOSS constitutes labor asymmetries that neglect the role of people and communities in producing open source software. Second, a lack of understanding around security, sustainability, and dependencies in the open source ecosystem creates information asymmetries that hinder efforts to responsibly use or strengthen it. Finally, a developer’s
lack of control of how people use an open source tool causes **usage asymmetries**, which can exploit openness to widely spread abuses or harms. Addressing each of these asymmetries requires action from various stakeholders in the open source ecosystem, from developers who build open source, users who consume it, researchers who study it, and policymakers who govern it.

To be clear, these asymmetries are not unique to open source software. The fact that software is FOSS does not make it inherently less trustworthy, less secure, or more susceptible to abuse. Many mitigations described here may equally apply to closed-source software as well. That said, there are two reasons why it is helpful to focus here on the open source component. First, building trust in FOSS requires an approach informed by an understanding of what makes open source unique in how it is produced and used. Second, the open source ethos of transparency and collaboration can inspire novel opportunities to strengthen trust.

**Labor**

There are a plethora of anecdotes illustrating the labor asymmetries inherent in open source production, in which developers who put large amounts of work to maintain an open source library struggle to obtain compensation to sustain their work. At its core, these are caused by a mismatch between production and consumption. It only takes a few people to build an open source software tool, which can be distributed at near-zero marginal cost to become a public good that is used by organizations around the world. A now-iconic xkcd comic sums up this mismatch: “all modern digital infrastructure” depends on “a project some random person in Nebraska has been thanklessly maintaining since 2003.”

The fact that FOSS is freely available as a public good obscures the people and communities whose labor produced it and whose health is crucial to sustain it. This opens the door to two risks: users of FOSS may free ride on open source developers’ labor by using their software without giving back to the community, and consumers—who may far outnumber developers—can therefore magnify the workload of developers merely by submitting bug reports, feature requests, or issues for maintainers to address. Overwhelmed maintainers may not be able to focus on tasks such as security updates, even as security standards increase with more users. This problem is especially pernicious when the users of open source software are corporations that, in essence, profit off the labor of open source developers without giving back to sustain their communities.

To address this problem, we must center people and communities and the human labor required to produce open source software:

**Incentivize giving back to open source**: Though companies play an important role giving back to open source, it is not enough to solely fund FOSS on a discretionary, philanthropic basis. Rather, commercial FOSS users should ultimately internalize these costs as a necessary part of security and compliance. For example, companies could be ranked based on how well they support their open source dependencies — incentivizing “ethical software supply chains.” Or a change to the liability regime for commercial software vendors could ensure they adequately support the open source communities they rely on before bringing a
product to market. Importantly, any financial support must be respectful of the existing governance structure and preferences of open source communities. If the project is not comfortable with funding, there are other ways a company can give back, such as allowing employees to contribute to a project during work time or joining an open source foundation to focus on systemic efforts.

**Support and sustain communities:** Too often, efforts to fund open source software prioritize the development of new features. But non-development roles are crucial to a FOSS project’s sustainability, from community managers to documentation writers; and so are rote maintenance tasks such as code rewrites and dependency upgrades. Funding efforts must include these priorities in order to empower open source communities and people in the long term. More research on understanding open source communities can also help inform such support.

**Build infrastructure to empower maintainers:** Automated tooling can help reduce the burden on individual maintainers by allaying the power imbalance of having to deal with users’ issues at scale. For example, we could invest in tooling that facilitates fuzz testing or remediating vulnerabilities in open source software; software that facilitates moderation, triage, and community management of user-submitted issues; or frameworks and tools that make it easier to create and govern an open source organization.

**Center and organize labor:** Open source developers and maintainers are at the heart of what makes FOSS tick. We should create more ways to recognize and compensate the efforts of open source maintainers, whether through awards or fellowships. We should invest in education for open source developers, such as an “open source school” with courses on how to manage contributions and communities and grow sustainable open source projects. Could developers even take inspiration from organized labor to draw attention to neglected open source projects, such as an “open source strike” where issues and feature requests go temporarily unanswered? Developers at companies may also be key to encourage their employers to change their FOSS consumption habits: as Coraline Ada Ehmke puts it, “[w]e as developers do have the ability to exercise some moral authority and we have the ability to decree meaningful consequences for the corporations.”

