OPEN SOURCE AUDITS IN MERGER AND ACQUISITION TRANSACTIONS

THE BASICS YOU MUST KNOW

Ibrahim Haddad, Ph.D.

In this paper, Haddad provides an overview and a practical guide on open source audits in merger and acquisition (M&A) transactions, and offers basic guidelines to improve open source compliance preparedness for both the target company and the acquirer.

1. Introduction

We live in an era defined by software. Virtually everything we do on a daily basis is in some way planned, shaped, analyzed and managed by software. Within that large software umbrella, open source software is king. Companies across all industries are racing to use, participate in, and contribute to open source projects for the various advantages they offer, from the ability to leverage external engineering resources that accelerate time to market, to enabling faster innovation, and having capacity to focus on differentiating values.

The saying "Open Source is Eating the Software World" also applies to corporate transactions, as virtually any technology acquisition will involve software in some form. The software due diligence process, in which the acquirer performs a comprehensive review of the target's software and their compliance practices, is becoming a standard part of any merger or acquisition. During this process it's common to come across open source software, which presents a set of verification challenges that are different from proprietary software.

In this article, we provide an overview of the open source audit process in M&A transactions. We expect to publish a second paper in this series to explore the various insights (technical, legal, business) that companies gain when going through this process).

2. Common open source usage scenarios

Before diving into the open source due diligence process, it helps to understand the various ways that open source software can be incorporated into a target's development process. This applies to situations where the company knowingly or unknowingly incorporates open source software into their source code base. As with traffic tickets, ignorance of your obligations is no excuse, so it is wise to understand the various ways that software from multiple sources can be used. The most common use scenarios of open source software are incorporation, linking, and modification.

Making changes to open source components, or injecting open source code in proprietary or 3rd party components, can affect the way that audit service providers discover and report such code. When engaging with an open source audit provider, it is often helpful to understand how their discovery approach captures open source code.

2.1 Incorporation

A developer may use a complete open source component or copy portions of a component into their software product's codebase. Since open source licenses come with a variety of obligations that may impact the company's legal responsibilities and the proprietary nature of their code, all such incorporation should be tracked, declared, and approved internally.

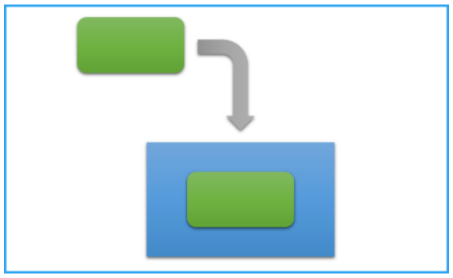

Figure 1: Incorporating open source code (green) within another body of code (blue)

コメントの追加 **[tani1]:** In Japanese translated version, I added dotted pattern into green boxes (for people who have vision difficulties in color).

Source code audits are designed to find undeclared incorporation of open source into a codebase, to avoid unpleasant surprises post-acquisition. The likelihood of undeclared incorporation increases when the target has not had sufficient developer training on open source compliance, or has relied upon transient worker like contractors or interns who don't maintain long term records.

The incorporation scenario is often not obvious when human eyes look at source code, but source code scanning tools with the ability to discover and match snippets can easily uncover such incorporation.

2.2 Linking

Linking is a very common scenario for instance when using open source libraries. In this scenario, a developer may link an open source software component with their software component (Figure 2). There are several terms that can refer to such a scenario such as static/dynamic linking, combining, packaging, or creating interdependencies. It is often easy to detect linking when visually scanning source code because libraries are generally included at the beginnings of files and the linked code is likely to be in a separate named directory or file.

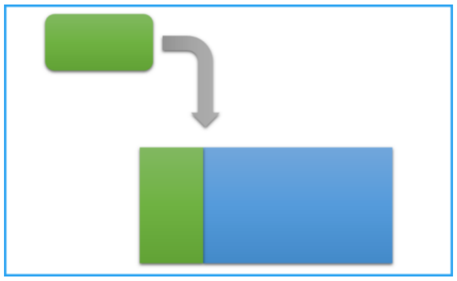

Figure 2: Linking open source code (green) with another body of code (blue)

コメントの追加 **[tani2]:** In Japanese translated version, I added dotted pattern into green boxes (for people who have vision difficulties in color).

Linking differs from incorporation in that the source code is kept separate, rather than being copied into a single combined form. Linking interactions happen either when the code is compiled into a single executable binary (static linking), or when the main program runs and calls the linked program (dynamic linking).

2.3 Modification

This is a very common scenario where a developer may make changes to an open source software component (Figure 3), including:

- Adding/injecting new code into the open source software component.
- Fixing, optimizing or making changes to the open source software component.
- Deleting or removing code.

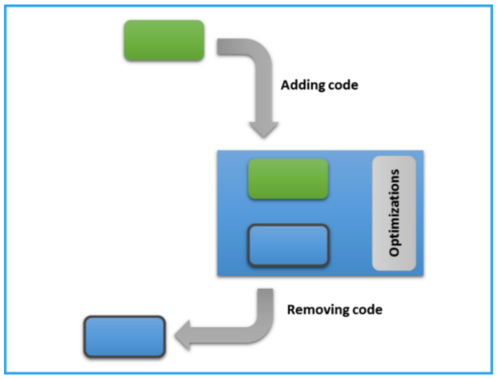

Figure 3: Modifications applied by developers to open source code (green)

コメントの追加 **[tani3]:** In Japanese translated version, I added dotted pattern into green boxes (for people who have vision difficulties in color).

