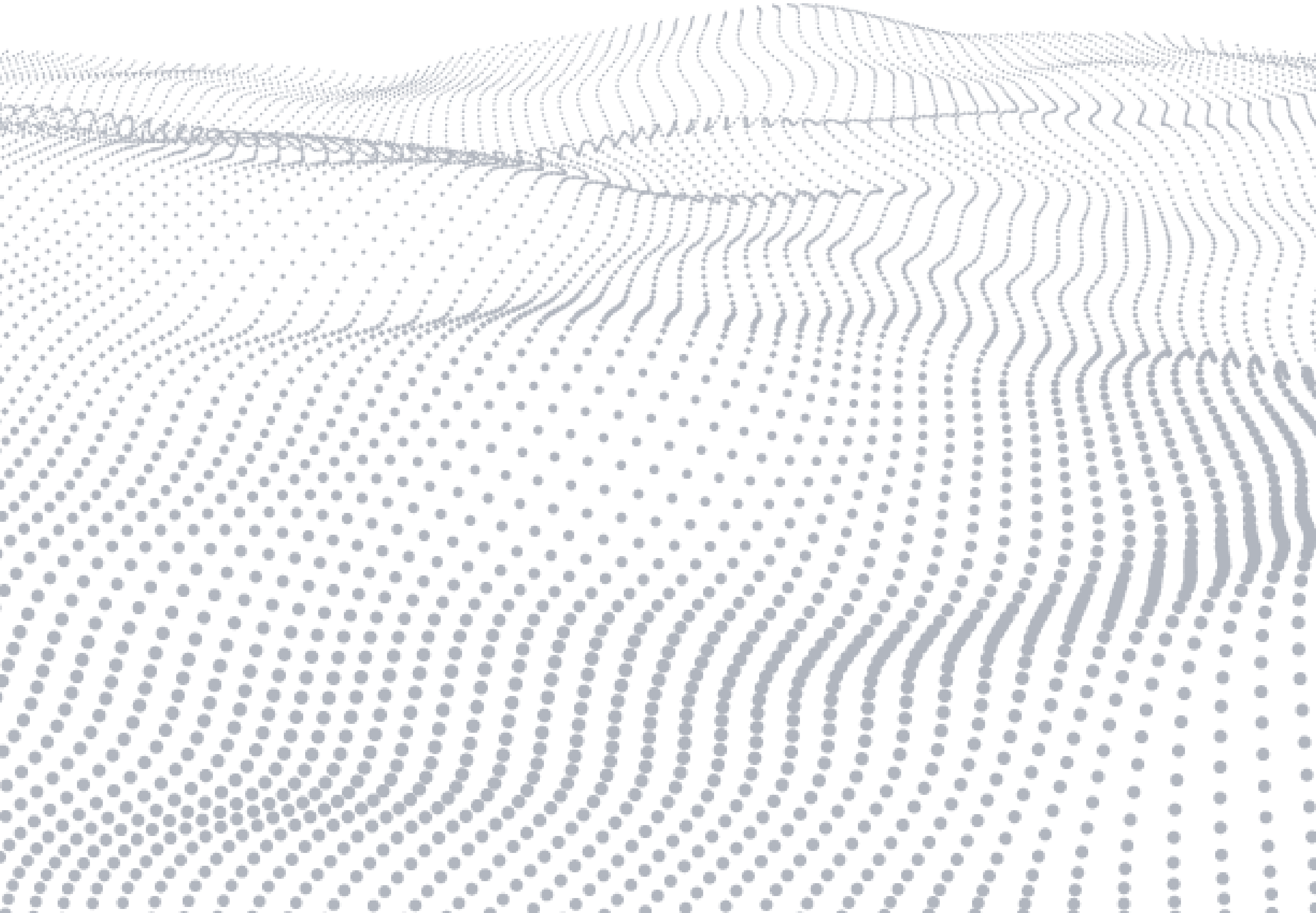
WHITE PAPER

Open Source Software in SaaS Offerings: Truths, Myths, and Considerations

By: Anthony Decicco & Stephen Pakan, GTC Law Group & Affiliates
Phil Odence, Black Duck



Introduction

Virtually every application today contains open source (more than 95% of the audited applications discussed in the Black Duck® "Open Source Security and Risk Analysis" report). And what's not to love? Open source software supports collaborative development and reduces the need to continuously re-invent the wheel. A developer need not write everything from scratch since many of the components have been written, reviewed, tested, and improved, and are widely and freely available to be incorporated into other projects and products. Open source lowers development and maintenance costs, and more importantly, allows companies to focus limited resources on the parts of their products that are new and innovative. In many industries, the use of open source also permits and promotes interoperability, flexibility, and customization.

