# An Examination of Open Source Software

Elijah C. Nicpon

College of Computer Science, Georgia Institute of Technology

CS 3001: Computing and Society

Dr. Clint Zeagler

15 November 2020

## Abstract

This paper seeks to evaluate how sustainable it may be for the software development industry to rely on open source software (hereinafter referred to as “OSS”). Considerations include the level of security that OSS projects can provide, the economics and ethics of for-profit companies relying upon software not driven by a revenue model, and potential areas of improvement for the industry.

## Introduction

Whether one is aware of it or not, open-source software projects power everyday life. Many key functionalities of modern computing devices and networks rely entirely upon them. Without OSS projects, if it were an Android or Linux[1] device, it wouldn't even boot. Trying to surf the web? Most browsers, including all Chromium[2]-based browsers as well as others like Firefox, are open source projects. Not to mention that most major website components, from entire CMSs (Content Management Systems) like WordPress, frontend tools like React, Angular, Vue, and Bootstrap, and databases like Neo4j, PostgreSQL, and Redis are entirely open source.

But what does "open source" really mean? Is it just a fancy way of saying free?

## Background: Defining Open Source Software (OSS)

Open source software (OSS) is any software released under a license that allows the general public to not only use the software for free but to also change and redistribute the source code for anyone with any purpose, enforced through an open source license such as MIT, Apache 2.0, AGPL, or GPLv3. OSS projects aren't typically driven by a for-profit revenue model, rather, they are driven by contributions from their communities. People choose to contribute for a variety of reasons, for practice, for fun, or for a sense of duty if they've utilized or profited from the software in the past. Employers may also direct their employees to work on adding features or fixing bugs in an open source project, sometimes dedicating entire departments to a single OSS project. A shining example of this is React, which was made open source by Facebook in 2013. But why would Facebook (now Meta) not only make the most popular JS library free but

[1] At the least in its open-source distributions such as Ubuntu, Debian, and Fedora.
[2]An open-source browser project which is the basis for Google Chrome, Microsoft Edge, Opera GX & Neon, Brave Browser, and many more.

also spend millions on development and maintenance each and every year? Answering this question requires a deeper understanding of how OSS projects are managed and developed.

## Background: OSS Project Management and Development

There are two main divisions of OSS projects. The first division is new projects that start initially as open source or go open source at an early stage of their development. The second is existing projects that were begun as closed source and may have multiple stable releases, but are released as open source later in the project lifetime.

Most projects have a project roadmap with goals that define aspects of success that the project wants to achieve. This may be in terms of features that are desired in the project, project popularity, or maintaining an active development community. As with any project, leadership is key. OSS project leaders are known as "maintainers" and are tasked with making development decisions and creating contribution roles and guidelines for committers and contributors. For new projects, maintainers may be the founding members of a project or a promoted committer, whereas pre-existing projects may be maintained by paid employees of the company that brought the project to open source. A contributor is anyone who contributes to a project in any way (including non-coding tasks like marketing), and a committer is someone who has contributed significantly to the project source code. Committers and contributors can either work towards goals on the project's roadmap or fix issues reported by users, often managed in an application such as Jira[3]. Larger projects have more responsibilities, which may lead maintainers to turn to an OSS incubator for help. For example, according to their website, The Apache Incubator has offered organizational, legal, and financial services to 315 OSS projects from 2002 to 2019.

[3] An agile software project management software by Atlassian that provides services including but not limited to issue tracking and workflow automation.

## Motivations to Open Source

Projects that elect to be open source from an early stage of development often do so because the project would be too complex or costly for a single person or small group to develop. As for larger projects that go open source in the latter stage of development, such as Facebook's open sourcing of React as mentioned above, Kochhar et al.'s 2021 paper, *Moving from Closed to Open Source: Observations from Six Transitioned Projects to GitHub*[4] enumerates a larger number of motivations to transition to open source. This includes building trust (both through transparency and price stability) with outside developers, expanding the pool of contributors and resources, expanding the pool of potential full-time employees for companies that maintain their open source projects, more prompt issue responses, and more business opportunities. Another example of this can be taken from just about any of Google's open source projects, including Android, Kubernetes, Angular, Chromium, Go (programming language), and TensorFlow. In the case of Android, Google made the decision to go open source to make it as widely available and popular as possible, as well as expand its development community. Since going open source, a huge base of applications has been developed by the open source community, and Android quickly became the obvious choice for most mobile phone manufacturers.