2.4 Note on development tools

It is important to be aware that certain development tools may perform some of these operations transparently. For example, a developer may use a development tool that automates certain portions of the development process. Examples of this include graphics frameworks that provide user interface templates, game development platforms that provide physics engines, or software development kits (SDKs) that provide connectors to cloud services. In order to provide these services, a tool must usually inject portions of its own code into the developer's work product when the code is built. The license for such injected code by development tools should be verified especially given the resulting work is often statically linked.

3. Open source audits

Every M&A transaction is different, but the need to verify the impact of acquiring open source obligations is a constant among all such deals. Open source audits are carried out to understand the depth of use and the reliance on open source software. In addition, they offer great insights about any compliance issues and even the target's engineering practices.

3.1 Why conduct an open source audit?

Open source licenses may impose restrictions on how software can be redistributed. These may be incompatible with the acquiring company's business, and should be uncovered early. Examples of ways the presence of open source software can impact the acquired assets include:

- Open source licenses usually impose certain obligations that must be fulfilled when code is distributed. One example is the GNU General Public License (GNU GPL), which requires derivatives or combinations to be made available under the same license as well. Other licenses require certain notices in documentation, or have restrictions for how the product is promoted.
- Failure to satisfy open source license obligations can lead to possible litigation, expensive re-engineering, product recalls, and bad publicity.

3.2 Should you commission an open source audit?

One common question is whether an open source audit is needed at all. The answer to that question differs by company, purpose of acquisition, and size of the source code. For instance, for small acquisitions, some companies prefer to just review the open source bill of materials (BoM) provided by the target (assuming it is available), and have a discussion with their engineering lead about their open source practices. Even if the purpose of the acquisition is to acquire the talent, an audit can help uncover whether there are undisclosed liabilities due to historical license obligations from products which already shipped.

3.3 Inputs and outputs

The audit process has one primary input and one primary output (Figure 5). The input to the process is the complete software stack subject to the M&A transaction that is being conducted. This includes proprietary, open source and 3rd party software. On the end side of the process, the primary output is a detailed open source software bill of material that lists:

コメントの追加 **[tani4]:** Figure 5 is missing. I already identified which is, so let me confirm via email (but please give me a caption, i.e. title of Fig5)

- All open source software used as component, their origin and confirmed licenses, and
- All open source snippets used in either proprietary or 3rd party software, their originating components and confirmed licenses.

4. Assessing the scope of an audit job

The size, scope, and cost of an audit varies by transaction, and generally increases with source code size and complexity. To provide a quote (cost and time) for an open source audit the auditor needs to get some basic understanding of the size and characteristics of the code base, as well as the urgency of the project.

The first questions from the auditor will be related to code metrics, such as the size of the source code base, the number of lines of source code, and the number of files that need to be audited. They will also ask if the codebase consists exclusively of source code, or if it includes binary files, configuration files, documentation, and possibly other file formats. Sometimes, it is also helpful for the auditor to know the file extensions subject to the audit.

Mature companies generally keep records about the open source components and versions used in their products and projects. Such information is very helpful and increases an auditor's understanding of the expected workload.

Because audit price discussions happen early in the process based on size and scope, the acquirer may not have access to all the information described above. At the very minimum, the auditor needs to understand the number of files to be scanned before proceeding, although additional information will help refine the estimates. When the auditor has enough information to understand the scope of the work, they will also need to understand the urgency, as this has a significant impact on the cost of an audit.

5. Audit methods

When performing an open source audit there are certain features in the tools that provide meaningful value to the acquirer. One of the most important features is the ability to search for open source code snippets that have been mixed into the proprietary code of the target company, and vice versa. Another feature is the ability to automatically eliminate false positives from the audit results minimizing the amount of labor needed to so manually.

There are three audit methods:

1. Traditional audit, in which the auditor gets complete access to all the code and executes the audit either remotely or on site.
2. Blind audit, in which the auditor does the work remotely and without ever seeing the source code.
3. "Do It Yourself" audit, where the target company or the acquirer performs most of the actual audit work themselves, with the tools, support or even the option for an random verification of results from the auditing company.

コメントの追加 **[tani5]:** Typo?

5.1 Traditional audit method

This method is named traditional by the author as it is the original method of source code scanning for open source compliance purposes. Traditional audits are those where a compliance auditor from a 3rd party auditing company gets access to the source remotely via a cloud system or physically while visiting on site and performs the source code scan.