As with any software written by others, there are compliance obligations for software you obtain under an open source license. If your company offers a product via a software-as-a-service (SaaS) model, you may have heard or read that some of the most common open source licenses, while being quite problematic for distributed software, may "give a free pass" to SaaS applications. However, while there is truth in the premise, you should resist the temptation to gloss over open source software obligations and restrictions and should be aware that there are plenty of other reasons that SaaS companies should pay attention to their use of open source. What follows is an overview of the potential pitfalls of using open source in SaaS applications and a discussion of the legal, security, operational, and strategic considerations; they are all manageable, but are not to be ignored.

Legal considerations

Some open source licenses directly address hosting

Not all open source licenses are silent with respect to hosting. For example, the GNU Affero General Public License (AGPL) v3 was written with SaaS solutions in mind as a way to extend the application of the GNU General Public License (GPL) to hosting. Per the preamble of the AGPL, it is "specifically designed to ensure cooperation with the community in the case of network server software." As such, use of AGPL-licensed code over a network may trigger an obligation to make the source code available. Even worse, modification of any AGPL-licensed code used as part of a SaaS offering or in combination with proprietary code may require the entirety of the offering to be subject to the AGPL, potentially destroying the commercial value of the product's code.

Other open source licenses are similar in their treatment of and impact on SaaS offerings, though with much more nuance. MongoDB's Server Side Public License also addresses hosting by including "making available to the public" as part of the definition of "propagate" and may also require the whole offering to be made available in source code form in certain situations. Similarly, Redis Labs created the Redis Source Available License to restrict use of its Redis modules in connection with products or services that are distributed or made available to third parties and that are databases, caching engines, stream processing engines, search engines, indexing engines, machine learning or deep learning or artificial intelligence serving engines, among others. While MongoDB and Redis Labs' original intent may not have been SaaS applications that merely use their software as a datastore, these license terms potentially ensnare SaaS applications and may have negative consequences, especially during due diligence review in connection with financing, M&A and IPOs.

In addition, some code is available under a choice of an open source license and a commercial license. Code licensed under such a dual licensing model incentivizes the licensor of such code to be diligent with respect to licensee compliance with the applicable open source license. iText (licensed pursuant to a choice of the AGPL v3 and a commercial license) is one such example. Per iText's AGPL web page, any deployment on a network requires "disclosing the full source code of your own applications under the AGPL license." As another example, SugarCRM is also AGPL-licensed with commercial options. To avoid potentially having to release source code of the proprietary portions of your company's code, it is imperative to know whether your company is hosting such software and whether your company's use is in compliance with such hosting-related open source licenses. Licensees have been taken to court for trying to "have their cake and eat it too," *i.e.,* using the free version in connection with a commercial offering and without meeting the AGPL obligations.

Most open source licenses lack express hosting rights

Most open source and freeware licenses do not expressly say whether one has the right to host the software; however, most common open source licenses do include a "use" right. Where a license permits distribution and does not otherwise prohibit hosting (other than possibly boilerplate restrictions against renting, leasing, and lending the software), some SaaS providers take the position that hosting rights can be implied for any code for which distribution is permitted. However, it is possible that these licenses do not grant adequate rights to host the licensed components or use the components in a hosted environment.

Conservative investors or acquirers may require that your company have express hosting rights for all such third-party software. In addition, some code available online lacks a license altogether, and in those cases, unless a suitable license can be implied, use of the code should be avoided as all rights not granted are reserved to the author by default.

Technically, there may still be a distribution

Even though hosting is generally treated as separate from distribution, many SaaS offerings still distribute some of their code. For example, many SaaS offerings include a distributed mobile client front end (often available from an app store).

More subtly, many SaaS offerings use a server-client model that results in client-side code that is downloaded to, and executed on, a separate device (such as in a browser). This client-side code is often overlooked by "pure SaaS" companies that think their entire solution is hosted, which can lead to failure to comply with the relevant restrictions and obligations tied to distribution. Some common JavaScript components that run on the client side of products are licensed pursuant to the GPL (Sencha Ext JS, which is also available under a commercial license, is an example) and so care must be exercised with client-side components. There are few "pure SaaS" companies, so development organizations need to stay on top of which components are acceptable to use on the client side and which must be restricted to the back end. Knowing which components may be distributed is paramount to open source license compliance.

Patent termination clauses and patent right grant-back clauses

Many open source components are licensed under terms that do not require disclosing proprietary code that merely links with the open source component, but many of these licenses contain patent-related clauses. For example, the Microsoft Public License, Common Development and Distribution License, and similar "public" licenses (such as the Eclipse Public License and the Mozilla Public License) contain defensive patent termination clauses that may impact your ability to enforce your patent rights against the licensor, contributors to the open source project, and in some cases, other users of the open source project. Even if your company is using open source software in a SaaS offering, these types of patent issues may still need to be taken into account.