Surprisingly, Kochhar et al.'s paper does not mention another major motivator to bring a project open source, which is security.

## OSS Security

[4] A paper that observed Microsoft's transition of 6 previously closed-source projects to open-source, including interviews with Microsoft developers and independent contributors of the open source project.

Since all code of an OSS project is publicly available for download, logic may dictate that any threat actor[5] with a reasonable amount of expertise could download the code and find weaknesses that could be exploited to the detriment of any person or enterprise which uses the software. While it is certainly true that anyone can download the source code of an OSS project, vulnerabilities in large OSS projects are few and far between. This is because, for every threat actor, there are dozens of dedicated software engineers and white hats[6] that rigorously test and review each patch and each version of the software, weeding out bugs and vulnerabilities along the way. Some of these review processes are even automated with repository-integrated CI/CD tools (such as GitHub Actions or GitLab CI/CD), meaning that every pull request[7] is assessed before it ever reaches the public, and issues are raised to human reviewers when appropriate. As previously mentioned, many OSS projects use an application like Jira to receive and triage bug reports. Chalet and Du's 2007 paper, *Microscopic Model of Software Bug Dynamics: Closed Source Vs. Open Source* offers an in-depth analysis of the efficiency of the bug-reporting methods open source and closed source projects employ, revealing that open source projects consistently outperform closed source projects with no bug report resubmission allowed between releases (see Fig. 1).

[5] A cybersecurity term describing any person or group that aims to exploit vulnerabilities in an information system.
[6] A cybersecurity term describing an information security professional that uses the same skills and technologies as malicious hackers, but to discover and report vulnerabilities rather than exploit them.
[7] Proposed change to a software project, also referered to as "merge request" on different platforms, or informally as a "PR"

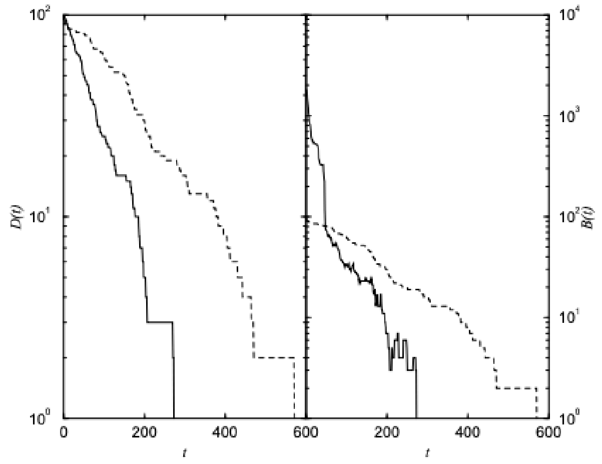

*Fig. 1.* Number of defective parts (left panel) and number of bugs (right panel) in an open source project (continuous lines), closed source projects with no bug report resubmission allowed between releases (dashed lines). (Chalet & Du, 2007).

Still, the concern mentioned at the beginning of this section is not unfounded, and the impact of the issue with OSS described above was clearly demonstrated in 2021, when the infamous Log4j vulnerability was discovered and exploited. However, even amidst failure, the benefits of OSS shine through. Since Log4j had so many stakeholders across multiple industries, software engineers from companies across the globe were deployed to assess the damage and formulate a response. While some companies only worked to produce a fix for their own versions of Log4j that they evolved for their own uses over time, the vast majority of companies with unaltered code opted to pool their resources with other stakeholders for the sake of a speedy fix through the Apache Software Foundation and its official repositories. The vulnerability was publicly disclosed worldwide on December 9th. By December 10th, Cloudflare[8] added rules that blocked HTTP requests containing characteristics of attack code, and VMWare[9] and Cisco[10] released patches. This global multifaceted response was far faster than responses to cybersecurity

[8] From: blog.cloudflare.com/cve-2021-44228-log4j-rce-0-day-mitigation/
[9] From: www.vmware.com/security/advisories/VMSA-2021-0028.html
[10]From: tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-apache-log4j-qRuKNEbd

breaches to private entities, because of the increased personnel and depth of knowledge of the global community.