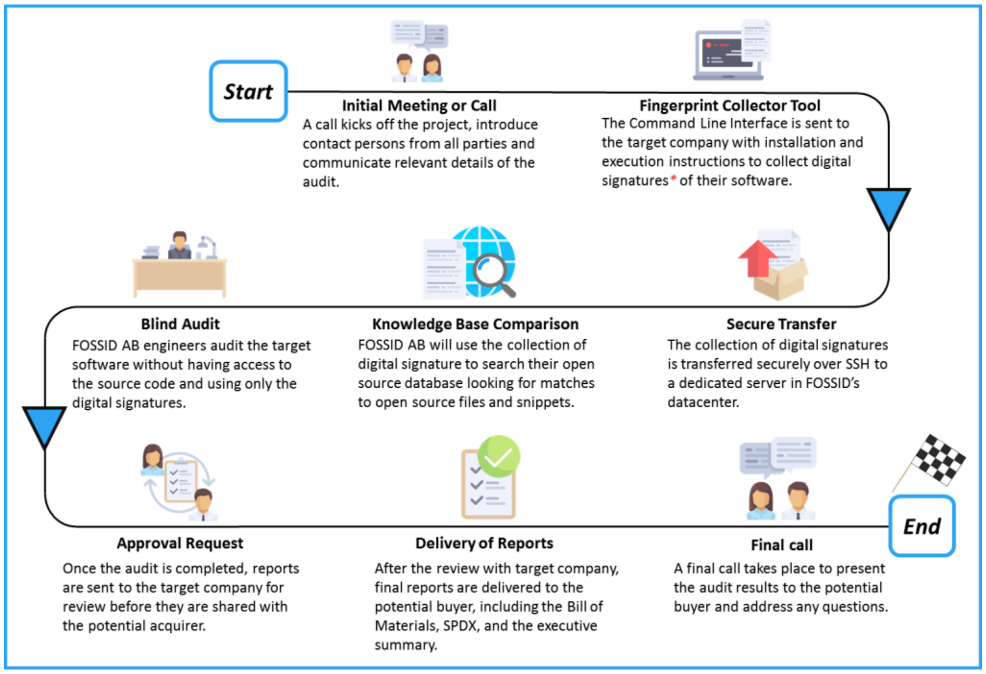
Start
Initial Meeting or Call
A call kicks off the project, introduce contact persons from all parties and communicate relevant details of the audit.
Fingerprint Collector Tool
The Command Line Interface is sent to the target company with installation and execution instructions to collect digital signatures* of their software.
Blind Audit
FOSSID AB engineers audit the target software without having access to the source code and using only the digital signatures.
Knowledge Base Comparison
FOSSID AB will use the collection of digital signature to search their open source database looking for matches to open source files and snippets.
Secure Transfer
The collection of digital signatures is transferred securely over SSH to a dedicated server in FOSSID's datacenter.
Approval Request
Once the audit is completed, reports are sent to the target company for review before they are shared with the potential acquirer.
Delivery of Reports
After the review with target company, final reports are delivered to the potential buyer, including the Bill of Materials, SPDX, and the executive summary.
Final call
A final call takes place to present the audit results to the potential buyer and address any questions.
End

Figure 6: Illustration of the traditional audit method in M&A transactions

コメントの追加 **[tani6]:** I think this figure is for "5.2 Blind audit" I attach candidate figure in the email.

Figure 6 illustrates the audit process following the traditional auditing method. Please note that the process may vary slightly from one service provide to another. A typical traditional audit process follows these steps:

- Auditor sends questions to the acquirer to have a better understanding of the job.
- Acquirer responds allowing auditor company to have a better understanding of the scope and audit parameters.
- Auditor provides quote based upon the responses.
- Agreement is reached on the quote. Next is singing service agreement, statement of work, non-disclosure agreement, etc.

*<Please note that "**Start**" in Figures 7, 8, and 9, assumes an actual start of the audit process when all agreements have been signed.>*

コメントの追加 **[tani7]:** I think this refers to Figures 6,7,and 8. Is that right?

- Auditor accesses to the target's code via secure cloud upload, or through a visit to the company for an on-site audit.
- Auditor scans the target's source code, cleans up the false positives, and evaluates the results.
- Auditor generates the report and delivers it to the client.
- A call or a face-to-face meeting follows to review the results with the auditor and address any questions.

This method is common across most audit service providers. It allows the opportunity to collect multiple bids for the same audit job and the ability to choose the best bid given your requirements. Following this model, the target company must be willing to transfer the code to the auditors or allow them to visit their offices to complete the job on-site.

5.2 Blind audit

The blind audit method was pioneered by FOSSID AB[1], a Stockholm based company, to address the confidentiality requirements of M&A transactions.

