MITCHELL | HAMLINE School of Law

Cybaris®

Volume 12 | Issue 1 Article 3

2021

The General Data Protection Regulation and Open Source Software Communities

Amye Scavarda Perrin

THE GENERAL DATA PROTECTION REGULATION AND OPEN SOURCE SOFTWARE COMMUNITIES

Amye Scavarda Perrin[1]

Table of Contents

[1] The author is a Mitchell Hamline Student with an expected graduation date in July 2020. Her biographical information may be found on LinkedIn at: https://www.linkedin.com/in/amyescavardaperrin/

Introduction

The EU's General Data Protection Regulation (GDPR) came into effect May 2018, and most companies did not have privacy policies in place for compliance with the law. Specifically, many companies in the United States were initially surprised that the GDPR was not limited to doing business directly in the European Union, but extended to European citizens all over the world, no matter their actual residency.[2] Further, this set of regulations extended to any personal data collected, with or without consent from the consumer. However, another group was also surprised by this: open source software communities. Despite the fact that general open source licenses explicitly disclaim liability, open source communities and projects have a relationship with personal data given current development models, and are subject to the GDPR.[3] However, given the history and the collective interaction of the open source communities through a culture of openness, compliance with the GDPR creates a conflict, balancing the previous methods of open contribution attributed to authors and allowing for private information at the same time. The conflict between the desire to be global citizens in cross-national communities that transcends borders and compliance with an overall rule leads to confusion, misunderstandings, and unclear outcomes. This confusion has created a chilling effect on collaboration, causing individuals to cease work on collaborative

[2] Regulation 2016/679, of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data and Repealing Directive 95/46/EC, 2016 O.J. (L 119) 1 [hereinafter General Data Protection Regulation].
[3] *See Legal:Privacy Policy*, FEDORA COMMUNITY, https://fedoraproject.org/wiki/Legal:PrivacyPolicy (last visited Dec. 1, 2019) (a good example of a privacy policy discussed in depth later on) [hereinafter *Fedora Community*].

projects and providing community services, while raising future questions about how to design systems to be able to work openly in a privacy focused world. The GDPR, while overall a positive step for privacy regulation, has left confusion and unclear legal guidance for open source community members seeking to collaborate.

This article explores the tension between privacy regulations and open source software communities. Part I will cover the genesis of open source arising out of copyright concerns and provide background for the current state of modern open source software, as well as describe how businesses were created in connection with open source. Part II moves towards how our current tooling evolved and how that work speaks towards the current level of transparency in open source along with the current expectations of development. These modern software development practices are particularly important because of the cultural norms surrounding open source software development. Part III provides both an overview of the GDPR requirements and applies the problem set of the regulation to modern open source software communities, with special attention called to personal information as part of development. There are distinct advantages to the application of privacy that are not limited to open source. However, within open source communities, the fear of sharing data and collaboration has caused confusion and has created chilling effects on collaboration. While the historical records and legitimate business purpose clauses provided by the GDPR give guidance to lawyers seeking to understand open source development and what is or is not in compliance, open source communities are not just about code. The code is the output; it gets created through collaboration. Part IV provides an example of a community that created a privacy policy to comply. Ultimately, the GDPR is valuable because it provides a valuable legal

framework for privacy & consumer expectations, but the conflict within open source communities is ongoing and currently unresolved with the existing form of the law.

Part I: The creation of open source

At its heart, open source is a way to use copyright laws to serve in favor of transparency rather than proprietary or (closed) software. Copyright is one of the four areas of intellectual property law, in addition to trademark, trade secret, and patent law.[4] Copyright law protects an author's right to license and allow the reuse of a work, such as distribution or reproduction.[5] Copyright protection is outlined in the Constitution in order "[t]o promote the Progress of Science . . . by securing for limited Times to Authors . . . the exclusive Right to their respective Writings…."[6] Thinking of copyright as a series of various rights and limitations, the series seeks a balance between various competing interests.[7] Authors, publishers, and users all have widely different needs, and copyright protection should seek to be able to find a balance between all of these.

[4] HOWARD B. ABRAMS & TYLER T. OCHOA, THE LAW OF COPYRIGHT § 1:1 (2018).
[5] ABRAMS, *supra* note 4.
[6] U.S. CONST. art. I, § 8, cl. 8.
[7] This is not to say that there are not intellectual property concerns in Open Source. There certainly are intellectual property concerns, but overall, copyright, copyleft and licensing has become the largest battleground. While we may have intellectual property, and very valuable property at that, it is not common to use it against each other.Red Hat's Patent Promise, first outlined in 2002, has had a large part to play in the lack of patent litigation in open source. *Patent promise*, RED HAT, (Sep. 21, 2017), https://www.redhat.com/en/about/patent-promise.

> As we explained at that time, our patent portfolio is intended to discourage patent aggression in free and open source software (FOSS). Since then, we have worked hard to discourage patent attacks through a range of initiatives, and have never used our patents offensively. We believe our defensive approach to patents has been beneficial to the open source community as well as Red Hat.