While it may seem counterintuitive, Jaap-Henk Hoepman and Bart Jacob's 2007 paper, *Increased Security Through Open Source,* concludes that in the long run, "openness of source increases its security." Hoepman and Jacobs cite sloppy code being exposed, users and developers both running tests independently to validate the source code and the development of new security tools as the leading factors of increased security. Also noted is the fact that the transparency allows prospective users to make their own conclusions regarding the security of the project, rather than going with a closed-source option that wouldn't allow them to independently verify its security.

## When an Open Source Project Stops Being Open Source

Sometimes, sizable software companies are built by the leaders of an open source project, like HashiCorp and Docker. Revenue streams are hard to come by for such companies, and often take the form of advisory services for entities that utilize their respective open source software products. One such company worth exploring in depth is Elastic, which was founded in 2012 by Shay Banon, the creator of Elasticsearch, an open source full-text database released in 2010. From 2012 to 2020, Elastic sold advisory services for their ELK Stack[11]. However, in 2021, Elastic announced a licensing change to SSPL[12], effectively going closed source, despite

[11] A tech stack for centralized logging and searching, consisting of Elasticsearch (a NoSQL database), Logstash (for data ingestion), and Kibana (for visualization), all of which are open source projects released by Elastic from 2010-2012.
[12] Server Side Public License, which is not recognized by the Open Source Initiative (OSI) as open source, and would require SaaS providers to publish their codebase in its entirety should they sell Elastic products under their new license.

promising "Apache 2.0[13] forever and always" as recently as December 2020, much to the protests of contributors not affiliated with Elastic. The reasons for this change are discussed later, for now, the focus is the impact of this change on the software industry, exclusive of Elastic and the parties discussed later.

For entities that managed their own ELK stack rather than opting for an often cheaper cloud-based solution, very little changed. However, for many cloud-based use cases, this change spelled a cumbersome and expensive migration process to a different full-text search such as Apache Solr or Vespa. Smaller Software as a Service (SaaS) providers lacking resources to create their own alternatives would need to work with Elastic or its competitors in order to offer a hosted full-text search to its customers or switch to an Elasticsearch alternative. Managed Security Service Providers (MSSPs), which rely upon constant bug fixes and security patches that previously relied on Elasticsearch backings for their IP, would also be forced to make an expensive decision.

## Open Source Violations

In order to holistically examine how healthy it is for the software industry to rely on OSS projects, we must consider what is generally accepted as the largest threat to the open source community: violations of open source licensing, and their repercussions on the software industry.

Returning to Elastic's storyline, since the ELK Stack was open source, *anyone* could take its source code and alter it for their own use, including a large corporation like Amazon. On January 14th, 2021, Elastic announced the licensing change, followed by this blog post (most relevant portions below; see the full blog post at www.elastic.co/blog/why-license-change-aws):

[13] A popular open source license that, among other things, would allow SaaS providers such as AWS to fork and sell the software for profit. Apache 2.0 is recognized by the Open Source Initiative (OSI) as open source.