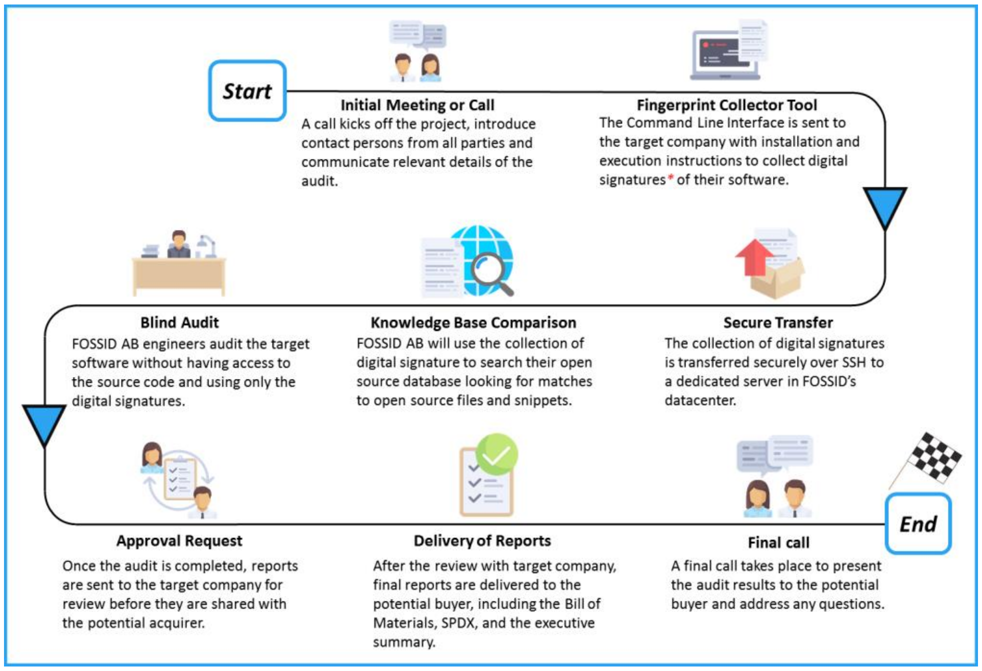
Start
Initial Meeting or Call
A call kicks off the project, introduce contact persons from all parties and communicate relevant details of the audit.
Fingerprint Collector Tool
The Command Line Interface is sent to the target company with installation and execution instructions to collect digital signatures* of their software.
Blind Audit
FOSSID AB engineers audit the target software without having access to the source code and using only the digital signatures.
Knowledge Base Comparison
FOSSID AB will use the collection of digital signature to search their open source database looking for matches to open source files and snippets.
Secure Transfer
The collection of digital signatures is transferred securely over SSH to a dedicated server in FOSSID's datacenter.
Approval Request
Once the audit is completed, reports are sent to the target company for review before they are shared with the potential acquirer.
Delivery of Reports
After the review with target company, final reports are delivered to the potential buyer, including the Bill of Materials, SPDX, and the executive summary.
Final call
A final call takes place to present the audit results to the potential buyer and address any questions.
End

Figure 7: Illustration of a blind audit process using FOSSID targeted for M&A transactions

Using their proprietary technology, they have the ability to perform audits and generate reports without looking at the source code. Figure 7 illustrates the blind audit process used by FOSSID AB and designed to provide confidentiality of source code in M&A transactions. One major advantages of a blind audit include the ability for the auditor to complete the review

[1] Please note that FOSSID AB refers to the company and FOSSID refers to the tool.

without having access to the source code. In addition, with sufficient precautions by the acquirer, the auditor may also not gain awareness of the target's identity offering a high level of confidentiality. As far as the author is aware, such audit method is not offered by any other company offering open source compliance services.

5.3 DIY audit

The Do-It-Yourself audit provides the acquirer or the target company time-limited access to the compliance cloud tools, enabling them to run the scan themselves. They can then perform the audits internally with complete access to the knowledge base and all reporting facilities. This is an approach that is particularly interesting for companies that have in-house employees with sufficient experience to interpret scan results and suggest remediation procedures. It can quickly become more cost-effective for companies that go through the M&A process several times per year. An independent certification can be performed to verify the findings, to further secure the integrity of the audit.

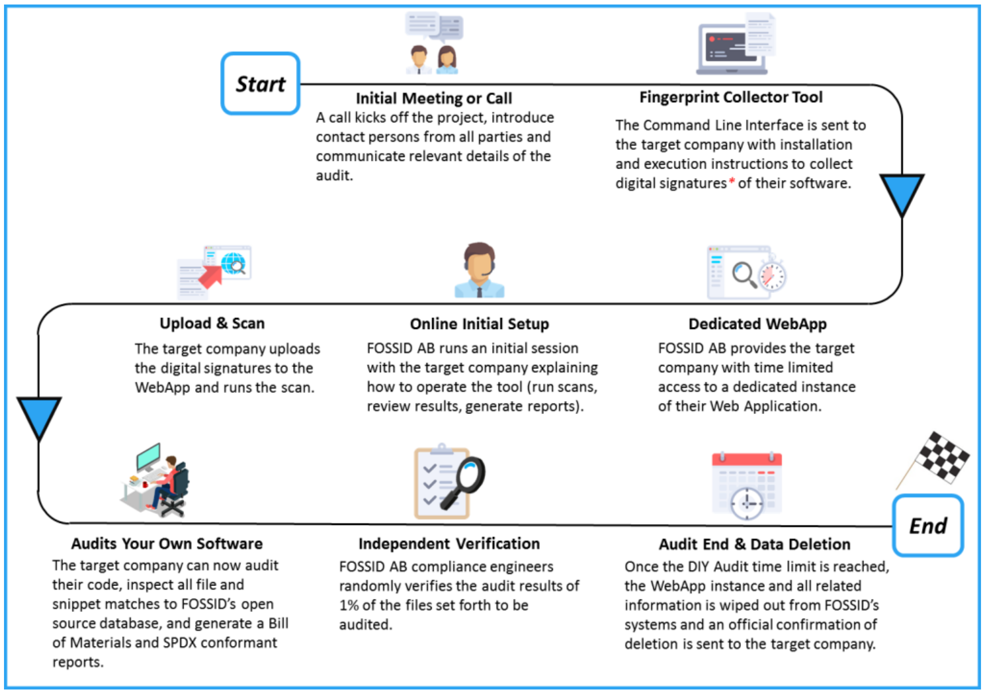
Start
Initial Meeting or Call
A call kicks off the project, introduce contact persons from all parties and communicate relevant details of the audit.
Fingerprint Collector Tool
The Command Line Interface is sent to the target company with installation and execution instructions to collect digital signatures* of their software.
Dedicated WebApp
FOSSID AB provides the target company with time limited access to a dedicated instance of their Web Application.
Online Initial Setup
FOSSID AB runs an initial session with the target company explaining how to operate the tool (run scans, review results, generate reports).
Upload & Scan
The target company uploads the digital signatures to the WebApp and runs the scan.
Audits Your Own Software
The target company can now audit their code, inspect all file and snippet matches to FOSSID's open source database, and generate a Bill of Materials and SPDX conformant reports.
Independent Verification
FOSSID AB compliance engineers randomly verifies the audit results of 1% of the files set forth to be audited.
Audit End & Data Deletion
Once the DIY Audit time limit is reached, the WebApp instance and all related information is wiped out from FOSSID's systems and an official confirmation of deletion is sent to the target company.
End

