THE LINUX FOUNDATION | Research

Open Source License Compliance

Challenges Ahead

Ibrahim Haddad, Ph.D., Vice President,
Strategic Programs (AI), *The Linux Foundation*

Foreword by Jimmy Ahlberg, Director,
Open Source Policy, *Ericsson*

January 2024

In partnership with
LF AI & DATA

Open Source License Compliance: Challenges Ahead

Open source software (OSS) license compliance requires **adhering to copyright notices and fulfilling license obligations** when incorporating OSS into products or services.

Ensuring compliance with OSS licenses can be complex and intricate, given the diverse range of licenses, the different terms and conditions, and the fast-paced nature of software development.

The process of compliance entails identifying all OSS incorporated in a product or a service and devising a plan to fulfill all applicable license obligations.

Effectively managing OSS license compliance requires an **advanced software composition analysis (SCA) tool** with comprehensive features.

Organizations can promote a culture of openness, accountability, and collaboration by **giving users visibility** into how they address compliance issues or inquiries.

By integrating compliance into the development process, organizations can reduce non-compliance risk while promoting a healthy internal open source governance culture.

Organizations can manage OSS license compliance effectively and at scale by **leveraging appropriate tools** and receiving internal support, mitigating compliance risks

SCA tools must be accurate and consistent, handle complexities, identify all OSS, and **stay updated with the OSS licensing landscape.**

Auditability is a critical challenge for organizations as they must maintain a transparent and comprehensive audit trail of all OSS and license compliance-related activities.

SCA tools should be able to integrate with the software development lifecycle and automatically scan code for open source components and licensing requirements.

Artificial intelligence–generated code presents new challenges that organizations can initially address using policy options and guidance provided to developers.

As OSS becomes more prevalent, it is **critical to establish a robust and automated compliance process** in place to avoid legal and reputational risks.

Contents

Foreword

"We are the middle children of history. Born too late to explore earth, born too early to explore space."

— ANONYMOUS

This line has stuck with me since I first heard it. This quote also holds relevance in open source and open source license compliance. I'm from a generation that has come into the industry after some great defining moments had already happened. I have only read the heated discussions on message boards and archived email lists after the fact and heard the "war stories" from those who were there then. The "exploration" was already done. In the ten years I have been part of this sprawling, confusing, complex, and wonderful community, I have participated in many heated and passionate discussions in person and online. I'm immensely grateful to those explorers, not only for the work they've done but also for their willingness to share their knowledge and expertise.

I remember the first time I came in contact with Ibrahim was when his e-book Open Source Compliance in the Enterprise came out. For me reading it was a true revelation. It was a guide to what had been explored by those before us and has helped shape my view, and many others with me, on what open source license compliance is and how it could and should be addressed methodically. This is a testament to Ibrahim's deep expertise in this field and his desire and ability to share that knowledge with others so that we can all improve collectively. Ibrahim is definitely one of the people who have come to define the field of open source compliance. Not only by being an explorer but also by guiding this field for many of us who were "born too late."

Before addressing the second part of the introductory quote of this foreword, I would like to paraphrase Lord Kelvin:

There is nothing new to be discovered in ~~physics~~ open source license compliance now. All that remains is more and more precise measurement.

Just as wrong as that statement was in 1897 for physics, just as wrong it would be for open source license compliance today. We might be standing on the shoulders of giants, which allows us to see further, and as Ibrahim points out in this thoughtful paper, our industry and technology have evolved. We now see further than before and have new challenges to explore. In particular, AI-generated code puts many new challenges for us to address.

When you read this paper, don't take it as a warning about our new challenges. Instead, read it as a guide to the future, a gauntlet thrown down, and a challenge to be addressed by all of us, the early explorers and those of us who came later. Together we can explore this new space. Perhaps we are the middle children of history in open source license compliance. If so, I would say that the challenges Ibrahim points out suggest this is a very interesting position to be in! I hope that you, by reading this paper, will come to share the excitement of these new challenges and take up the gauntlet trying to address these and all the other unforeseen challenges we will face in open source license compliance in years to come.

JIMMY AHLBERG
DIRECTOR, OPEN SOURCE POLICY, *ERICSSON*

Abstract

Open source software (OSS) has emerged as a critical element in modern software development, with many organizations leveraging its cost savings, flexibility, and innovation benefits. However, with the adoption of OSS comes the responsibility of complying with the licensing requirements. Non-compliance with open source licenses can have significant legal, financial, and reputational consequences for organizations and impact the overall open source ecosystem.

Open source license compliance can be challenging and complex due to the diversity of licenses, the varying terms and conditions, and the rapid pace of software development. The open source license landscape is constantly evolving, with new licenses released frequently; some are entirely new, whereas others are modifications of existing licenses.