**Information**

Information asymmetries in the FOSS ecosystem mean that people do not understand enough about the open source ecosystem in order to more effectively use and support it. Unlike commercial vendors who sell software to specific customers, open source maintainers often don’t have visibility into who uses their software for free, and who is using outdated or vulnerable versions. Conversely, consumers of open source software may not have visibility into what OSS dependencies they are using, which ones are secure or sustainable, and how they can give back to them. Finally, funders and policymakers may not understand the importance of open source communities or know which OSS tools are critical to the ecosystem, precluding them from identifying systemic risks and efficiently mitigating them.

To solve this, we need to encourage more transparency:
**Share more data:** Open source consumers should adopt and publish Software Bills of Materials (SBOMs) to share which packages they depend on. Centralized repositories such as package managers and source code hosting providers can share, contingent on privacy protections, additional data around package and version downloads and usage patterns. Such data will allow creating dependency graphs of the entire ecosystem, which lets us identify which dependencies are used where. And it’s not limited to software: for example, for the AI ecosystem, dependencies include datasets and AI models as well. Platforms should also encourage developers to include more information to increase trust in what they publish, from code signing that verifies where the code came from to greater transparency on data used to train AI models.

**Develop metrics:** As we collect more data, we need to interpret and operationalize it to help ecosystem actors make better decisions. Improving and developing new metrics can help. For example, we can measure the health and sustainability of communities through the CHA OSS project’s tools; rank the most critical libraries using a criticality score; and use this data to directly inform processes around open source dependency selection or project funding prioritization. Though metrics necessarily cannot capture everything, they are a good starting point to interpret data at scale and should be complemented by a qualitative review when informing decisions.

**Educate the public:** Open source software is used by everybody, and trust issues in open source affect everyone. Anyone who buys software or uses a computing device needs to understand what open source software and communities are: the labor that underlies our digital world. A public understanding of the labor behind our digital infrastructure is crucial to ensure our policymakers make the best decisions around it. Could a brief introduction to open source software be included in computer science or even civics curricula in schools?

**Usage**

Finally, usage asymmetries concern the fact that open source developers may not be able to control how a tool is used. For example, an open source developer could release a proof-of-concept for a vulnerability that is then used by others to commit computer crimes, or a developer could widely release an open source AI model that is used by others to generate disinformation or commit fraud. To be fair, perhaps all technology is inherently dual-use. But unlike other technologies, source code’s expressive nature makes it harder to regulate its use and distribution and easier to apply freedom of speech protections to it. Regulation also has to consider the risks of stifling collaboration and innovation within open source communities. But there are practical actions we can already take to build in safeguards and values into open source software:

**Reclaim agency over technology’s development:** The fact that FOSS is freely available to be used for any purpose may make it easier for developers to disclaim agency over how others use it; after all, anyone can fork your code and do anything with it. But in fact, it’s the other way around. Technology is not neutral; it is ultimately political, reflecting human values, whether consciously or unconsciously added.
FOSS is so freely available, the kinds of open source technologies that we invest in can shape the entire ecosystem by making certain types of tools easier to use and more accessible. If we want a safer and more trustworthy ecosystem, we should create and invest in FOSS tools that support this goal—such as trust and safety tooling, guardrails and interpretability toolkits for AI systems, and tooling for content authenticity and provenance.

**Add friction and use defaults:** Software’s very design creates affordances that influence how it is used. One way to do this is to add friction—making certain actions harder—and use defaults—making certain actions easier. On the one hand, user interface design choices known as “dark patterns” use friction and defaults to deceive and mislead consumers. On the other hand, some tools already use friction and defaults to prevent harms, from safety and accessibility warnings to secure-by-default efforts to forwarding limits on messages. These changes are especially potent for open source libraries, because they are used so widely and have established *de facto* standards across the industry, from Hadoop to Linux to Kubernetes.

This is an underexplored area of change, with many questions that need more research. For example, how effective was Mastodon’s decision to limit full-text search to prevent “negative social dynamics”? Can major AI open source libraries include safety features to make it harder to abuse them to create disinformation? Can major open source libraries extend “security-by-default” to promote values such as “accessibility-by-default,” “veracity-by-default,” or “safety-by-default”? Even though a few determined “superusers” may be able to get around friction and defaults, will these changes nevertheless prevent the broad majority of harms? And how much are such changes limited by switching costs: when does a safety change becomes so burdensome that users decide to fork the library and remove the change? What does the outsized power of open source developers mean for how FOSS projects should think about their own governance?