Figure 8: Illustration of a DIY audit process using FOSSID targeted for M&A transactions

Figure 8 provides an illustration of this audit method using the tools from FOSSID AB. This approach has several advantages such as the ability to start the audit as soon as needed since it uses internal resources and not dependent on the availability of 3rd party auditors, potentially shortening the timelines and reducing an external source of cost. Any compliance problem can be addressed immediately, since it is being conducted by the people who have direct access to the code and can apply fixes directly. Finally, the audit can be verified by the provider of the audit tool to ensure correctness and completeness. As part of their DIY offering, FOSSID AB offers the random verification of 1 % of the files set forth to be audited by the target company.

6. Note on the final report

Many of the auditing tools can also be tuned so that they highlight potential issues. After viewing the results carefully, you might find most of them to be non-issues. So be prepared for what might appear to be a lot of noise. The noise may come from things such as leftover code that is in the code tree but not used. Therefore, the initial report may be lengthy and unfiltered and you should be prepared to invest time to filter the report to find the real issues.

As for SPDX, since it is mentioned in all three figures (Figures 6, 7 and 8), an SPDX conformant report is usually provided on demand. Therefore, if you would like your audit service provider to provide you such a report, you will need to request it.

7. Security and version control

It is a generally accepted truth that software ages like milk, not wine. Security vulnerabilities are a concern with all code whether it is open source or not. However, in open source projects these vulnerabilities are publicly exposed as well as the process of fixing them. This exposure can happen either before or after the fix is implemented, and outdated open source code could potentially contain vulnerabilities that are actively exploited in the wild. While security and version control are not part of the open source compliance due diligence process, companies providing source code scanning services may also offer a service mapping identified open source components against known open source security vulnerabilities.

8. Pre- and post-acquisition remediation

By this point, the acquiring company should have a clear idea how the target uses and manages open source software, and how successful they've been at satisfying their open source license obligations. The acquirer and target should use this information to negotiate remediation for any open source compliance issues. If any issues are uncovered in the audit, there are a few options for resolving them as a part of the pending transaction. The first option is to simply remove any offending code. If the open source software only augments proprietary code, it may be possible to eliminate it entirely. Another option is to design around the offending component, or re-write any code using cleanroom techniques. If the section of code is truly essential or if it has been previously distributed, the only remaining option is to bring the code into compliance. The cost of each option can be used when determining the valuation of the target. Whatever option is chosen, it's crucial to identify the individuals who participated in incorporating the open source code, and to get them involved in the remediation effort. They might have additional documentation or knowledge that can be useful in resolving any issues.

9. Preparing for an audit as an acquisition target

Passing an open source compliance audit is not hard if you're prepared. However, it is very unlikely to happen if you only begin preparing when an acquirer shows interest. These activities are meant to go hand-in-hand with your daily business and development activities. The objective of these activities is to ensure the company tracks all open source components, and respects open source license obligations resulting from your use of these open source components. These same measures can be of great help if your company becomes a target for a corporate transaction, as it minimizes the risk of surprises.

9.1 Know what's in your code

This is the golden rule of compliance. You need to maintain a complete software inventory for all software components including with their origin and license information. This covers software components created by your organization, open source components, and components originating from third parties. The most important point is having a process for

コメントの追加 **[tani8]:** Typo?

identifying and tracking open source components. You don't always need a complex compliance program, however you should have five basic elements: policy, process, staff, training, and tools.

9.1.1 Policy and process

The open source compliance policy is a set of rules that govern the management of open source software (both use of and contribution to). Processes are detailed specifications as to how a company will implement these rules on a daily basis. Compliance policies and processes govern various aspects of using, contributing, auditing, and distribution of open source software.

