**M&A取引での  
オープンソース監査  
（OPEN SOURCE AUDITS IN MERGER AND ACQUISITION TRANSACTIONS）**

必須となるその基礎知識  
（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.*

*本書では、企業の吸収合併（Ｍ＆Ａ）取引におけるオープンソース 監査の全体像および実践的ガイドについて触れていきます。また、買収元、買収先両社における準備としてのオープンソース コンプライアンスを改善していく上で必要な、基礎的ガイドラインについても説明します。*

1. **はじめに**（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.

私たちはソフトウェアによって「定義された」時代（Era defined by software）に生きています。日々私たちがしていることのすべては実質的に、何らかの方法で、ソフトウェアによって計画、具体化、分析され、そして管理されています。その大きなソフトウェアという傘の下でもオープンソース ソフトウェアは、最も重要なものでしょう。すべての産業で企業はオープンソース ソフトウェアからの恩恵を求め、それを使い、そこに参加し、そこへコントリビュートすることを競っています。その恩恵とは、Time to Marketを加速する社外エンジニアリング リソースの活用力に始まり、イノベーションの加速、価値差異化への集中など多岐に亘っています

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.

「オープンソースが ソフトウェアの世界を食べている（Open Source is Eating the Software World）」といわれる話は、企業の取引にも当てはまります。テクノロジー企業の買収はどんなものであれ、何らかの形でソフトウェアに関係してくるからです。買収元が買収対象のソフトウェアやコンプライアンスプロセスに対し包括的レビューを行う、ソフトウェア デューデリジェンス（適正評価）のプロセスは、吸収合併において標準的なものになってきています。こういったプロセスがプロプライエタリ ソフトウェアとは異なる検証課題のあるオープンソース ソフトウェアに及ぶことが一般的になってきているのです。

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).

こういったことを踏まえ本書では、M&A取引におけるオープンソース ソフトウェアの監査について触れていきます。このプロセスを実施していくなかで企業が得られる（技術面、法令面、ビジネス面といった）各種洞察については、本書に続く形で公開できればと思います。

2. **共通的なオープンソース使用シナリオ**（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.

オープンソースのデューデリジェンスの話に入る前に、買収先の開発プロセスでオープンソース ソフトウェアが組み込まれうるさまざまな方法を理解することが助けとなります。このことは、企業が意識的に、もしくは無意識に自社ソースコード ベースにオープンソース ソフトウェアを組み入れるシチュエーションが当てはまります。交通違反キップを切られ、自らの義務を知らなかったといっても言い訳にはなりません。これと同じように複数のソースからソフトウェアが使用されうるさまざまなやり方を理解しておくことは賢明なことなのです。

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.

オープンソースのコンポーネントに変更を加えることや、オープンソースのコードをサード パーティのプロプライエタリコードに注入（Inject）することは、監査サービス プロバイダのコード発見・報告手法に影響してくる可能性があります。オープンソースの監査実施者と関わる際に、彼らがどうやってオープンソースのコードを発見し、捕えるか、そのアプローチを理解することはしばしばと助けとなってくれるのです。

2.1 取り込む（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.

開発者がソフトウェア製品のコードベースの中にオープンソース コンポーネント全部もしくは部分的なコピーを使用する場合があります。オープンソース ライセンスは企業の法的責任や、自社コードの商用的特徴に影響しうるさまざまな義務を伴いますので、こういった取り込み（Incorporation）は、追跡され、宣言され、社内で承認されるべきものと言えるでしょう。



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

図1：オープンソース コード（緑色ドット）の別コード体系（青色）への取り込み（Incorporation）

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.

ソースコード監査は宣言されていないオープンソースのコードベースへの取り込みを発見し、買収後の喜ばしくないサプライズを回避するために設計されます。宣言されない取り込みは、買収先がオープンソース コンプライアンスについて開発者のトレーニングを十分に実施してこなかったり、外注先やインターンといった長期的記録管理を行わない期間労働者に依存しつづけてきたりする場合に発生する可能性が高くなります。

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.

人間の目でソースコードを見た場合、この組み入れがはっきりとわからないことが多いのですが、スニペット（ソフトウェアの断片）を発見し、突合させる機能のあるソースコード スキャンツールによって組み入れを容易に明らかにできる場合があります。

2.2 リンクする（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.