Ensuring compliance with open source licenses requires a deep understanding of the licensing requirements, a well-defined compliance program, and effective tools and strategies to manage the process.

In this context, organizations must adopt best practices and implement effective compliance strategies to use OSS while maintaining compliance with all applicable open source licenses. This process involves understanding the implications of open source licenses, identifying open source components within software development projects, and fulfilling all license obligations.

This paper delves into some top challenges in open source license compliance. By understanding the complexities of these challenges and implementing effective compliance strategies, organizations can mitigate any associated risks and benefit from their participation in OSS projects to drive innovation and growth. In a future report, we aim to provide recommended practices to address these challenges.

Introduction

Traditionally, organizations built their software platforms and stacks using proprietary and third-party commercial software sourced through negotiated licensing terms. This approach made it easy to identify the provider of every software component in the software stack, and organizations mitigated potential risks through license and contract negotiations with their software providers or vendors. Over time, organizations began incorporating OSS into their platforms and software stacks due to its various advantages. Open source components provided compelling features, enabled faster time-to-market through distributed development, and allowed source code customization. This emerging practice led to the emergence of a new multi-source development model.

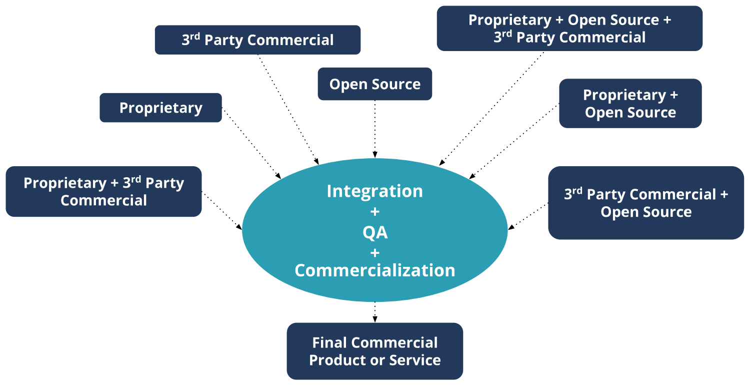

3rd Party Commercial
Proprietary + Open Source + 3rd Party Commercial
Open Source
Proprietary
Proprietary + Open Source
Proprietary + 3rd Party Commercial
Integration + QA + Commercialization
3rd Party Commercial + Open Source
Final Commercial Product or Service