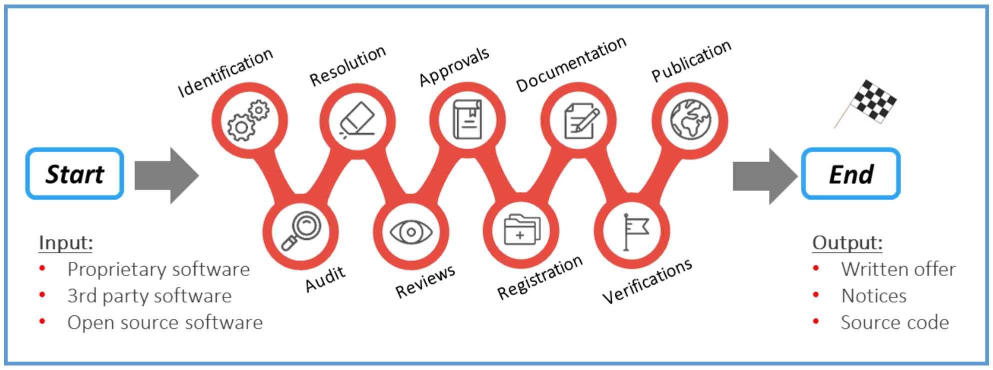

Figure 9: Sample end-to-end open source compliance process

Figure 9 illustrates a sample compliance process, with the various steps each software component will go through as part of the due diligence as you build your product or software stack.

1. Identify all incoming source code
2. Audit source code
3. Resolve any issues uncovered by the audit
4. Complete appropriate reviews
5. Receive approval to use open source
6. Register open source in the software inventory
7. Update product documentation to reflect open source usage
8. Perform verification to all steps previous to distribution
9. Distribute source code and perform final verifications in relation to distribution

The output of the process is an open source BoM that you can publish, along with a written offer and various copyright, license and attributions notices fulfilling the legal obligations of the components in your BoM. For a detailed discussion on the open source compliance process, please download the free e-book "Open Source Compliance in the Enterprise", published by the Linux Foundation.

9.1.2 Staff

In large enterprises, the open source compliance team is a cross-disciplinary group consisting of various individuals tasked with the mission of ensuring open source compliance. The core team, often called the Open Source Review Board (OSRB), consists of representatives from engineering and product teams, one or more legal counsel, and a compliance officer. The extended team consists of various individuals across multiple departments that contribute on an ongoing basis to the compliance efforts: Documentation, Supply Chain, Corporate Development, IT, and Localization. However, in smaller companies or startups, this can be as simple as an engineering manager supported with a legal counsel. Every company is different.

9.1.3 Training

Education is an essential building block in a compliance program, to help ensure that employees possess a good understanding of policies governing the use of open source software. The goal of providing open source and compliance training is to raise awareness of open source policies and strategies, and to build a common understanding of the issues and facts of open source licensing. It should also cover the business and legal risks of incorporating open source software in products and/or software portfolios.

コメントの追加 **[tani9]:** Does this "It" refer to "common understanding"? (There seems to be other options such as "training" or "goal")

9.1.4 Tooling

Open source compliance teams often use tools to automate source code audits, to discover of open source code, and identify its licenses. Such tools include a compliance project management tool, software inventory tool, and source code and license identification tools.

9.2 Be in compliance

If you have shipped products containing open source software, whether intentionally or not, then you will need to comply with the various licenses governing those software components. Hence the importance of knowing what's in your code, as a complete bill of materials makes compliance much easier.

Being in compliance is not a simple task, and it varies from product to product based upon the licenses and the structure of the code. At a high level, being in compliance means that you:

1. Track all use of open source software.
2. Compile a finalized open source BoM for all software in the shipping image of product.
3. Fulfill the obligations of the open source licenses.
4. Repeat the process every time you issue a software update.
5. Respond quickly and seriously to compliance inquiries.

9.3 Use latest releases for security purposes

One of the benefits of a comprehensive compliance program is that it's easier to find products with insecure versions of open source components and replace them. Most source code scanning tools now provide functionality to flag security vulnerabilities disclosed in older software components. One important consideration when upgrading an open source component is to always ensure that the component retains the same license as the previous version. Open source projects have occasionally changed licenses on major releases. To avoid a situation where you are using a version with security vulnerabilities, companies are encouraged to engage with open source project communities. It is not reasonable or feasible to be active in all of the open source projects you use, therefore a certain level of prioritization is needed to identify the most critical components. There are various levels of engagement, ranging from joining mailing lists and participating in the technical discussions, to contributing bug fixes and small features, to major contributions. At minimum, it is very beneficial for corporate developers working on a specific open source project to subscribe to and monitor the mailing list for any reports related to security vulnerabilities, and available fixes.

9.4 Measure up your compliance efforts

The easiest and most effective first step for organizations of all sizes is to engage with the OpenChain Project and to obtain "OpenChain Conformant" status. This is done by filling out a series of questions either online or manually. The questions used for OpenChain Conformance help to confirm that an organization has created processes or policies for open source software compliance. OpenChain is an industry standard, similar to ISO 9001. It is focused on the "big picture," with precise processes and policy implementations up to each individual organization. OpenChain Conformance shows that open source compliance processes or policies exist, and that further details can be shared when requested by a supplier or customer. OpenChain is designed to build trust between organizations across the global supply chain.