*So why the change? AWS and Amazon Elasticsearch Service. They have been doing things that we think are just NOT OK since 2015 and it has only gotten worse. If we don't stand up to them now, as a successful company and leader in the market, who will? Our license change is aimed at preventing companies from taking our Elasticsearch and Kibana products and providing them directly as a service without collaborating with us…*

*…We think that Amazon's behavior is inconsistent with the norms and values that are especially important in the open source ecosystem. Our hope is to take our presence in the market and use it to stand up to this now so others don't face these same issues in the future.*

*In the open source world, trademarks are considered a great and positive way to protect product reputation... So imagine our surprise when Amazon launched their service in 2015 based on Elasticsearch and called it Amazon Elasticsearch Service. We consider this to be a pretty obvious trademark violation. NOT OK…*

*…When the service launched, imagine our surprise when the Amazon CTO tweeted that the service was released in collaboration with us. It was not. And over the years, we have heard repeatedly that this confusion persists. NOT OK.*

*When Amazon announced their Open Distro for Elasticsearch fork, they used code that we believe was copied by a third party from our commercial code and provided it as part of the Open Distro project. We believe this further divided our community and drove additional confusion…*

> *…Recently, we found more examples of what we consider to be ethically challenged behavior. We have differentiated with proprietary features, and now we see these feature designs serving as "inspiration" for Amazon, telling us their behavior continues and is more brazen. NOT OK.*
>
> *We collaborate with cloud service providers, including Microsoft, Google, Alibaba, Tencent, Clever Cloud, and others. We have shown we can find a way to do it. We even work with other parts of Amazon. We are always open to doing that; it just needs to be OK…*

Shay Banon, Elastic founder and CEO, received lots of criticism for this move, having once been (and still a self-proclaimed) open source activist. Still, it is important to remember that the licensing decision was not just for his own self-preservation, but for that of an entire company of thousands of employees relying on their salaries. Regardless, whether one opts for the David and Goliath story, where Elastic takes on the Goliath AWS, or the story that paints Elastic as the reverse-Robinhood, where Elastic steals from everyone to give to themselves, the change financially costs the industry at the end of the day. Further, this was only one well-documented and well-publicized event between two large software companies. Had the Elasticsearch project not had access to Elastic's legal and financial resources, this issue wouldn't have reached the courts or the press. For example in 2021, TikTok's PC streaming app was accused of copying code from Open Broadcast Studio (OBS), which was released under GPLv2.[14] Since TikTok did not release the source code of the modified application, they were in

[14] General Public License (GPL) is often considered the most "open" of open source licensing. While anyone can modify it for their own use, the GPL demands that if one wishes to distribute modified software in an executable format, one must also release its "complete corresponding source code."

direct violation of the license. However, since OBS didn't have the resources to bring its case to court, TikTok suffered no consequences.

## Software as a Collaborative Effort

Given the assumption that it would be impossible for one person to independently gain an in-depth understanding of all topics in the field of computer science (or any field for that matter), in order to build applications at the frontier of current human capabilities, software engineers must work together. Without collaboration, every software engineer would have to reinvent the wheel from scratch, but in this case, the wheel is a software project that could potentially have millions of lines of code. Developing complex applications in a silo could take many lifetimes. But when engineers work at different companies with different products, such collaboration becomes more difficult, especially when it comes to building and defending their respective company's IP and revenue stream. How can engineers collaborate while simultaneously protecting their conflicting interests?

OSS projects present themselves as the perfect opportunity for collaboration between software engineers on neutral territory. Even though no one owns the product, everyone can reap the rewards and customize the product for their own use cases. These projects may have never been completed without the resources of the open source community, and projects with enough support may even be finished faster than if they had been developed closed source by a single entity. This is where the power of the open source community shines brightest. Together, the community is stronger than any single corporation. If there's a large enough demand for a product and enough engineers are willing to back it, OSS projects can accomplish it, and even bust decades-old monopolies in the process. An example of this starts with the public's displeasure towards Adobe's monopoly over creative software. OSS projects like Audacity, GIMP, and Inkscape were started by engineers who recognized the demand to offer competing direct alternatives to some of Adobe's product line, which in turn forces Adobe to innovate to remain competitive.