FIGURE 1: Multi-source development model

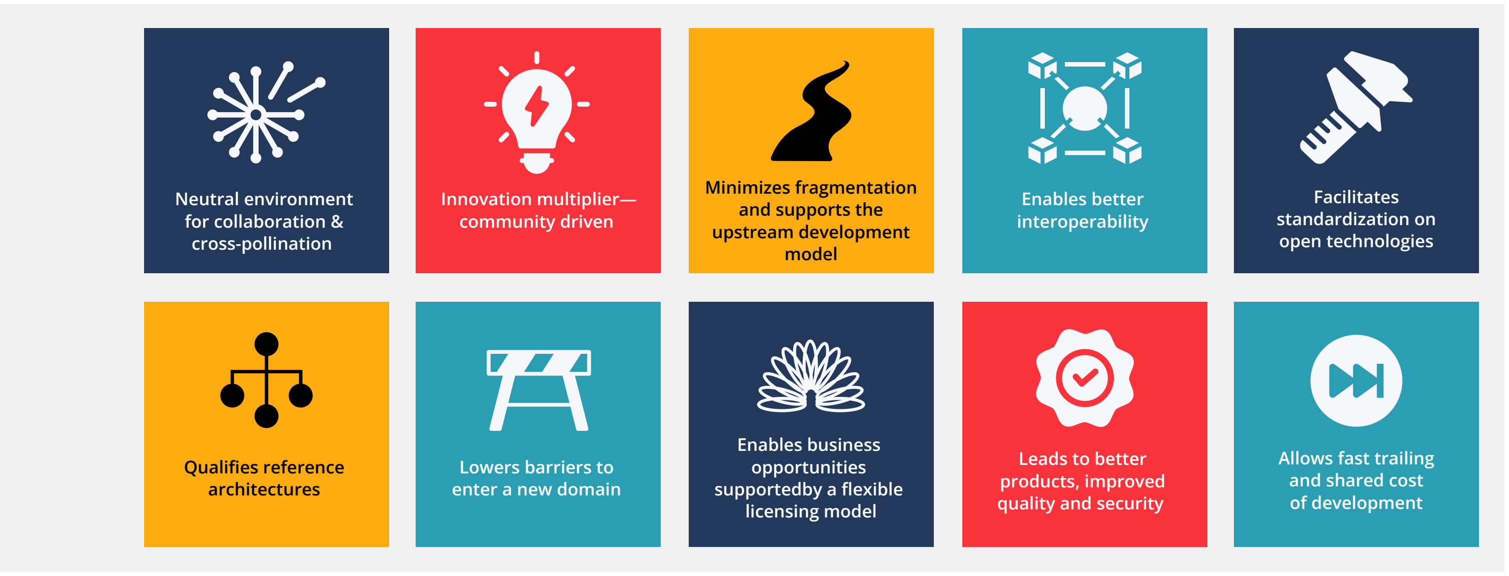

Neutral environment for collaboration & cross-pollination
Innovation multiplier—community driven
Minimizes fragmentation and supports the upstream development model
Enables better interoperability
Facilitates standardization on open technologies
Qualifies reference architectures
Lowers barriers to enter a new domain
Enables business opportunities supportedby a flexible licensing model
Leads to better products, improved quality and security
Allows fast trailing and shared cost of development

FIGURE 2: Advantages offered by the open source development model

Under this model, a product can now have any combination of the following:

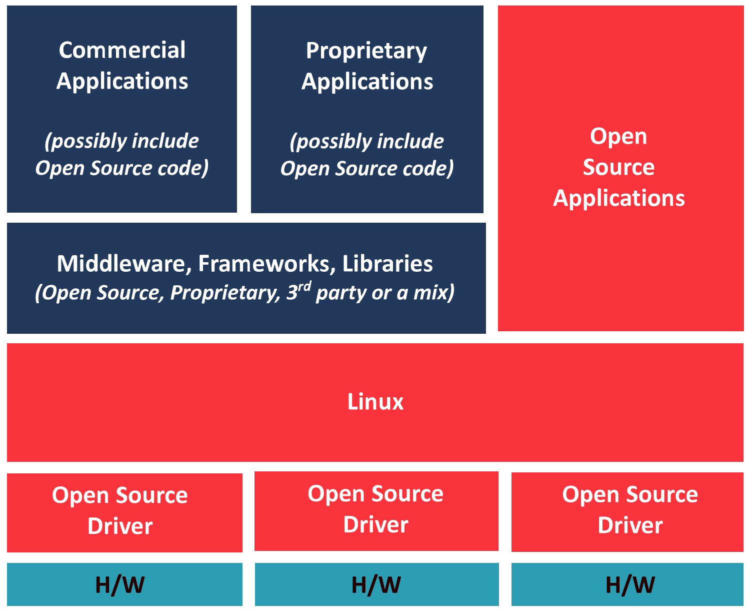
- Proprietary code developed by the company, which may contain open source code
- Proprietary code integrated with open source components but not contributed back to the upstream open source projects
- Third-party commercial code received under a commercial license, which may also contain open source code
- Open source code developed by the open source community and received by the company under an open source license
- Publicly available code that is neither open source nor proprietary, but under unknown or poorly understood custom licenses

While this new development model offers many advantages, it poses new challenges for organizations. They must ensure they comply with the relevant open source licenses, which can vary widely based on what OSS they incorporate in their stack. Failure to comply with the open source license obligations can lead, in some instances, to legal and financial risks.

Figure 1 illustrates the multi-source development model and the different combinations of sources for incoming source code. With this model, software components can consist of source code originating from various sources and be licensed under different licenses. For example, software component A can include proprietary and third-party source code. In contrast, software component B can consist of proprietary source code and source code from one or more open source projects.

As the amount of OSS incorporated into proprietary software stacks increased, the business environment became more complex and unfamiliar to many organizations entering the open source ecosystem and starting to integrate OSS into their products and services. This complexity led to challenges and potential risks not previously encountered in traditional proprietary software development. Organizations can now navigate the various licensing requirements of different open source components and ensure compliance with these licenses to mitigate legal and financial risks.

Figure 3 illustrates how organizations have adopted OSS across different platforms or software stack levels. One of the key differences between the proprietary and multi-source development models is that OSS licenses are not negotiated. Unlike with proprietary software, there are no contracts with software providers, typically open source developers or projects. Instead, the individuals who initiate an open source project choose a license, and once the project reaches a certain scale, it becomes virtually impossible to change the license.



Commercial Applications
(possibly include Open Source code)
Proprietary Applications
(possibly include Open Source code)
Open Source Applications
Middleware, Frameworks, Libraries
(Open Source, Proprietary, 3rd party or a mix)
Linux
Open Source Driver
Open Source Driver
Open Source Driver
H/W
H/W
H/W

FIGURE 3: A simplified architectural view of a modern software platform—open source has proliferated every building block

When using the multi-source development model, organizations must understand the implications of tens of different licenses and combinations of licenses that may come from thousands or even tens of thousands of licensors or contributors who are copyright holders. As a result, organizations must manage compliance risks through robust compliance programs and careful engineering practices rather than through company-to-company license and agreement negotiations.