Id. By and large, this has more or less been the case. Further, some standards of intellectual property are fundamentally not available in free and open source software. For example, trade secret relies on the information being kept a secret. While there may be *operational* trade secret, it is a logical fallacy to expect trade secrets around code. It's theoretically impossible to have a trade secret when your code is openly developed, freely available to download, and freely available to share. Within the areas of trademark, the association of a particular brand is not as meaningful for free and open source software. After a reputation has been established for a certain community, trademarks come into play, but they are not a foundational

Copyright law (and intellectual property law at large) works to solve the problem of information and sharing: creating high-quality information is usually associated with a significant cost, acquiring that information is less costly, and using it has almost no cost.[8] Software falls under copyright protection as a "non-dramatic literary work."[9] However, when copyright protection is used to the extreme, follow-on innovation and re-use are hampered. This issue led a small group in the software industry to consider alternatives: copyleft, which became some of the first open source licenses.[10] Under copyleft licenses, an author allows for a restriction that any derivative work also be shared, and that it be released free of charge to the public community of interested software developers. The essential idea of copyleft is to use copyright to facilitate the sharing and reuse of copyrighted content.[11]

The making of open source

Open source concepts reportedly first arose out of the UNIX platform created in collaboration with Bell Telephone Laboratories' Research Division, the Computer Systems Research Group of the University of California at Berkeley, and the UNIX Systems Group at AT&T.[12] Initially, developers came together to collaborate in

part of the landscape. Acquiring a trademark for the mere idea of a software project isn't meaningful, a more mature project would consider a trademark - and enforcing that trademark is usually beyond the initial goal of a community. So, as far as being able to create and allow for free and open source software, the only tool available is through copyright. Copyright both constrains and frees open source development and is the sole legal instrument of intellectual property to rely on. This has led to some unexpected results in practice.

[8] VAN LINDBERG, INTELLECTUAL PROPERTY AND OPEN SOURCE: A PRACTICAL GUIDE TO PROTECTING CODE 7 (2008).

[9] *Id.* at 4.

[10] *What is Copyleft?*, FREE SOFTWARE FOUNDATION, https://www.gnu.org/licenses/copyleft.html (last visited Nov. 26, 2019).

[11] *Id.* at 7.

[12] PETER H. SALUS, A QUARTER CENTURY OF UNIX 1-2 (1994).

connection with computer science programs, and collaboration was part of the science underpinning of their work.[13] However, as part of the breakup of AT&T under antitrust regulation, UNIX as a platform was taken private, and the benefits of collaboration were lost.[14] The loss of collaborative efforts was in part because of the exclusive rights of copyright law, and the idea was born to leverage copyright to facilitate the sharing of software code as a condition of using the work. This reaction to the loss of collaboration, among other factors,[15] caused the formation of the Free Software Foundation (the "FSF") in 1983 and the creation of the General Public License.[16] The FSF outlined: "Free software" means software that respects users' freedom and community.[17] Specifically, it means that "the users have the freedom to run, copy, distribute, study, change and improve the software."[18] Many of these rights already existed, but the requirement that all modified and extended versions became part of the copyleft licenses.[19] The Free Software Foundation defines copyleft as:

> Copyleft is a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well.
>
> The simplest way to make a program free software is to put it in the public domain, uncopyrighted. This allows people to share the program and their improvements, if they are so minded. But it also allows uncooperative people to

[13] There are several opening moments for the creation of open source, UNIX is one of many that arose at a time where computing power was expensive and separating the hardware from the code that ran it. *See* Computers: History and Development, JONES TELECOMMUNICATIONS & MULTIMEDIA ENCYCLOPEDIA, https://web.archive.org/web/20140213183848/http://www.dia.eui.upm.es/asignatu/sis_op1/comp_hd/comp_hd.htm (describing five generations of modern computers from 1945 to the end of the Twentieth Century).
[14] PETER H. SALUS at 63.
[15] Part of this loss of collaborative support also meant that it was unclear about the future of the previous work! When the project disbanded, who owned the work?
[16] *What is free software and why is it so important for society??*, FREE SOFTWARE FOUNDATION, https://www.fsf.org/about/what-is-free-software (last visited Oct. 8, 2019).
[17] *Id.*
[18] *Id.*
[19] *Id.*

> convert the program into proprietary software. They can make changes, many or few, and distribute the result as a proprietary product. People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away.[20]

Under these particular open source licenses, the Free Software Foundation explicitly defines user freedom as the highest goal, "we use copyright to guarantee [users'] freedom. That's why we reverse the name, changing 'copyright' into 'copyleft.'" [21] Under copyleft, the primary focus is on the restriction to continue to share work, that any works created with the original code must also be made open and available, with attribution for the original source. [22]

As the Free Software Foundation was formed, and the proliferation of copyleft licenses grew, a problem came up: How do we determine what is open? The Open Source Initiative (the "OSI") was formed to help guide and answer that question, such that now the OSI is seen as the home of all licenses.[23] The OSI defines open source directly:

> Open source doesn't just mean access to the source code. The distribution terms of open source software must comply with the following criteria:
>
> 1. Free Redistribution
>
> The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
>
> 2. Source Code

[20] *What is Copyleft?*, FREE SOFTWARE FOUNDATION, https://www.gnu.org/licenses/copyleft.html (last visited Nov. 26, 2019).
[21] *Id.*
[22] *Id.*
[23] OPEN SOURCE INITIATIVE, https://opensource.org/ (last visited Nov. 9, 2019). Note: "The Open Source Definition was originally derived from the Debian Free Software Guidelines (DFSG)." https://www.debian.org/social_contract#guidelines

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.[24]

Under this definition of open source (which remains in place today,) the key portions that led to the growth of modern open source are centered around the free distribution, the availability of code, and the ability for any developer to use and review the software and then contribute back. Because of this, codifying what open source meant also allowed for a wide variety of different licenses to arise. Generally, there are a limited number of licenses that have been adopted and govern the understanding of how an open source community interacts.[25]