## Conclusion and Improvements

While OSS projects serve as a necessary medium of collaboration for those in the field of software development, this doesn't mean they will continue to be self-sustainable going forward. The success of OSS projects will only last as long as people and companies are willing to support them. While it may seem like there will always be people willing to support OSS projects right now, it is important that the industry continues to foster this fragile spirit. Situations like the 2021 disagreement between Elastic and AWS discourage committers, who felt like their contributions towards Elasticsearch in its open source stages were taken advantage of by both Elastic (by going closed source) and AWS (by violating Elasticsearch's license), ultimately wounding the entire open source community. Avoiding similar situations in the future is imperative to the long-term health of the open source community. This includes incentivizing communities as well as protecting them.

Violations on smaller projects often go undetected. Even when they are detected, the cost of pursuing the case in court often outweighs the potential benefit. Reporting systems need to be established and stricter penalties for violations need to be set and enforced, especially strong copyleft licenses[15] such as GPLv2, GPLv3, and AGPL.

[15] Licenses that require source code must be released alongside binary distributions of software for any derivative works.

As for incentivizing OSS projects, both monetary donations and donations in the form of paying employees to contribute should be more common practice among large for-profit companies. Microsoft's FOSS[16] program is a perfect example of a way a company can give back to the developers of open source projects that they utilize. If more programs like these are supported by other industry giants alongside increased protection, I have a positive outlook on the future of open source projects and the industry's ability to rely upon them for the coming years.

[16] Microsoft Free and Open Source Software (FOSS) Program: Allows employees to vote each month on an open source project to receive $10,000

## Resources

*10 Free Open Source Alternatives to Adobe Creative Suite*. (2011, June 21). TechRadar. https://www.techradar.com/news/software/applications/10-free-open-source-alternatives-to-adobe-creative-suite-1305714

Apache Software Foundation. (n.d.). *Log4j – Apache Log4j Security Vulnerabilities*. Logging.apache.org. https://logging.apache.org/log4j/2.x/security.html

CHALLET, D., & DU, Y. L. (2005). MICROSCOPIC MODEL OF SOFTWARE BUG DYNAMICS: CLOSED SOURCE VERSUS OPEN SOURCE. *International Journal of Reliability, Quality and Safety Engineering*, *12*(06), 521–534. https://doi.org/10.1142/s0218539305001999

Fuggetta, A. (2003). Open source software—an evaluation. *Journal of Systems and Software*, *66*(1), 77–90. https://doi.org/10.1016/s0164-1212(02)00065-1

Hoepman, J.-H., & Jacobs, B. (2007). *Increased security through open source*. Https://Dl.acm.org/; Association for Computer Machinery. https://dl.acm.org/doi/fullHtml/10.1145/1188913.1188921?casa_token=bFS2oE4zi2gAAAAA:MmPLZefnlplPeZuo_A23DkoG3XMLXk3786tcP4ijDBfAQ3j12KU5-QUMCyrNrT_m5h10I4YS7GqsAw

Institut für Rechtsfragen der Freien und Open Source Software. (n.d.). *GNU General Public License (GPL) | ifrOSS*. Ifross.org. Retrieved December 6, 2022, from https://ifross.org/?q=node/3

Kochhar, P. S., Kalliamvakou, E., Nagappan, N., Zimmermann, T., & Bird, C. (2019). Moving from Closed to Open Source: Observations from Six Transitioned Projects to GitHub.

*IEEE Transactions on Software Engineering*, *47*(9), 1–1. https://doi.org/10.1109/tse.2019.2937025

Louw, C., Paffenholz, R., Verset, C., & Krause, G. (2022). Global Good Open Source Software Development in Response to the COVID-19 Pandemic – Perspectives from SORMAS Implementation in Europe. *Studies in Health Technology and Informatics*, *294*. https://doi.org/10.3233/shti220553

Tiwari, N. (2014). *How open sourcing Android made it a mobile market leader | Opensource.com*. Opensource.com. https://opensource.com/business/14/7/how-open-sourcing-android-made-it-mobile-market-leader

Yin, L., Chen, Z., Xuan, Q., & Filkov, V. (2021). Sustainability forecasting for Apache incubator projects. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. https://doi.org/10.1145/3468264.3468563