This new development model pressured organizations to deeply understand open source licenses, their compliance obligations, and effective processes for tracking and managing OSS used throughout the development lifecycle.

Enter Open Source Compliance

Open source compliance is a crucial aspect of adopting OSS and incorporating it into products and services. Collaborating with open source communities can provide significant benefits, but it also comes with essential responsibilities.

Complying with open source licenses involves observing copyright notices and satisfying license obligations that arise from using OSS. An effective open source compliance program should ensure compliance with the terms of open source licenses while safeguarding a company's intellectual property and third-party suppliers from unintended disclosure or other adverse consequences.

By implementing a comprehensive open source compliance program, organizations can mitigate the risks associated with using OSS and fully realize the benefits of collaborating with open source communities.

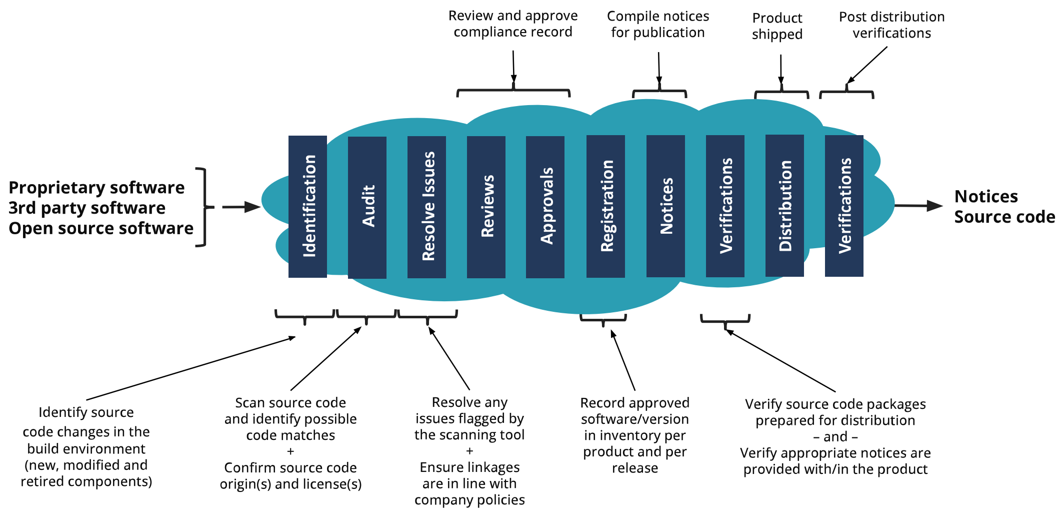

Review and approve compliance record
Compile notices for publication
Product shipped
Post distribution verifications
Proprietary software
3rd party software
Open source software
Identification
Audit
Resolve Issues
Reviews
Approvals
Registration
Notices
Verifications
Distribution
Verifications
Notices
Source code
Identify source code changes in the build environment (new, modified and retired components)
Scan source code and identify possible code matches + Confirm source code origin(s) and license(s)
Resolve any issues flagged by the scanning tool + Ensure linkages are in line with company policies
Record approved software/version in inventory per product and per release
Verify source code packages prepared for distribution – and – Verify appropriate notices are provided with/in the product

FIGURE 4: Overview of an open source compliance process

Implementation of open source compliance processes can vary from company to company based on several factors: the underlying product development process into which compliance must fit, the size and nature of the code base, the number of products turned out, the amount of externally supplied code, the size and organizational structure of the company, and so on. However, the core compliance elements usually remain the same:

1. Identifying the open source in the code base
2. Reviewing and approving its use
3. Satisfying open source license obligations

The compliance due diligence process identifies all OSS used in a product intended for external distribution and a plan to meet the attendant license obligations.

Figure 4 offers a high-level overview of a sample end-to-end compliance process and illustrates the various compliance steps or phases that components containing OSS go through before they get approved for use in a product intended for external distribution. Other ways of organizing the compliance process may well accomplish the same goals of ensuring compliance.

An effective compliance process will include several steps that vary from one organization to another based on how they structure and govern their internal open source efforts:

- Identification of all software entering the organization
- Auditing of all source code
- Resolving any issues uncovered by the audit
- Completing appropriate reviews

- Receiving approval to use OSS
- Registering all used OSS in the software inventory
- Updating end-user documentation to reflect OSS used in the product
- Performing verification of all previous steps before distribution
- Distributing open source package's code, including modifications (if any) when applicable
- Performing final verifications

For a detailed discussion of the source compliance process, we recommend the e-book **Open Source Compliance in the Enterprise (2nd Edition)**, published by the Linux Foundation. The e-book outlines best practices for organizations to adopt and use open source code in products and services as well as participate in open source communities in a legal and responsible way.