リンク（Linking）は、たとえばオープンソースのライブラリを使用するときなどで、非常によくあるシナリオの一つです。このシナリオでは、開発者はオープンソース ソフトウェアのコンポーネントと自社ソフトウェアコンポーネントをリンクさせています（図2）。このシナリオに対応する用語はいくつかあり、たとえば静的リンク（Static link）、動的リンク（Dynamic link）、結合（Combining）、パッケージング（Packaging）、相互依存性の生成（Creating interdependency）といったものがあります。ライブラリがファイルの最初でインクルードされることや、リンクされるコードが別のディレクトリやファイルにあることからソースコードの目視確認などでリンクは発見され、検出は一般的には容易なものとなります。



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

図2：オープンソース コード（緑色ドット）の別のコード体系（青色）へのリンク（Linking）

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).

リンクが取り込み（Incorporation）と異なるのは、結合し一体化した形態にコピーされるものではなく、ソースコードが分離されている点にあります。リンクの相互作用は、コードが一つの実行バイナリにコンパイルされるとき（静的リンク）、主プログラムが実行され、リンクされたプログラムを呼び出すとき（動的リンク）のいずれかの場合に生じます。

2.3 改変（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)

改変は、開発者がオープンソース ソフトウェアのコンポーネントに変更を加える非常に一般的なシナリオで（図3）、以下のようなものがあります：

* オープンソース ソフトウェア コンポーネントへ新しいコードを追加（Adding）/注入（Injecting）する
* オープンソース ソフトウェア コンポーネントに対し問題解決し（Fixing）、最適化し（Optimizing）、変更を加える（Making change）
* コードの削除（Deleting）または除去（Removing）



図3：オープンソースのコードへの開発者によって適用された改変（緑色・ドット）

2.4 開発ツールについての注記（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.

特定の開発ツールでこういった作業を透過的に実施してくれるものがあることを知っておくことは重要です。たとえば開発者は、開発プロセスのどこか一部を自動的に行ってくれる開発ツールを使うことがあります。こういったものには、ユーザー インターフェースのテンプレートを提供するグラフィックスのフレームワークや、物理エンジンを提供するゲーム開発用のプラットフォーム、もしくはクラウド サービスへのコネクタを提供する開発環境キット（SDKs：Software Development Kit）などがあります。開発者の作成物がビルドされるときには、これらの処理を提供するため開発ツールのコードの一部を注入しなければなりません。生み出された作成物がしばしば静的リンクされるということを特に考慮すると、このように開発ツールによって注入されたコードのライセンスを検証する必要があります。

3. オープンソース監査（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.

個々のM&A取引はいずれも異なるものではありますが、オープンソースの義務をも買収によって獲得するという点でそのインパクトを検証する必要性は、すべての取引について普遍的なことといえます。オープンソース監査はオープンソースの使用の深さと依存度について理解するために実行されます。これに加えて網羅的なコンプライアンスの観点、さらには買収先におけるエンジニアリングの実務についてすばらしい洞察を与えてくれるものとなります。

3.1 なぜオープンソース監査を行うのか？（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.

オープンソース ライセンスはソフトウェアの再頒布方法に制約を課すことがあります。こういった制約は買収元の企業のビジネスと相反するかもしれないので、早期に発見されるべきでしょう。オープンソース ソフトウェアがあることで買収先アセットに影響しうる例として、次のようなものがあります。

* オープンソース ライセンスは一般的に、コード頒布の際に何らかの義務を課すことがあります。一つの例がGNU General Public License (GNU GPL)で、同じライセンスの下で派生物もしくは結合物を公開させるものがあります。その他にもドキュメント内での通知、告知などを求めたり、製品の販売促進のやり方に制約を課したりものもあります
* オープンソース ライセンスの義務の不履行が、訴訟、費用のかさむ再設計、製品リコールや風評被害といったことにつながる可能性もあります

3.2 オープンソース監査を委託すべきか？（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.

一つの共通的な質問として、そもそもオープンソース監査が必要なのか、という話があります。その質問への答えは企業や目的、ソースコードのサイズによって異なります。たとえば、小規模な買収であれば、買収対象の企業から（それがあるとして）提供されるオープンソースの部品表（Bill of Material, BoM）をレビューするだけの対応を好む企業もあれば、エンジアリング リーダーと共にオープンソース実務について議論を実施する企業もいます。買収の目的がたとえ人材の獲得にあったとしても、監査をすることは、出荷済み製品の過去の経緯として残っているライセンス義務として、明確にされていないような責任があるのかどうかを明らかにすることを助けてくれるのです。

3.3 インプットとアウトプット（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:

* 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.