Security considerations

Security concerns related to open source software use

Particularly if your product is a SaaS solution, given its availability and accessibility, you need to know what open source components are in your code for security reasons. OWASP, the Open Web Application Security Project, is known for its list of Top 10 web application security risks. Number 6 is "Vulnerable and Outdated Components." It is widely known that, in 2017, Equifax's system was breached and may have resulted in the release of the personal information of over 143 million Americans. The source of the breach was traceable to an open source component, Apache Struts, that contained a security vulnerability for which a patch was available, but it had not been implemented by Equifax. The House Oversight Committee concluded in 2018 that Equifax's security practices and policies were sub-par and Equifax could have prevented the massive data breach. Regardless of whether Equifax could have avoided the breach by keeping up to date with Apache Struts patches, the point remains that, if you do not have a firm grasp on what open source components make up your SaaS product, you cannot keep up to date with ongoing security issues. In an investment and acquisition context, it is important to complete focused diligence in this area to avoid buying a breach and lawsuit.

Operational considerations

Respecting the spirit of open source licenses

A core principle of open source is attribution—don't use the code without giving credit to the authors. As a result, it is best to have a process in place to inventory the open source components on which you are relying, and that carries all the way through to providing notice and attribution. Since most common open source licenses were drafted before SaaS use was widespread, the attribution obligations are often triggered by distribution. Some SaaS companies take the position that their offerings involve no distribution, so they do not need to provide attribution, which may be seen as going against core open source principles. Even though the industry may have decided that certain usage patterns are exceptions to, or otherwise permitted by, some open source licenses, such exceptions or use may not be within the spirit of such licenses and may be viewed negatively by the open source community, potentially creating reputational and operational issues.

As another example, many companies consider there to be a “dynamic linking” exception to the GNU Lesser General Public License (LGPL) requirement that the corresponding source code for applications that use LGPL-licensed components must be made available. However, relying on this exception may require that you grant certain reverse-engineering, modification, and relinking rights, which many companies wish to avoid. Additionally, a SaaS company’s reliance on this exception has the problem of potentially violating the spirit of the license, since code for a SaaS application is typically not made available to the customer in any form, so it would be impossible for a customer to reverse-engineer, modify, and relink it (even if the license terms were to allow it). As yet another example, many open source licenses, including the “public” licenses discussed above, may require that modifications made to the open source code that are then distributed, be made available under the same license terms as the open source code. Potentially skirting this obligation by considering hosted use of the code as not a distribution may violate the spirit of the license since the functionality of the modifications has been made available as included in the SaaS offering, but the code has not. Although violating the spirit of the open source licenses may not result in lawsuits against a company, the open source community at large may look upon your company unfavorably, and this could result in poor adoption of a company’s platforms or products.

Avoiding orphaned or dead projects

There are often heightened risks in using open source software where there is no active community behind the code. These components are typically more likely to have unidentified vulnerabilities, and users are essentially on their own to find and fix problems. In order to assess the vibrance of the various open source communities you are drawing upon, you first need to have a catalog of the components in your code base, and this is equally important for SaaS and on-premises offerings.

Strategic considerations

An on-premises solution may be required in the future

Although the general trend is certainly moving to the cloud, it is possible that you may want or need to switch to an on-premises/distributed model in the future. For example, increased governmental regulation of health and financial data could require particularly sensitive data be kept offline. In the wake of cloud security breaches, you may have one very important customer wary of the security of your company’s hosting provider and desiring some or all your service be provided to them from servers within their own infrastructure. Without foresight in the selection of open source components in your SaaS offering, any change in the distribution model may require significant amounts of re-engineering time, effort, and cost.

Even if your company is convinced that it will never create an on-premises solution, a potential acquirer may have different plans, and your decisions regarding open source review and compliance (including positions on the GPL family of licenses and other copyleft licenses) could complicate or even jeopardize a transaction or have an impact on price. In certain transactions (such as in an asset purchase), the transaction itself could be considered a distribution under certain licenses.

Surviving due diligence

An important thing to remember is that in due diligence, you may not be the decision-maker regarding what is acceptable. Positions you take regarding the use of open source in your SaaS applications may seem appropriate to you, but a potential investor or acquirer may have a more conservative view. This is important to keep in mind as you are making your decisions, including regarding the potential pitfalls noted above, both to guide them and to understand that sometimes even something that appears to be consistent with best practices may ultimately fall short in someone else’s view.

Conclusion

The use of open source software in cloud applications is not without risk. From the legal, security, operational, and strategic points of view, knowing what open source components make up your SaaS offering, and making sure that such use aligns with the obligations and conditions of the applicable open source licenses is just as important as with traditional software distribution models. Code scan solutions, such as those from Black Duck, and legal compliance audits will help ensure your company is comfortable with its use of open source software in the cloud.

For more information

Learn more about Black Duck® audits.

About Black Duck

Black Duck® offers the most comprehensive, powerful, and trusted portfolio of application security solutions in the industry. We have an unmatched track record of helping organizations around the world secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies. As the recognized leaders, experts, and innovators in software security, Black Duck has everything you need to build trust in your software. Learn more at www.blackduck.com.

©2024 Black Duck Software, Inc. All rights reserved. Black Duck is a trademark of Black Duck Software, Inc. in the United States and other countries. All other names mentioned herein are trademarks or registered trademarks of their respective owners. August 2024