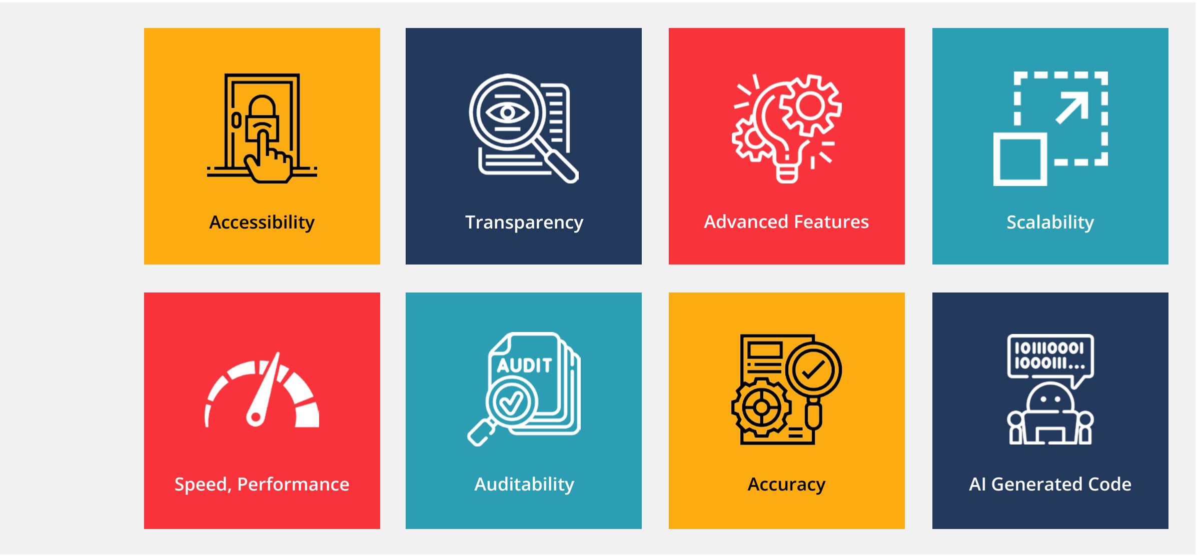


FIGURE 5: Core compliance challenges for the next decade

New Era, New Challenges

The landscape of open source compliance has evolved significantly in recent years, with more complex challenges emerging as the use of OSS becomes increasingly prevalent in software development. In the past 20 years, we have addressed challenges such as identifying and attributing open source components, ensuring compliance with license terms, and establishing processes to manage the consumption of and contribution to OSS. However, today's challenges are more complex due to the proliferation of OSS, increased use of cloud services, new licensing regimes, and AI-generated code trained using OSS.

As the use of OSS continues to grow, organizations face the challenge of managing compliance across their entire software stack at a larger scale than ever experienced before. Hence, sophisticated compliance strategies and tools are required to address new challenges to support enterprises in ensuring they can effectively manage compliance across their entire software stack.

In the following sections, we will explore eight key challenges:

- Accessibility
- Transparency
- Advanced features
- Scalability
- Speed
- Accuracy
- Auditability
- AI-generated code

Accessibility

Ensuring accessibility is a significant challenge in open source license compliance. With numerous open source licenses in use, organizations must have the necessary tools and resources to identify and comply with the terms of these licenses. However, it is crucial to consider that not all developers have the same expertise with open source licensing. Therefore, the tools used to support open source compliance activities should be easily accessible to all developers, regardless of their experience level. Ensuring compliance starts with developers, where the intake of OSS and contribution to open source projects happen.

Deployed compliance tools should have a user-friendly interface, clear documentation, and accessible training resources to help users navigate the tool. With the developers being directly engaged in the compliance process, they can understand the licensing requirements and ensure compliance with the terms of the licenses. In addition, the compliance tools should easily integrate into the software development process, with seamless integration with other tools used by the development team. Such integrations will support the compliance efforts by making them an integral part of the development process rather than a separate task looked at as an overhead to the engineering and development effort.

By making compliance accessible and integrated into the development process, organizations can reduce non-compliance risk while promoting a culture of good open source governance.

Transparency

Transparency is a vital aspect of open source license compliance that helps to foster trust and credibility between organizations adopting OSS and the open source project communities. Organizations must have access to reliable and transparent compliance tools that identify all OSS incorporated in their products and services and help them establish the mechanism to comply with all applicable open source licenses. To achieve transparency, organizations should have an issue tracker, which allows individuals to report potential compliance concerns or discovered issues and a transparent process for handling these issues and inquiries. Regular communication with users is crucial for informing them about the progress of looking into their concerns.