The Linux Foundation's Self-Assessment Checklist is an extensive checklist of compliance best practices, in addition to elements that must be available in an open source compliance program to ensure its success. Companies are invited to use this internal, self-administered checklist to evaluate their compliance in comparison to compliance best practices.

コメントの追加 **[tani10]:** Link is missing. But, official LF site corresponding to this, is not reachable. I will ask Nori-san.

9.5 Educate

コメントの追加 **[tani11]:** Contents seem to similar to 9.1.3. What's your intention to describe separately?

The goal of open source and compliance training is to raise awareness of open source policies and strategies, and to build a common understanding of the issues and facts of open source licensing. Training may also cover the business and legal obligations of incorporating open source in products. It also serves as a way to publicize and promote an organization's compliance policies and processes, and to promote a culture of compliance.

コメントの追加 **[tani12]:** I'm curious about what "strategy" means. Which is in below ?
(A) Open Source Strategy
(B) Business Strategy
(C) Both of A and B
(D) Others

There are formal and informal training methods. Formal methods include instructor-led training courses where employees have to pass a knowledge exam to pass the course. Informal methods include webinars, brown bag seminars, and presentations to new hires as part of the new employee orientation session.

10. Preparing for an audit as the acquiring company

As an acquirer, there are actions to take and decisions to make before the audit is commissioned, and then after you receive the results.

10.1 Chose the right audit model and right auditor for your needs

As previously discussed, there are three primary audit methods that can be used and you will need to decide which is most suited to your specific situation, given the parameters you are working with.

10.2 Know what you care about

The report from the source code audit may provide a significant amount of information, depending on the complexity of the scanned code. It is important in to identify which licenses and use-cases are regarded as critical.

10.3 Ask the right questions

The open source audit report offers a lot of information about the target's source code and the licenses involved. However, there are a lot of other data points that will require further investigation to get clarifications or confirmations on compliance related concerns. In this section, we offer a collection of questions as a starting point to frame what is important to you, and what questions you should address with the target company.

- Has the target used code with licenses that could jeopardize the IP of the target or acquirer?
- Are there any code snippets with unknown origin and/or unknown license?
- Are the target's open source compliance practices sufficiently mature and comprehensive?
- Does the target company track known vulnerabilities in their open source components?

- When distributing products, does the target provide all necessary materials to satisfy open source license obligations (written offer, various required notices, and source code when applicable)?
- Does the target company's compliance process aligned with the speed of development to meet product release schedules?
- Does the target have a process in place to respond to all internal and external requests for source code in a timely manner?

10.4 Identify items to be resolved before executing the transaction

In some cases, an open source audit may reveal instances of licenses or compliance practices that are not acceptable to the acquirer. At which point, the acquirer can request these instances to be mitigated as a condition for closing. For instance, the target company may use a code component that comes license under "License A", but the acquiring company has a strict policy against using any source code licensed under "License A". In such a situation, both parties will need to discuss and figure out a possible solution.

コメントの追加 **[tani13]:** What does "which" refer to?

10.5 Create a compliance improvement plan for post-acquisition

This is especially important when the acquirer is a large company buying a smaller startup that will continue to operate as a subsidiary. In this scenario, the acquirer often helps the target establish a formal compliance policy and process, provides training on their own practices, and offers ongoing guidance and support.

11. Recommended compliance-related development practices

Several papers have been written with detailed recommendations for establishing development practices that support open source license compliance activities. In this section, we will briefly highlight the most important practices that when followed, you will eliminate the majority of common compliance issues encountered when working with open source software.

11.1 Recommended practices

- Request approval to use open source software before you commit the code into the product repository.
- Request approval before you link proprietary code to an open source library or vice versa, unless the license of the library code is already pre-approved by company policy.
- Update the changelog for every file you modify to reflect the date of change, the author, and a short one-line description of the change applied.
- Document the interfaces between any code you are writing and open source software, as it helps others understand the interactions and clarify compliance concerns.
- Save the web page describing a source code package's license as a PDF, to document the state of the project when you downloaded it.
- Save an unaltered copy of the package in a backup location, along with the license information.
- When upgrading an open source software component, verify if the license is still the same. License changes can occur between versions.
- Verify that the license in the source code package matches what is described on the project web site. In the event of discrepancy, contact the project for clarification.

11.2 Avoid these mistakes

- Do not remove or disturb existing licensing or copyright information. All such information must remain intact.
- Do not rename open source components.
- Do not copy/paste open source code into proprietary or 3rd party source code (or vice versa) without prior approval.

- Do not commit open source or 3rd party source code into an internal product source tree without prior approval.
- Do not merge or mix source code incoming under different licenses without proper approval.
- Do not discuss compliance practices with individuals outside of your company.

コメントの追加 **[tani14]:** Why is that? Trade secret related issue? I would like to know with some details or examples..