---

[24] *The Open Source Definition*, OPEN SOURCE INITIATIVE, https://opensource.org/osd (last visited Nov. 25, 2019). The OSI in recent years has come under fire for not allowing for innovation. While this may be accurate, this fails to take into account the mission of the Open Source Initiative. Their focus, initially, was to be able to create and define what open source is, and what qualifies as open source. Since 1998, the reliance on software has changed greatly. The argument that the OSI should continue to be the only source of truth has the ring of a strict constructionist argument versus an interpretation argument. Specifically, the requirement that a license must not discriminate against a field of endeavor is an argument against the OSI being available to accept a license that has an ethical component to its overall charter.

[25] The General Public License is not the only license, OSI notes that:

> The following OSI-approved licenses are popular, widely used, or have strong communities:
>
> Apache License 2.0
> BSD 3-Clause "New" or "Revised" license
> BSD 2-Clause "Simplified" or "FreeBSD" license
> GNU General Public License (GPL)
> GNU Library or "Lesser" General Public License (LGPL)
> MIT license
> Mozilla Public License 2.0
> Common Development and Distribution License
> Eclipse Public License version 2.0

See Licenses & Standards, OPEN SOURCE INITIATIVE, https://opensource.org/licenses (last visited Nov. 26, 2019).

The community of developers themselves provide no warranty for whatever they produce, but as part of this, they declare who the authors of the software are.[26] This becomes part of the code that the software produces.[27] As part of this overall code creation, the information of who the authors are and how to contact them becomes part of the record of who contributed to a particular group, at what time, and with some nature of the contribution. This knowledge of contribution is part of the record for the creation and provenance, and as such, a regulation around the treatment of personal information conflicts.

The adoption of Linux (licensed under the General Public License) created an ecosystem that expects and understands that code will be freely available under an open source license, and that the maintainers of the code and original creators will sign their name and contact information to it.[28] This has several implications, particularly in a privacy-focused world.

The open source model has grown dramatically over the last two decades, and there are economic reasons for this growth. The impact of not working in open source is that not every problem has to start from the very beginning and can take advantage of prior work and prior innovations in the space. With open source, the advantages from having contributors from a wide variety of vendors and companies is much more compelling. Several other projects may already exist in the same problem space that solve several lower-level problems, so that the project can focus on a niche area to solve. As open source licensed code requires that any future works be made available and public, the

[26] *Id.*
[27] *Id.*
[28] Kernel.org historic source code, https://mirrors.edge.kernel.org/pub/linux/kernel/Historic/

current trend is that the overall success of a project built in the open is more likely.[29] Further, the ability to build consulting businesses around the software has also increased its adoption: the software is free, the expertise is a business. The developers contract for support and maintenance contracts instead of having the software be a commodity.

Part II: Creating version control and how that embedded personal information

In 2005, Linus Torvalds created git to better manage contributions in open source.[30] Git is a version control system that allows for many contributors to work on the same code at the same time without causing issues for each other. [31]Previously, other version control systems had been introduced: Concurrent Versions System (cvs) and Subversion (svn) both allowed for collaboration but were limited in scope. [32]Previous version control systems were designed for small groups working on the same pieces of code but not all at the same time, with little tooling in place to resolve conflicts between different developer's work. As development teams became more distributed, software became more complex. As the complexity of software grew, the complexity of the systems designed to support it also grew.

Business Implications of a Community Built on Open Data

[29] Why the license matters: Without the creation of open source, none of this would be a consideration. However, as open source software becomes more and more popular, the nuances of the licenses matter less. Ultimately, copyright still exists, but now, the common thinking in the space is that the copyleft and permissive licenses led us to creating spaces where collaboration was enforced and is not optional.
[30] Initial Commit of Git by Linus Torvalds, GITHUB.COM, https://github.com/git/git/commit/e83c5163316f89bfbde7d9ab23ca2e25604af290.Note: "git" is not a proper noun and should not be capitalized.
[31] *Id.*
[32] CVS—Concurrent Versions System v1.11.23, https://www.gnu.org/software/trans-coord/manual/cvs/cvs.html and Apache Subversion, https://subversion.apache.org/

As open source software has turned into a business, an initial point of considerable confusion was that the use of open source itself is a business model. A common misconception is that by releasing code into the open, you will gain advantages in the market. While this is sometimes true, this is not a complete business model. Open source is a development model on which businesses can be based. Open source itself can provide a much broader set of developers and their expertise, but the openness of the code is only a small part of the successful open source development model. The successful adoption of open source depends on the collaboration that occurs in the space, and to collaborate effectively, data sharing needs to occur. A method of version control focused on open collaboration came into play to satisfy the requirements of the General Public License (GPL) and allow for much faster collaboration: git.[33] Using version control, in particular, was a tremendous help, and git is currently a popularly adopted version control system within open source software communities.