Providing a Software Bill of Materials (SBOM) is becoming a common method of providing transparency to users of a product or service. These SBOMs provide visibility into component usage, open source licensing and potentially vulnerability status of the open source components in use.

Transparency is critical for building trust and credibility in open source license compliance. Organizations can promote a culture of openness, accountability, and collaboration by giving users visibility into how they address compliance issues or inquiries, demonstrating a commitment to responsible open source development practices.

Advanced Feature Set

Open source license compliance can be challenging, especially in today's increasingly complex software development landscape. As OSS becomes more prevalent, it is essential to have a robust compliance process in place to avoid legal and reputational risks. A compliance tool with an advanced feature set must complement complex development and ensure comprehensive compliance. The compliance tool should have advanced features such as automated license identification, enabling developers to identify open source components and their associated licenses. Additionally, the tool should have license obligation management features that guide its users on complying with the license terms and obligations. It should also have risk analysis capabilities to identify and recommend mitigating potential compliance risks based on the organization's internal policies. Furthermore, the compliance tool should support integration with other tools the development team uses, such as source code repositories, continuous integration and delivery systems, and project management tools. Such integrations will help ensure that the compliance tool can seamlessly be connected to the existing development workflow and used by all development team members.

It is important that tools discover and manage all open source artifacts that are used to build and run an organization's products and services. Many current SCA tools only look at components that are managed by package managers. This type of scanning does not examine important artifacts like source code files, non-package managed binary libraries, and cut & pastes of source code. Additionally, many of these tools only look at top-level licensing for a component and its transitive dependencies. Complete open source compliance requires intra-package scanning or passing through of pre-scanned SBOM-like open source disclosures for the third-party components.

Additionally, the tool should support different software development environments, such as containers, cloud-native applications, and serverless architectures. Let's take containers as an example. Containers enable developers to package software in a lightweight and portable format, making it easier to deploy and manage. However, containers also introduce new open source compliance challenges, such as managing the licensing of container images and ensuring compliance across different container registries. A compliance tool with advanced container compliance features can help ensure container images comply with open source licensing requirements and minimize non-compliance risk.

We recommend **An Open Guide To Evaluating Software Composition Analysis Tools**, published by the Linux Foundation, for more information about evaluating open source license compliance tools.

Scalability

When it comes to ensuring open source license compliance at a large scale, scalability is a critical factor. As organizations continue to rely on OSS, the volume of code and number of users can quickly become overwhelming. Therefore, it is essential to have a compliance infrastructure and supporting tools to handle the scale and complexity of wide adoption and OSS in products and services.

However, scalability is about more than just the tool's technical capabilities. It also requires organizational support, such as having dedicated resources and personnel to manage the compliance process at scale. It can include compliance specialists, training programs, and governance policies ensuring compliance is integral to the software development process. It is becoming common for organizations to create an Open Source Program Office (OSPO) to provide these internal services.

Ensuring open source license compliance at a large scale requires a compliance tool with a scalable architecture, optimized performance, and support for integration with other tools. It also requires organizational support, such as dedicated resources and personnel. Some organizations may decide to use outside services to provide staff augmentation, or in order to perform baseline compliance audits.

Speed

The speed of ensuring compliance for all incoming open source code is a significant challenge as organizations must keep up with their internal development pace and the volumes of OSS incorporated in their products and services. Any organization's open source license compliance infrastructure should provide results quickly without slowing down the development process. An organization can address this challenging goal by implementing a lightweight compliance policy and process, automated using a tool that can scan code at lightning speed and identify open source components and licensing requirements quickly and accurately. Furthermore, the compliance tool should provide an API that allows other tools to query for licensing information and a command-line interface that enables developers to scan code and receive results quickly. Organizations should be aware of any tradeoffs and risks associated with trading speed for more complete analysis and compliance.

Another essential aspect of speed is the ability to automate the compliance process fully. The compliance tool should be able to integrate with the software development lifecycle (SDLC) and automatically scan code for open source code and licensing requirements. Such integration will help to reduce the workload of the dedicated compliance team, ensure consistency and accuracy in the compliance process, and accelerate the development process.

Accuracy

Accuracy is essential in open source license compliance as it ensures that organizations comply with the licensing requirements of open source components. Compliance tools should be able to accurately identify open source components, detect license obligations, and provide risk analysis. They should also be able to handle complex scenarios, such as when multiple licenses apply to the same component or when there are license dependencies between components.

Furthermore, compliance tools should provide reliable results even when dealing with large or legacy code bases. They should be able to handle different programming languages and frameworks and be flexible enough to adapt to changes in the code base or licensing requirements.

Since open source compliance relies heavily on accurate open source license detection, copyright management and license obligation management, it is important to select a tool with strong license detection and management features and data. It is common for tools to show a "best guess" instead of the actual license text present in the code and lack the ability to create accurate license disclosures. Having the ability to discover modified open source licenses is an important compliance feature since these modifications can greatly change the obligations required by the license.