監査プロセスでは主となるインプットが一つ、主となるアウトプット一つあります（図5）。プロセスのインプットは、買収取引にさらされるソフトウェア スタック全体となります。ここにはプロプライエタリ、オープンソース、そしてサード パーティソフトウェアがあります。プロセスの反対側、つまり主となるアウトプットは、詳細にわたるオープンソースの部品表で、以下がリストされたものとなります。

* コンポーネントとして使用されているすべてのオープンソース ソフトウェア、起源やライセンス
* プロプライエタリもしくはサードパーティにあるすべてのオープンソースのスニペット、その起源となるコンポーネントおよび確認されたライセンス



図5

4. 監査業務のスコープを評価する（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.

監査のサイズ、スコープそしてコストは取引ごとに変わるもので、一般的にはソースコードのサイズと複雑さとともに増加します。オープンソース監査に対する（コストと時間の）見積もりを出すためには、監査人はサイズやコードベースの特徴、そのプロジェクトとの緊急性についての基本的な理解をもつ必要があります。

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.

監査人から挙がるであろう最初の質問は、ソースコードベースのメトリクスに関するものでしょう。たとえば、監査対象のコードベースのサイズ、ソースコードのライン数、ファイルの数などです。また、彼らとしては、コードベースがソースコードだけの場合はどうなのか、一方でバイナリファイルやコンフィグレーションファイル、ドキュメント、その他のファイルフォーマットのものを含んでいる場合はどうなのか、ということを質問するでしょう。監査対象のファイルの拡張子を知ることはときとして、監査人にとって有益なものでもあります。

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.

成熟した企業では通常自社プロダクトやプロジェクトで使われているオープンソース コンポーネント、バージョンについて記録を残していきます。こういった情報は非常に有益で、監査人が見込むワークロードについての理解を向上させてくれます。

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.

その理由は、監査価格の議論がサイズや範囲に基づくものとしてプロセスの中で早期に起こるからです。買収元は前述のような情報にアクセスすることができないかもしれません。少なくとも監査人は、追加情報が見積もりの精度を上げてくれることはありますが、まずはスキャンするファイルの数を作業開始の前に理解しておく必要があります。監査人が作業範囲（Scope of Work）を理解する上で十分な情報が得られれば、彼らは緊急性の理解も必要とするでしょう。それが監査のコストに著しいインパクトを与えることになるからです。

5. 監査手法（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.

オープンソース監査を実施する際に活用するツールには買収先にとって有意義な価値を提供する機能があります。その中で最も重要な機能は、買収対象企業のプロプライエタリ コードに混入（もしくはその逆）してしまった、オープンソースコードのスニペットを検索する機能です。また、検知した結果に対する誤検知（False positive）を自動的に削除してくれる機能もあります、これによって、手作業を最小にすることができます。

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 audit or 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.

監査の手法として3つを挙げます。

1. 伝統的な監査。監査人がすべてのコードへの完全なアクセスをもっていて、実地もしくはリモートで監査を実施します。
2. ブラインド監査（見えない状態の、手探りの監査）。監査人はソースコードを見ることなく、リモートで作業を行います
3. DIY（Do It Yourself）監査。買収先企業もしくは買収元が自分自身で大半の監査作業を実際に行います。この監査でツール、監査企業のサポートを得られることがあり、監査企業からの無作為的な検証結果などが提供されることがあります

5.1 伝統的な監査手法（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.

著者がこの手法を「伝統的（Traditional）」としたのは、これがオープンソース コンプライアンスを目的としたソースコードスキャンのもともとの手法だったためです。伝統的な監査では、サードパーティの監査企業の監査人がソースコードにクラウドシステム経由でリモートからアクセスしたり物理的に現地へ足を運んだりしてソースコードを実施します。



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



図6：M&A取引における伝統的な監査手法

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.>

* 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.

図6では、次に挙げる伝統的な監査手法のプロセスを示しています。このプロセスは、サービスプロバイダごとに微妙に異なってくるので留意してください。伝統的な監査プロセスの典型的なものとして以下のようなステップがあります。

* 監査人が、作業内容をよりよく理解するために買収先に質問状を送付する
* 買収先は、監査人が監査スコープとパラメータをよりよく理解できるようこれに答える
* 監査人が、この応答をもとに見積もりを提供する
* 見積もりについて合意され、サービス契約書、作業明細書、守秘義務契約書（NDA）などにサインされる

＜注：図６、７、８にある「**開始**」は合意文書すべてにサインされた後の実際の監査プロセスを想定しています。＞

* 監査人が、対象企業のコードをクラウドへのセキュアなアップロード、もしくは実地訪問により査定する
* 監査人が、対象企業のソースコードをスキャンし、誤検知分を処理し、結果を評価する
* 監査人が、レポートを生成し依頼主に送付する
* 電話会議、もしくはフェイス ツー フェイスのミーティングによって、結果をレビューし、質疑のやり取りを実施する

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.