Using git allowed for multiple developers to work on the same piece of code in tandem. It also provided a way to find the authoritative source of the last known working code.[34] While git improved the speed of development greatly, it also required a need to create a permission-based system based on individual access, and at the same time, every contributor signs off on their contributions with their name and their email address, proving that they wrote the code that is contributed, that they had the rights to do at the time, and that this code was accepted by the project.[35] This code and contributor record becomes part of the open record of the project in perpetuity. [36]

[33] Initial Commit of Git by Linus Torvalds, GITHUB.COM
[34] Git- log, https://git-scm.com/docs/git-log
[35] Git-commit, recording changes to the repository: https://git-scm.com/docs/git-commit
[36] *Id.*

For example, the first commit of git shows the name and date of who contributed the software, and when. The first commit of git reads:

> **Linus Torvalds** committed on Apr 7, 2005
>
> 0 parents commit e83c5163316f89bfbde7d9ab23ca2e25604af290[37]

Linus, in his initial commit, outlined the understanding of trust in this new system:

> TRUST: The notion of "trust" is really outside the scope of "git", but it's worth noting a few things. First off, since everything is hashed with SHA1, you _can_ trust that an object is intact and has not been messed with by external sources. So, the name of an object uniquely identifies a known state - just not a state that you may want to trust.
>
> Furthermore, since the SHA1 signature of a changeset refers to the SHA1 signatures of the tree it is associated with and the signatures of the parent, a single named changeset specifies uniquely a whole set of history, with full contents. You can't later fake any step of the way once you have the name of a changeset.
>
> So, to introduce some real trust in the system, the only thing you need to do is to digitally sign just _one_ special note, which includes the name of a top-level changeset. Your digital signature shows others that you trust that changeset, and the immutability of the history of changesets tells others that they can trust the whole history.
>
> In other words, you can easily validate a whole archive by just sending out a single email that tells the people the name (SHA1 hash) of the top

[37] *Id*
GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager. It doesn't do a whole lot, but what it does do is track directory contents efficiently.

changeset, and digitally sign that email using something like GPG/PGP.[38]

There is no other way to add one's work. Anonymous commits would never be allowed under the system of trust described from the initial commit because open source development relies on a strongly typed relationship management system. A circle of trust is established by an individual's actions in the community, and a single individual's code is part of their reputation in the community.

Version control and the rise of cloud computing in development

Another challenge in creating open source software was creating a place for collaboration. The result was a single host with many developers participating, and nearly all of the hosting platforms for development focused on the version control system of git. A hosted solution has become a popular way to contribute and consume software, and it allows for collaboration all over the world with few barriers to entry. These hosted platforms became a way for end-users (people who weren't initially developing the software) to be able to consume projects without directly contributing, and marketing for open source projects became a great deal easier with hosted platforms. Because of the gains available, a larger majority of development now happens in these hosted platforms, but is based on the system that was initially designed by Linus Torvalds.

In the last fifteen years, open source software grew to an ecosystem with many different component parts, including end-users who may not directly contribute to the software, but are still involved through feedback and reporting errors. The method of being able to take reports requires collecting some personal information like where the

[38] *Id.*

software is installed, the user that is running the software, and the IP address, which can lead to a physical location.[39] The continued growth of software communities depends on being able to take contributions in many different forms, and the vast majority of contributions involve signing with a digital signature and contact information. It is this signing with personal information that becomes an issue against a regulation governing private information.

Part III: GDPR has fundamental and unexpected consequences on open source.

The General Data Protection Regulation impacts open source software development communities from the perspective of using individual's personal data to manage development in an open and transparent way. The European Parliament enacted the General Data Protection Regulation in April 2016, which became effective on May 25, 2018.[40] The General Data Protection Regulation addresses concerns around misuse of personal data, corporate security breaches, and the inability of the law to respond to these issues.[41] Under the Data Protection Directive, several different variations of regulations complying with the directive were passed, leading to discrepancies in enforcement and wider regulatory function. The General Data Protection Regulation replaced the Data Protection Directive with a single regulation for all EU member states providing for

[39] An example of information needed: https://bugzilla.mozilla.org/show_bug.cgi?id=964021.
This notes the project, which release it was, and what steps need to be taken to reproduce a bug. Here, this doesn't require the IP Address, but a networking project might need this.
[40] Regulation 2016/679, of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data and Repealing Directive 95/46/EC, 2016 O.J. (L 119) 1 [hereinafter General Data Protection Regulation].
[41] Directive 95/46/EC, of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data, 1995 O.J. (L 281) 31.

meaningful standardization over the previous system.[42] Lastly, the GDPR allows for further clarification of the rights of the individual and what data is considered to be especially sensitive, and open source projects are particularly impacted by this regulation due to their reliance on open authorship.

Defining personal data and sensitive data

Under the GDPR, "personal data" is information related to someone who is or could be identified from that data.[43] While the term "personal data" is particularly broad, it includes information that identifies someone, such as by ID number, date of birth, name, or email address. [44] This personal data also includes information that can identify someone indirectly.[45] This personal can include information that can be used in context with direct identification, like company and job title, or information that could be used to link to the data subject, such as IP addresses.[46] Additionally, the GDPR also defines "sensitive data" and allows for greater restrictions on the collection or processing of this kind of data. [47]Sensitive data is defined as: "racial or ethnic origin; political opinions, religious or philosophical beliefs, or trade union membership; genetic data; biometric data for the purpose of uniquely identifying a natural person; data concerning health; data concerning a natural person's sex life or sexual orientation." [48]

Here, open source projects are not likely to have data that would be categorized as sensitive data. Instead, open source projects will have personally identifying information

[42] General Data Protection Regulation, *supra* note 2, at art. 2; *see also id.* at 2 (Recital 8).
[43] General Data Protection Regulation, *supra* note 2, at art. 1.
[44] *Id.*
[45]*Id.* at art. 4 (General Data Protection Regulation Definitions).
[46] *Id.*
[47] *Id.*
[48] *Id.*

from the data subject. While the data subject freely provides such information, this data consists of names, email addresses, and frequently companies, as companies support the development of open source software. [49] The GDPR defines the act of processing data as any time an operation is performed on that piece of data. [50] This can include collecting, storing, viewing, transmitting, and deleting the data.[51] Additionally, the regulation does not distinguish between human and automated processing.[52] Because all of this embedded personal information is freely available, it is challenging to limit "processing" to be only a certain set of actions. As a project publishes its source code freely and without limitations, anyone viewing the code and accompanying source code record could be considered "processing," and this has caused confusion in projects around compliance.