Additionally, open source license compliance quality varies greatly in the open source community itself. The Compliance tool should be able to alert on mismatches between discovered licenses and the top-level license disclosed by the open source project or seen in the repository manager meta-data.

The compliance tool should be regularly updated to keep pace with changes in open source licensing and new open source components. Organizations can minimize compliance risks and meet their open source licensing obligations with accurate compliance tools.

Auditability

Auditability is important in open source license compliance because organizations must provide a clear audit trail of all activities related to open source license compliance. The compliance infrastructure should provide auditability facilities, which means that organizations should be able to track changes and identify who made them and when. The tooling deployed should also be able to provide a history of compliance activities, including any exceptions or deviations from the standard process. This aspect is essential for legal compliance, risk management, and building trust with customers and open source communities. Additionally, the tools used should be able to generate reports that can be used for auditing purposes, demonstrating compliance with open source license requirements.

Artificial Intelligence–Generated Code[1]

With recent advancements in AI, AI-generated code is fueling a new wave of developer empowerment by automating the process of writing software. Artificial intelligence tools are capable today of supporting developers in the generation of functions, classes, algorithms, and even entire programs based on high-level instructions provided by developers. Such tools have proven extremely beneficial for rapid prototyping and experimentation and can quickly generate functional code snippets to test ideas and concepts. While AI-generated code shows a lot of promise and is helping bridge the skills gap in software development for less experienced developers, it presents its own set of challenges. Ensuring license compliance with AI-generated code can be a complex task regarding the licenses of the code the AI system was trained on, the license of the code the AI system is reproducing in whole or in part in its output, whether that output includes verbatim copies of open source code, and whether that code may be considered a derivative work.

For this paper, we would like to raise specific challenges to address when using AI-generated code and incorporating it either in open source components or software that is developed internally for any given organization:

1. **Copyright challenges:** Generative AI tools reproduce portions of materials (in this case, source code) that they were trained on, some of which may be copyrightable subject matter.
2. **License compatibility challenges:** In the case of publicly available OSS that AI models were trained on, the OSS license applicable to the pre-existing OSS code reproduced in the AI output may be incompatible with the applicable project license.
3. **License compliance challenges:** Many AI tools do not currently provide information about the origins and license of the source code they provide as output to prompts. Therefore, even if these AI systems or models were trained using only permissively (or otherwise compatible) licensed OSS code, the challenge of license compliance still exists: How can downstream users comply with the notice and attribution requirements of the license terms applicable to source code reproduced in the AI's output without knowing who the copyright holders are or the license terms?
4. **Software bill of materials (SBOM) challenges:** When information about the origin and license of the source code provided by the AI system is missing, down-stream users cannot produce an accurate SBOM. This situation raises a specific red flag due to the inability to track security vulnerabilities.

However, not all of these challenges are new to open source. Even without generative AI, there is the risk that a contributor will copy materials they are not authorized to, or source code that is incompatibly licensed, and contribute them to an open source project. However, generative AI can present this risk at a broader scale and systematized manner versus isolated instances of individual developers contributing to third-party code irresponsibly.

1 Original work by Joanna Lee, Vice President of Strategic Programs (Legal), Linux Foundation. Please refer to the Acknowledgment section.

How do we address these concerns and challenges?

Although the paper's goal focuses on presenting open source license compliance challenges, generative AI is such a hot topic that we will expand the discussion further than the other challenges mentioned above.

Policies and guidelines can be used to mitigate any license compliance risks derived from using AI-generated code. There are several policy options that an organization can adopt when considering AI-generated source code:

Scenario 1: Conservative

In this scenario, the organization might adopt a conservative approach and advise its developers not to use AI tools for code generation.

Scenario 2: Selective

a. **By Use:** In this scenario, the organization might adopt a permissive approach and conditionally allow for some uses based on context (e.g., debugging) but not accept other uses and contexts, such as generating code to implement new functionalities.

b. **By Tool:** In this scenario, the organization might adopt a more permissive approach by conditionally allowing its developers to use and adopt contributions from AI systems or tools that provide notice and attribution when outputting source code that originated from an open source project. In such cases, the organization can ensure compliance and license compatibility while banning AI tools that don't provide provenance information for open source code included in the output.

Scenario 3: Empower Developers

In this scenario, the organization might allow its developers to use AI tools for code generation while providing recommendations and guidelines for using such tools.

One way to address this issue is to carefully review the licenses of any open source code used in AI training data or algorithms and to ensure that any resultant code is appropriately licensed under those terms. It's also important to keep detailed records of the sources of any open source code used in AI training or generation so that you can trace the origin of any resultant code and ensure that it complies with the applicable open source licenses.