この手法は、ほとんどの監査サービスプロバイダで一般的なものです。そのため、同じ監査業務に対しては、複数企業からの入札集め、要求に合った最良の入札者を選択することもできます。このモデルであれば買収対象の企業としてもコードを監査人へ送付したり、監査人が実地に訪れたりすることもいとわないでしょう。

5.2 ブラインド監査（Blind audit）

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

ブラインド監査は、ストックホルムを拠点としたFOSSID AB社[[2]](#footnote-2)によって開発された、M&A取引における守秘義務要求に対応した手法です。



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



図 7: M&A取引を想定したFOSSIDを用いたブラインド監査

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 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.

彼らはプロプライエタリな技術を用いて、ソースコードを見ることなく監査を実施し、レポートを生成することを可能にしています。図7に、FOSSID ABを用い、M&A取引におけるソースコードを機密にするようデザインされたブラインド監査のプロセスを例示しました。ブラインド監査の一つの大きなメリットはソースコードへアクセスせずに監査人がレビューを完了できる点にあります。さらに買収元が事前に十分配慮し監査人にも買収先を知ることをさせない、といったハイレベルの機密性を提供することができます。私が認識している範囲では、オープンソース コンプライアンス サービスの提供企業でこういった手法をとれるところは他にはないようです。

5.3 DIY監査（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.

Do-It-Yourself監査は買収元もしくは買収先が自らスキャンが実施できるよう、時間限定でクラウドのコンプライアンス ツールへアクセスできるようにします。ナレッジベースやレポート機能への網羅的なアクセスによって内部監査が実施できるようになります。このアプローチは、スキャン結果を解釈しその改善手続きを提案するための十分な知見がある社員がいる企業にとって特に興味深いものとなります。M&Aプロセスを年間数回は実施するような企業にとってはこの手法はより費用効率を素早く改善することができます。発見事項の検証のため、さらなる監査の完全性確保のために独立した形で認定を行う手法もあります。



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



図 8: M&A取引を想定したFOSSIDを用いるDIY監査プロセス

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.

図8はFOSSID AB社のツールを用いた監査手法を例示しています。このアプローチにはいくつかメリットがあります。社内リソースを使用するのでサードパーティの監査人の対応可能状況に依存せず必要な時にすぐに監査を開始できる点や、将来的な時間の短縮、社外のコスト発生源の削減といったことがあります。コンプライアンスの問題がどんなものでも、直接アクセス・解決できる人によって実施されるので、すぐに取り組むことができます。また最終的にこの監査における正確性や網羅性を確保するため、監査ツール提供者によって検証することもできます。たとえばFOSSIDにおけるDIYサービスとしての一部として、対象企業に監査されるものと示されたファイルの1%に対し無作為検証が提供されています。

6. 最終レポートおける注記（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.

多くの監査ツールは、潜在的に存在する問題へ焦点を当てるようチューニングすることができます。結果を入念にレビューしてみると、多くが課題ではなかったということがあります。こういったノイズとして出てくるものに対しては事前に対応すべきです。ノイズには、コードツリーの中で使われない残存コードのようなものなどがあります。このためレポートでは最初に長くフィルターがかかっていない状態で結果がでてくる場合があるので、そのためこういったものに対し、時間をかけてでも事前に準備して真の問題が発見できるようフィルターを用意しておくべきなのです。

7. セキュリティとバージョン管理（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.

ソフトウェアはワインのようなものでなく、牛乳のように時間の経過ともに劣化するもの、ということは一般的に受け入れられていることです。それがオープンソースかどうかによらず、コードにはセキュリティの脆弱性がついて回ります。しかし、オープンソースのプロジェクトにおいて、こういった脆弱性はその解決プロセスと同じように公衆の面前にさらされるという点では違っています。こういった情報が公開されるのは改修策が実装される前と後、両方のケースがありますが、更新されなくなったようなオープンソースのコードについては、世界中からの攻撃を受けやすい脆弱性を潜在的に含みやすいものでもあるともいえます。セキュリティとバージョン管理は、オープンソース コンプライアンスのデューデリジェンスとしての範疇ではないですが、ソースコード スキャンのサービスを提供する企業として、特定したオープンソースのコンポーネントを既知のオープンソースのセキュリティ脆弱性と照らし合わせてマッピングするようなサービスを併せて提供される場合があります。

8. 買収前、買収後の改善（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.