There are at least three areas of the GPDR that may implicate information that is directly embedded in open source and open source software: (1) the provisions around the protections of personal information; (2) the provisions around processing of data; and (3) the rights of the data subject, in particular, the right to be forgotten. In this first requirement, open source communities usually have no way to be able to designate a controller or processor. Under the GDPR, a "controller" is the company who determines the purpose and means of processing.[53] While a "processor" is a third party that processes it on a controller's behalf. This distinction is particularly relevant when personal data will

[49] Corporate Open Source Programs are on the Rise as Shared Software Development Becomes Mainstream for Businesses- August 30, 2018 - https://www.linuxfoundation.org/uncategorized/2018/08/corporate-open source-programs-are-on-the-rise-as-shared-software-development-becomes-mainstream-for-businesses/
[50] *Id.* at art. 6 (The General Data Protection Regulation's lawfulness of processing).
[51] *Id.*
[52] *Id.*
[53] *Id*. at art. 4.

be transferred to or from a third party. Unfortunately, most open source projects have neither.

GDPR's Principles are Aligned with Open source but Conflict in Practice

First, GDPR provides for lawfulness, fairness, and transparency.[54] This directive or principle also provides for the consent of the data subject. [55] Second, GDPR provides that the collected personal data is limited to legitimate purposes. Third, that the data that is collected is "limited to what is necessary in relation to the purposes" also known as ("data minimization").[56] Fourth, the personal data that is stored should be accurate and kept up to date. [57] Every reasonable step to erase or correct inaccurate data should be taken. [58] Fifth, that the data stored is not kept longer than necessary, and that the manner in which it is kept "in a form which permits identification of data subjects for no longer than is necessary for the purposes . . . in order to safeguard the rights and freedoms of the data subject". [59] Sixth, that the data is stored in a way that is secure, including "protection against unauthorised [sic] or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational [sic] measures ('integrity and confidentiality')".[60] Seventh, and lastly, the holder or controller of personal data is

[54] *Id.* at art. 6 (The General Data Protection Regulation's lawfulness of processing).
[55] *Id.* at art. 6(1)(a); *see also id.* at 7-8 (Recital 40, explaining the lawfulness of data processing).
[56] *Id.*
[57] General Data Protection Regulation, *supra* note 2, at art. 5(1)(d) (The General Data Protection Regulation principles relating to processing of personal data); *see also id.* at 7 (Recital 39, The General Data Protection Regulation principles of data processing).
[58] *Id.*
[59] General Data Protection Regulation, *supra* note 2, at art. 5(1)(e) (The General Data Protection Regulation principles relating to processing of personal data); *see also id.* at 7 (Recital 39, principles of data processing).
[60] General Data Protection Regulation, *supra* note 2, at art. 5(1)(e)

responsible for all six of the above principles, as well as being able to demonstrate compliance in all six of the above principles.[61]

Based on the principals and guidance that open source grew out of, these seven principals are in alignment with the vast majority of the open source software community. First, open source communities operate under freely given personal data and usually no sensitive data. Secondly, the copyleft communities are particularly focused on lawfulness, fairness, and transparency, but this may be an incomplete alignment. [62]Third, any personal data that is collected is limited to the legitimate purposes for which it was collected. Fourth, data minimization is a shared goal between the GDPR and most projects. A project exists to solve a problem for code, not to collect personal data. Fifth, any personal data provided is kept up to date and public, as everything is publicly available in a version-controlled system. Points five and six should be generally met as software communities focus on data security and storage more directly than other business-related communities. Lastly, if communities were to act collectively as a controller of personal data, the communities would be in compliance. But, as a controller, recital 156 allows for the legitimate business purpose exception that an open source community would fall under.[63] However, the application of these principals is unclear, given as the requirement for processing is met by having the personal data be available openly.

Lawfulness Defined

[61] *Id.* at art. 5(2). (The General Data Protection Regulation principles relating to processing of personal data).
[62] *What is Copyleft?*, FREE SOFTWARE FOUNDATION, https://www.gnu.org/licenses/copyleft.html (last visited Nov. 26, 2019).
[63] *Id.* at recital 156.

A conflict for lawfulness and open source arises under the definition for "lawfulness." Article 6 of the General Data Protection Regulation defines lawfulness directly.[64] Several different purposes are described, and in order to process personal data, one or more of these purposes must be demonstrably present.[65] For example, compliance with current law is an allowable use of processing personal data.[66] Additionally, if a data subject has given consent for one or more allowable purposes, that meets the requirements of Article 6(1)(a).[67] Under Article 6(1)(d), a controller can show that "processing is necessary in order to protect the vital interests of the data subject or of another natural person." [68] As mere "viewing" qualifies for "processing," a challenge arises here.[69] If a project allows for the public record to be available, this could be challenged under the lawfulness clause, and no distinction is currently provided for in the principals that directly applies to the open source use.