Conclusion

Open source license compliance can be challenging, but organizations can overcome these challenges using the right tools and strategies. The tools to support open source license compliance efforts should be easily accessible to all developers, provide transparency on open issues, have an advanced feature set, be scalable, provide speedy results, and be accurate and auditable. In addition to using the right tools, organizations should have clear, lightweight, and concise policies and processes for open source license compliance, provide training and support for developers, and regularly review and update their compliance practices. By doing so, organizations can ensure they comply with open source licensing requirements and build trust with the open source community.

The next years in the license compliance realm will address the challenges discussed in this paper and some other challenges that we will uncover along the way. The Linux Foundation hosts several efforts that bring together organizations to collaborate on addressing various challenges in the open source ecosystem. We invite you to join us and work throughout our initiatives to advance the state of open source license compliance.

Feedback

The author apologizes in advance for any errors and is grateful to receive corrections and suggestions for improvements.

Acknowledgments

The author would like to express his gratitude to the Linux Foundation Research staff and leadership for their invaluable contributions to the development of this paper.

Special thanks to **Jimmy Ahlberg** (Director of Open Source Policy at Ericsson) for contributing the Foreword and discussing the topic during the Open Source Summit EU.

Jeffrey Luszcz (Director of Open Source, PEAK6) has made significant contributions (time and edits) by reviewing the paper, offering suggestions and edits to add ideas and clarify content that has led to significant improvement to the discussion.

Major thanks as well go to **Joanna Lee**, Vice President of Strategic Programs (Legal) at the Linux Foundation, for her research work on Generative AI as it relates to open source compliance, as summarized in the paper under the section “Artificial Intelligence–Generated Code.”

Thank you all!

Linux Foundation References

Efforts

1. **OpenChain:** The OpenChain Project is a community effort hosted by the Linux Foundation to develop standards and best practices for open source license compliance.
2. **SPDX**: The Software Package Data Exchange (SPDX) is a standard for identifying the contents of software packages and the licenses that apply to them. SPDX provides a standardized way of documenting open source license compliance, which can help businesses manage their compliance obligations more efficiently.
3. **Open Compliance Program**: The Linux Foundation's Open Compliance Program provides resources and tools for businesses looking to improve their open source compliance practices. The program includes a set of best practices and training materials for managing compliance obligations.
4. **TODO Group**: The TODO Group is a community of organizations collaborating to establish best practices, tools, and programs for OSS development. The group focuses on issues such as open source program management, governance, and compliance and provides resources such as guides and training to help organizations navigate these topics.

Training

- **Open Source Licensing Basics for Developers**
- **Implementing Open Source License Compliance Management**
- **Introduction to Open Source License Compliance**
- **Generating a Software's Bill of Materials**

Events

- **Open Compliance Summit**
- **Linux Foundation Legal Summit**

E-Books and Papers

1. **Open Source Guides for the Enterprise**
2. **An Open Guide To Evaluating Software Composition Analysis Tools**
3. **Recommended Open Source Compliance Practices for the Enterprise**
4. **Assessment of Open Source Practices in M&A Transactions**
5. **Open Source Audits in Merger and Acquisition Transactions**
6. **Publishing Source Code for FOSS Compliance: Lightweight Process and Checklists**
7. **A Glimpse into Recommended Practices in FOSS Compliance Management Process**
8. **Free and Open Source Software Compliance: The Basic You Must Know**
9. **A Five-Step Compliance Process for FOSS Identification and Review**
10. **FOSS Compliance: Who Does What—Roles and Responsibilities**
11. **Practical Advice to Scale Open Source Legal Support**
12. **Achieving FOSS Compliance in the Enterprise**
13. **Establishing Free and Open Source Software Programs: Challenges and Solutions**

About the Author

Ibrahim Haddad (Ph.D.) is the Vice President of Strategic Programs for AI and Data at the Linux Foundation. He is focused on advancing the open source AI platform and empowering generations of open innovators. He leads both the LF AI & Data Foundation and the PyTorch Foundation. Before joining the Linux Foundation, he served as the Vice President of R&D and the Head of the Global Open Source Division at Samsung Electronics. Haddad had previously held technology and R&D management roles at Ericsson Research, the Open Source Development Labs, Motorola, Palm, and Hewlett-Packard.

LinkedIn | **Website**

Founded in 2021, **Linux Foundation Research** explores the growing scale of open source collaboration and provides insight into emerging technology trends, best practices, and the global impact of open source projects. Through leveraging project databases and networks and a commitment to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations the world over.

To reference the work, please cite as follows: Ibrahim Haddad, Ph.D., "Open Source License Compliance: Challenges Ahead," foreword by Jimmy Ahlberg, The Linux Foundation, January 2024.