**Licenses and Liability:** A final effort worth mentioning is on the legal side. Licenses both enabled FOSS in the first place and are now being used to try to tackle usage asymmetries. For example, the Ethical Source movement and Open & Responsible AI licenses aim to prohibit certain uses of code or AI models, respectively. While such legal initiatives are creative and may have promise, they should ensure they don’t harm the collaboration and openness that enables open source software communities to thrive. Secondly, efforts to change the software liability regime may be able to hold accountable users who abuse FOSS. Again, though, such efforts should take care not to disrupt open source collaboration, for example, by putting onerous compliance requirements on open source community developers. Rather, the software vendors or end-users who use and deploy FOSS should be responsible for the consequences of how they use it.

**Conclusion**

Open source is not leaving us, and it has created a wonderful system of infrastructure and tools that we all rely on today. The best way to build trust in the FOSS ecosystem is not to further close it off: rather, we should adopt an approach inspired by the open source ethos to mitigate the three asymmetries of labor, information,
and usage. Such a course of action requires action from all stakeholders in the ecosystem: consumers, developers, researchers, and policymakers. Such a course of action is required to ensure we continue to benefit from the fruits of openness while ensuring we are responsibly embracing and trusting it.

---

**Ashwin Ramaswami** is an open source maintainer, developer, and policy researcher. He also works on web application architecture and cybersecurity. He holds a B.S. in Computer Science from Stanford University, where he was the first CTO at the Stanford Daily, and is currently pursuing a J.D. degree at Georgetown Law.

### Footnotes


0. Steven J. Vaughan-Nichols, Oct. 11, 2020, “How Microsoft went from ‘Linux is a cancer’ to ‘Microsoft Loves Linux’.” —


0. Though not all open source software is maintained in this way—some OSS is maintained by large, well-resourced companies or foundations—those projects are not the focus of this section. ↩

0. Randall Munroe, “Dependency,” xkcd. ↩


0. Such funding is both too small and too volatile. For example, see the recent layoffs of corporate open source teams, Matt Asay, Jan. 30, 2023, “Google blew it with open source layoffs,” InfoWorld. ↩

0. Another model involves companies paying for a security contract with the maintainers of the open source libraries they rely on, which is Tidelift’s model. ↩


0. For examples of some work in this space, see Open Technology Fund, Jan. 30, 2023, “Open Technology Fund Announces ‘Free and Open Source Software’ Sustainability Fund”; Open Source Technology Improvement Fund, “The OSTIF Mission.” ↩

0. For some examples of such efforts, see “Digital Infrastructure Incubator,” Code for Science & Society; “Digital Infrastructure Research and Development,” Digital • Infrastructure • Fund. ↩

0. For some early efforts at fellowships, see Adrian Woodhead, January 19, 2022, “Our First Dan Kaminsky Fellow,” HUMAN; Bogomil Balkansky & Lauren Reeder, May 4, 2023 “New Fellowship: How Sequoia is Supporting Open Source,” Sequoia Capital. ↩

0. For some initial efforts in this space, see: Sumana Harihareswara, “Growing Your Contributors,” Open Source Collective. Abby Cabunoc Mayes, April 4, 2023, “Calling all open source maintainers,” GitHub Blog. ↩


0. Efforts such as Ecosyste.ms, libraries.io, and deps.dev are doing preliminary work around dependency graphs in open source. ↩


0. Mike Hanley, June 4, 2021, “Updates to our policies regarding exploits, malware, and vulnerability research,” GitHub Blog; Grant Robertson, August 18, 2021, “Deepfaking and Open Source: How Anyone Could Give a Queen’s Speech,” Digit News.


0. Shelley McKinley, March 10, 2023, “Don’t leave developers behind in the Section 230 debate,” GitHub Blog; Alex Engler, August 24, 2022, “The EU’s attempt to regulate open-source AI is counterproductive,” Brookings.


0. Mirko Boehm & Davis Eisape, “Standard setting organizations and open source communities: Partners or competitors?”, First Monday.


Fuzz testing or fuzzing is an automated software testing method that injects invalid, malformed, or unexpected inputs into a system to reveal software defects and vulnerabilities. A fuzzing tool injects these inputs into the system and then monitors for exceptions such as crashes or information leakage. (Source)