Personal Data is Part of the Record

Because of the reliance on version control, personal data becomes a direct part of the historical business record in an open source project. Within a project, all data is publicly available about who committed code, documentation, photos, and whether they had the appropriate rights to do so. Under the test of "lawful", this qualifies as the intent behind being able to give the personal data to a project was to support the development of open source software.[70] Further, a data subject submitting a contribution is aware that any

[64] *Id.* at art. 6.
[65] *Id.* at art. 6(1) (lawfulness of processing).
[66] *Id.* at art. 6(1)(c) (lawfulness of processing).
[67] *Id.* at art. 6(1)(a) (lawfulness of processing).
[68] *Id.* at art. 6(1)(d) (lawfulness of processing).
[69] *Id.* at art. 6(1)(d) (lawfulness of processing).
[70] *Id.* at art. 6(3).

data surrounding that contribution is made public. Under both of these conditions, this would be a legitimate use of personal data.

Open source has Enforcement Concerns Under the GDPR

Article 8 of the General Data Protection Regulation provides for guidance on how to report a violation, focusing on member states in Article 77.[71] Further, the agency that has the complaint updates the reporter on progress and outcome, including any judicial remedy under Article 79.[72] In short, the member states have various agencies that are to enforce the law, based on a certain number of complaints, and have broad discretion to take into account circumstances that led to the complaints.[73] These data regulators have the power to be able to levy heavy fines for non-compliance, with up to 20 million euros or 4% of annual global revenue, whichever is greater.[74] This significance in penalty has led to a great deal of both compliance as well as fear of non-compliance. This same significance in penalty also leads towards an attitude of strict compliance, instead of balancing the goals of the regulation, the needs of an organization, and the feasibility of a solution.

Applying the "Legitimate Business Purpose" clause to open source software

Unless overridden by the data subject's interests to the contrary", a legitimate business purpose clause is also provided for in the GDPR, so that "[p]ersonal data can be processed if doing so is consistent with "legitimate interests." [75] Here, an open source

[71] *Id.* at art. 8., *Id.* at art. 77.
[72] *Id.* at art. 79.
[73] *Id.* at art. 83.
[74] *Id.* at art. 84.
[75] *Id.* at art. 6(1)(f) (lawfulness of processing).

project can make a claim to a legitimate business purpose as the public record is part of the project, and is its entire documentation for existing. However, particularly in around the legitimate business purpose usage, these are subject to balancing the interests of a data subject. A legitimate interest could be subject to a review of "where there is a relevant and appropriate relationship between the data subject and the controller in situations such as where the data subject is a client or in the service of the controller." [76] However, no controller function exists in most open source projects, complicating this rule.

Additionally, the GDPR also allows for the performance or creation of a contract, but this is limited to a contract to which the data subject is a party.[77] No contracts currently exist within the open source space to allow for this obligation, but contracts could be created in the future. One option is that the "developer certificate of origin" may serve as this contract in the future. As version control itself provides challenges, a further question exists in contributing software about whether or not a developer certificate of origin ("DCO") would serve to help protect open source software communities.[78] A developer certificate of origin includes under section (d) an affirmation that the work is public.[79] Section (d) reads: "I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I

[76] *Id.* at 9 (Recital 47, overriding legitimate interest).
[77] *Id.* at art 6.
[78] lofidevops, *What Additional Benefits Does the DCO Provide?*, Discussion post to *Open Source Beta*, STACK EXCHANGE (Apr. 10, 2018, 1:03 PM), https://opensource.stackexchange.com/questions/6533/what-additional-benefits-does-the-dco-provide (last visited Oct. 20, 2019). A quote from a project contributor that signals confusion: "A project may not want to implement a DCO because this extra bureaucracy turns off potential contributors, and signing a DCO could make pseudonymous contributions impossible. Maintaining records of the signed DCOs could make the project subject to privacy regulations such as the EU-GDPR." *Id*, at comment by amon (Feb. 15, 2018, 1:28 PM).
[79] *Developer Certification of Origin*, LINUX FOUNDATION, https://developercertificate.org/ (last visited Nov. 26, 2019).

submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved."[80]

The developer certificate of origin indicates that there is no expectation of privacy or confidentiality of authorship. [81] By signing the contribution with a developer certificate of origin, the data subject is fully aware of the public nature of the work. This may serve under the "requirement of a contract," but further guidance is needed to outline if this understanding could form a binding contract. Further, the developer certificate of origin leads to the concept of consent in open source to contribute work.

The Definition of Context and Consent

The GDPR defines both context around processing of data and consent.[82] Context is outlined as a "careful assessment including whether a data subject can reasonably expect at the time and in the context of the collection of the personal data that processing for that purpose may take place."[83] Initially, personal data can be processed if the data subject gives their consent. However, the validity of that consent is subject to the requirement that it be "specific" and "informed."[84] Recital 32 states: "Consent should be given by a clear affirmative act establishing a freely given, specific, informed and unambiguous indication of the data subject's agreement to the processing of personal data relating to him or her, such as by a written statement, including by electronic means, or an oral statement."[85]

[80] *Id.*
[81] *Id.*
[82] *Id.* at art. 6.
[83] *Id.*
[84] General Data Protection Regulation, *supra* note 2, at art. 6(1)(a); *see also id.* at 6 (Recital 32).
[85] General Data Protection Regulation, *supra* note 2, at 6 (Recital 32).