12. Conclusion

Open source due diligence is generally one task in a long list of tasks that need to be successfully completed in an M&A transaction. However, it is still an important aspect of the general due diligence exercise given the central role of software and potential IP risks. While the open source due diligence may seem a lengthy process, it often can be completed quickly, especially if both parties are prepared, and working with a swift compliance service provider.

How can you be prepared?

If you are the target, you can maintain proper open source compliance practices by ensuring your development and business processes include:

- Identifying the origin and license of all internal and external software.
- Tracking open source software within the development process (components and snippets).
- Performing source code reviews for new or updated code entering the build.
- Fulfilling license obligations when a product ships or when software is updated.
- Offering open source compliance training to employees.

If you are the acquirer, you should know what to look for and have the skills on-hand to address issues quickly:

- Decide with the target company on the appropriate audit method to use, and which 3rd party to engage for the audit. Note that some don't have ability to do blind testing, some do not support the DIY, and others do not have the ability to discover code snippets.
- If possible, get multiple bids for the audit and learn more about your audit service providers. It's not just about the cost, but also about having the precise output that will help you address any concerns you may have. Make sure you have the internal expertise to compare each bid equally, and that they include all audit parameters such as:
    - Audit method, inputs and outputs
    - Primary contact persons at target and acquirer for speedy discussions of issues that arise
    - Timeline and logistics especially if it involves an on-site visit
    - Confidentiality parameters
    - Code vulnerabilities and version control analysis
    - Cost, normal process and expedited

コメントの追加 **[tani15]:** I 'm confused a bit. Which do you refer to "primary contact persons" here?

(A) Persons of the bidders (for the target/acquirer)
(B) Persons of the target/acquirer (for bidders)

Because the bidders should clarify the name representing them and corresponding to the audit bit, I think above A will be your answer, but let me confirm.

Open source compliance is an ongoing process, not a destination. Maintaining good open source compliance practices enables companies to be prepared for any scenario where software changes hands, from a possible acquisition, a sale, or product or service release. For this reason, companies are highly encouraged to invest in building and improving upon their open source compliance programs.

###

Resources

Open Source Compliance in the Enterprise

Published by The Linux Foundation, Open Source Compliance in the Enterprise is a practical guide for enterprises on how to best use open source in products and services, and participate in open source communities in a legal and responsible way.

Practical GPL Compliance

Published by The Linux Foundation, Practical GPL Compliance is a compliance guide for startups, small businesses, and engineers, particularly focused on complying with the versions of the GNU General Public License (GPL). Its goal is to provide practical information to quickly address common issues.

OpenChain Curriculum

The OpenChain Curriculum is designed to help organizations meet the training and process requirements of the OpenChain Specification. It can also be used for general open source training and – because of its public domain licensing – can be partially or fully re-used for internal or external purposes without limitation.

Compliance Basics for Developers

A free open source compliance course from the Linux Foundation targeted for developers.

Software Package Data Exchange (SPDX)

SPDX is a set of standard format for communicating the components, licenses and copyrights of software packages.

List of Commercial Providers* of Open Source Compliance Solutions

- Black Duck Software
- Flexera Software
- FOSSA
- FOSSID AB
- nexB
- Protecode (Synopsys)
- Rogue Wave Software
- WhiteSource Software

Open Source Compliance Tools*

- FOSSology is an open source license compliance software system and toolkit.
- Binary Analysis Tool is an open source tool that assist in compliance activities. It examines binary code looking for compliance issues.

* *The author apologized in advance if these lists are missing any provider or tool. If so, please email info@linuxfoundation.org to be added to the appropriate list.*

Acknowledgments

[OpenChain project](), for the permission to use material and re-worked graphics presented in Section 2.

Brian Warner (Senior Manager, Open Source Strategy and Engineering, Samsung), for his editing efforts and proof reading.

Phil Odence (VP &GM, Black Duck Software) for being available to discuss the M&A audit process.

Jon Aldama (VP of Product, FOSSID AB) for being available to discuss the blind and DIY audit process.

Jose L. Lopez (Senior Corporate Counsel, SamsungNEXT), David Marr (VP of Legal, Qualcomm), Nithya Ruff (Senior Director of Open Source Practice, Comcast), and Gil Yehuda (Senior Director of Open Source, Oath) for their reviews and valuable feedback that helped improve the paper and highlight some of the details that would have been otherwise missing.

Shane Coughlan (Program Manager, OpenChain), for his review and ensuring OpenChain's mentions are accurate.

コメントの追加 **[tani16]:** He transformed to Program Director, if my memory is right. ☺

コメントの追加 **[tani17]:** Let me confirm. Does this sentence mean like this , right?
- Shane reviewed this paper
- And he checked descriptions about OpenChain in this paper ,
- And ensured the descriptions were correct.

About the author

Ibrahim Haddad (Ph.D.) is Vice President of R&D and the Head of the Open Source Group at Samsung Research America. He is currently serving as Vice President of the Open Connectivity Foundation and the Director on the Board representing Samsung Electronics. He is the author of “Open Source Compliance in the Enterprise”, a practical guide for enterprises on how to best use open source in products and services, and participate in open source communities in a legal and responsible way.

コメントの追加 **[tani18]:** I omit this section as talked other day.

Web: http://www.ibrahimatlinux.com/
Twitter: @IbrahimAtLinux