For freely given consent to exist, a real choice must exist, and it has to be given voluntarily. Specifically, inaction or pre-filled forms do not qualify as consent: "Silence, pre-ticked boxes or inactivity should not therefore constitute consent."[86] Further, the data subject must be able to withdraw consent at any time.[87] Within an open source project, consent is not usually an issue, as shown by the Developer Certificate of Origin. [88] Contributions are acknowledged with consent.[89] The instances where consent does arise are where a contributor wishes to remove their contribution after it has become part of the public record.[90] This is generally not allowed under most open source communities' governance models and leads to friction for complying with the GDPR, as will be described under the rights and requests section which appears below.

Requests and Rights

Under the GDPR, specific rights are given to individuals regarding their personal data, and specific requests and actions can be made in connection with those rights.[91] Initially, the right of access is described under Article 15.[92] Specifically, a data subject is allowed to request the purpose of the processing, whether their data is being processed, how long the data has been or will be stored, and who the recipients are.[93] Further, the

[86] *Id.*
[87] *Id.*
[88] *Developer Certification of Origin*, LINUX FOUNDATION, https://developercertificate.org/ (last visited Nov. 26, 2019).
[89] *Id.*
[90] A user asks: "*How to Delete Issue?*" *See* keithmorr, *How to Delete Issue?*. General discussion post, DRUPAL (May 3, 2010, 1:34 PM), https://www.drupal.org/forum/general/general-discussion/2010-05-03/how-to-delete-issue (last visited May 2, 2020). "In general we don't delete content . . . ." *Id.* at comment by WorldFallz. This is a common practice among open source communities.
[91] *Id.*
[92] *Id.* at art. 15(c).
[93] General Data Protection Regulation, *supra* note 2, at art. 15 (The General Data Protection Regulation right of access by the data subject).

GDPR allows directly for the data subject to be aware of "particular recipients in third countries or international organisations [*sic*]."[94] Article 15 also allows for knowledge of "the existence of automated decision-making" including "meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject."[95] Further, the right to rectification is outlined in Article 16.[96] Data subjects can have inaccurate data both updated and corrected. In particular, where personal data "is no longer necessary in relation to the purposes for which they are collected or otherwise processed," the right to rectification or erasure is defined.[97]

The GDPR also defines the right to erasure (also known as the "right to be forgotten". [98]In limited circumstances, personal data can be erased. [99]Specifically, this is "relevant in particular where the data subject has given his or her consent as a child and is not fully aware of the risks involved by the processing, and later wants to remove such personal data, especially on the internet."[100] Article 18 defines the right to restriction of Processing.[101] Also, in limited circumstances, data subjects can restrict processing of their personal data.[102] This personal data can still be stored unless a Request for Erasure under Article 17 was also made.[103]

Open source communities are particularly impacted by the right to be forgotten. The right to access and rectification is easily solved as all work is done in the open and is

[94] *Id.* at art. 15(c).
[95] *Id.* at art. 15(h).
[96] *Id.* at art. 16(1).
[97] *Id.* at 12 (Recital 65; the General Data Protection Regulation right of rectification and erasure).
[98] *Id.* at art. 17.
[99] *Id.*
[100] *Id.*
[101] *Id.* at art. 18.
[102] *Id.*
[103] *Id.* at art. 17

publicly available. However, the right to be forgotten after a contribution is made is particularly challenging. As the software is created in public, the author's name is attached to the work as described previously. Removing an author would require the removal of that code as well. Deleting code and other contributions has a significant impact for a project to maintain continuity. As these contributions become part of the historical record for a project, the ability to remove a single commit or a piece of code without impacting the rest of the work is an unsolvable problem. Many other pieces of work may depend on that work that an author is attempting to remove, and the right to be forgotten is not applicable in open source due to the collective authorship nature of the work.

Part IV: A Further Example of Confusion and Compliance in Open Source

From a development perspective, open source communities inherently have personal data; this is not a choice. However, some communities can collect additional personal data about users in a way to inform development priorities. This can be collecting the IP addresses of website visitors in order discover geographical density of users or asking users to provide more data as part of a community survey to better understand how project features were being used. Evaluating that data against the possible consequences of the GDPR is a valid exercise as a community focused on collaboration. The questions that arose out of applying GDPR were mostly around communication. For example, a project can maintain a project newsletter, but should keep the email list of who is subscribed private. That communication is designed to support open source development and serves a legitimate business purpose. Further, no

need exists to process the data further, and as long as a mechanism exists around being able to request and remove data, this may be an acceptable use of personal data.

More direct communication was also unclear: many projects communicate through synchronous text chat called Internet Relay Chat or IRC. As the GDPR came into effect, services that copied the text chat into a log that was publicly available shut down due to fear of being liable.[104] This would be protected under Article 89 as an archive for historical research purposes.[105] However, this interpretation of the GDPR was not a universally held interpretation, and many individual developers chose to remove tools or websites that were designed to help foster open source communities due to the unclear nature of the law. As noted in Part III, a developer certification of origin may help to resolve this in the future. Overall, several factors combined created confusion in open source communities. The lack of clarity around communication, the lack of understanding of liability, and the fundamental mistake around the mechanics of contribution create barriers toward understanding how open source and the General Data Protection Regulation align.[106]

An example of a community privacy policy that was rewritten to conform to the GDPR is Fedora.[107] Fedora is a free open Linux desktop distribution that is supported by

[104] Lincoln Loop, *Saying Goodbye to BotBot.me*, LINCOLN LOOP, https://lincolnloop.com/blog/saying-goodbye-botbotme/ (last visited Nov. 26, 2019).
[105] General Data Protection Regulation, *supra* note 2, at art. 89 (The General Data Protection Regulation safeguards and derogations relating to processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes).
[106] Further, a few bad actors have used the regulation to encourage this fear. *See* ShipYourEnemiesGDPR.com, https://web.archive.org/web/20190529015002/https://shipyourenemiesgdpr.com/ (last visited Dec. 1, 2019). As with any highly fined, highly publicized legislation, someone is going to attempt to do harm with it.
[107] Fedora Community, *supra* note 3.

Red Hat and has a long historical record of work.[108] As part of evaluating the privacy policy that explains how Fedora works, several important pieces came into play. The applicability of the regulation and the challenges associated with acquiring additional data to verify applicability, then directly defining personal data, further outlining the possible uses of that data, as well as what sensitive data is used by the project and for what purpose, and why.

As previously noted, the GDPR has a particular focus for EU and EU nationals – regardless of location.[109] The GDPR applies to everyone with an EU affiliation, this provided challenges for the Fedora community.[110] Limiting the ability to access and remove personal data to just one group became an impossible burden, and instead, the choice was made to allow this for all Fedora community members. In order to confirm if someone was an EU citizen, the project would have to acquire additional identifying information. Acquiring additional identifying information for verifying identity did not make sense for a software project to have. Under Article 11, this is explicitly noted:

> If the purposes for which a controller processes personal data do not or do no longer require the identification of a data subject by the controller, the controller shall not be obliged to maintain, acquire or process additional information in order to identify the data subject for the sole purpose of complying with this Regulation.[111]

[108] Fedora Community, https://getfedora.org/

[109] General Data Protection Regulation at art. 18.
[110] General Data Protection Regulation at recital 14.
[111] General Data Protection Regulation, *supra* note 2, at art. 11.

Under the requirements of the GDPR, Fedora also outlined direct personal data.[112] Further, the Fedora policy also noted what data was publicly available.[113] Explicitly, any third-party sharing (which was limited), was also defined:

> Fedora may share your personal data with third parties under any of the following circumstances:
>
> Your publicly available personal data in the Fedora account system, as described above, is accessible by anyone unless you, as the account holder, opt out as already described in this Privacy Statement.
> As required to provide service, and for e-mail housing (as a consequence of uses already described in this Privacy Statement). It is in Fedora's legitimate business interest to provide all users an accurate record of data and content provided by Fedora's services, and to maintain the integrity of that data and content for historical, scientific, and research purposes. This data and content may include but is not limited to email, code changes, comments, and artifacts.
> As required by law (such as responding to a valid subpoena, warrant, audit, or agency action, or to prevent fraud).
> For research activities, including the production of statistical reports (such aggregated information is used to describe our services and is not used to contact the subjects of the report).[114]

Specific notes around how and where a user would be tracked to help improve the project (cookies) are also noted in the privacy policy along with what purpose the data served. [115]

Lastly, Fedora provided a way to request data through a form that required logging in with previously created credentials, assuring both (1) a pre-existing relationship with the project and (2) a verifiable way to provide personal data.[116] At issue is the possible

[112] *Fedora Community*, *supra* note 3 (section on "The Information We Collect").
[113] *Id.* (section on "Publicly Available Personal Data").

[114] *Id.*
[115] *Id.*
[116] *Fedora Community*, *supra* note 3 (section on "Your Rights and Choices in the EEA").

misuse of the system to provide personal data for an unauthorized subject, and this system allows for only qualified users to request data.[117] Here, the response to the regulation was valuable to the project both by streamlining data processes but then establishing a process that could be automated (such as bringing up records), and controlled. The benefits of the regulation were in the process and the creation of the privacy policy, not the burden in fulfilling the goal.

Part V: Further Questions: Towards a Future Regulation with Additional Guidance

The introduction of the GDPR allowed for the ability for open source communities to understand and then codify how personal data was used in service of open source development. The creation of open source software, specifically the focus on personal trust, created a public record that has personal data built in directly in a way that is immutable. That inherent tension between privacy as a right and open source development as a public work should direct future development, and any future open source project should take privacy into account when designing contribution systems. At this point in time, the introduction of stronger privacy regulations all over the world is inevitable, and the GDPR leads the way. Our collaborative spaces should acknowledge that privacy as a right directs communities to limit their use of personal data in development, acknowledging that the system is built on trust and transparency, and the current and forthcoming privacy regulations should acknowledge special uses of personal data that are not commonly considered. However, any further privacy regulation,

[117] *Id.*

particularly any focused on an online space, should take open source development and practices into account in order to fully serve the public good.

Cybaris®

Cybaris®, an Intellectual Property Law Review, publishes non-student articles and student comments on all areas of intellectual property law, including patents, copyrights, trademarks, licensing, and related transactional matters.
mitchellhamline.edu/cybaris

Intellectual Property Institute

Cybaris® is a publication of the Intellectual Property Institute at Mitchell Hamline School of Law.
mitchellhamline.edu/ip

MITCHELL | HAMLINE
School *of* Law

875 Summit Avenue, Saint Paul, MN 55105
mitchellhamline.edu