この段階で買収元企業は、買収対象がどのようにオープンソースを使用し管理しているか、オープンソースのライセンスの義務の履行についてうまくやってきたか、といった点で明確な情報を持っていなければなりません。この情報は、さまざまなコンプライアンス上の問題に対する改善策を両社で協議するために用いられるべきものとなります。監査において問題が明らかになった場合、目下の取引の一部としてそれらを解決するためには選択肢がいくつかあります。最初の選択肢は問題を引き起こすコードすべてを単純に削除することです。そのオープンソース ソフトウェアが単にプロプライエタリのコードの不足を補っているだけというのであれば、完全に削除できるものかもしれません。もう一つの選択肢は、問題となっているコンポーネント周辺をくまなく設計するか、またはクリーンルーム方式（他社の著作権やトレードシークレットを侵すことなく独自開発する手法）ですべてのコードを書き直すことです。それぞれの選択肢で係るコストの値は対象の買収価格を決定する際に適用されうるものとなりえます。いずれを選択するのであっても、オープンソースのコードを組み入れるのに誰が関わったのかを特定し、その人たちに改善対応へ協力してもらうことは非常に大事なことになります。問題を解決するのに有益な文書や知識を彼らが持っている可能性があるからです。

9. 買収対象としての監査準備（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.

オープンソース監査を通過することは、きちんと備えていればそう大変なことではありません。しかし買収先が関心を見せたとき初めてその準備を始める、といったことだとすると話はまったく違ってきます。ここで示す（監査を通るための）活動は日常のビジネスや開発と密接に関係しており、その目的がすべてのオープンソース コンポーネントを追跡し、自分たちが使っているオープンソース コンポーネントから生じるオープンソース ライセンスの義務を尊重することを企業に対し確かなものにしていくことにあるからです。これらと同じ方策は、その企業が企業取引のターゲットになるときに、サプライズのリスクを最小にしてくれる点でも、大きな助けとなってくれます。

9.1 コードの中身を知る（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 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.

コードの中に何があるのかを知ることはコンプライアンスにおける黄金律（Golden rule）です。起源やライセンス情報を含むすべてのソフトウェア コンポーネントに対し、それらを網羅した目録を保持していく必要があります。そこでは、自身の組織で作成されたコンポーネントやオープンソース コンポーネント、そしてサードパーティを起源としたコンポーネントがカバーされています。ここで一番大事なのは、オープンソースのコンポーネントを特定し、追跡していくプロセスをもつことにあります。必ずしも複雑なコンプライアンス プログラムが求められるわけではないのですが、「ポリシー」、「プロセス」、「スタッフ」、「トレーニング」、「ツール」の5つの基本要素は具備しておくべきでしょう。

9.1.1 ポリシーとプロセス（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.

オープンソース コンプライアンス ポリシーは、オープンソース ソフトウェアの管理（使用とコントリビューションの両方）を統制する一連のルールです。プロセスは、企業がこれらのルールを日常ベースで実践していく方法に関する具体的な仕様のことをいいます。コンプライアンス ポリシーとプロセスがオープンソース ソフトウェアの、使用、コントリビューション、監査や頒布といったさまざまな側面で統制していくことになります。



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

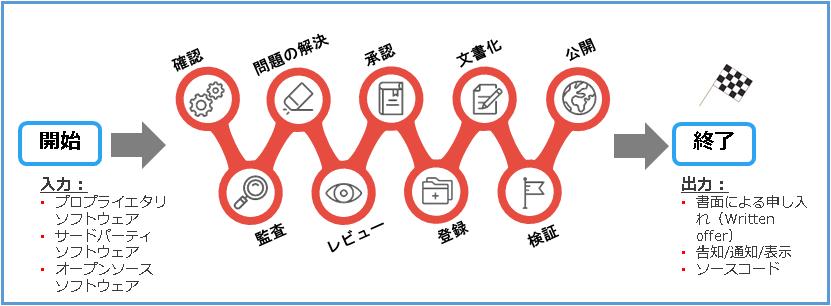


図9：オープンソース コンプライアンス プロセスの開始からから終了まで（サンプル）

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

図9ではサンプルとしてのコンプライアンスを例示しています。企業が製品やソフトウェア スタックを開発する際に、各ソフトウェア コンポーネントはデューデリジェンスの一部としてこれらのさまざまなステップを経ることになります。

1. 外部から入ってくるすべてソースコードを特定する
2. ソースコードを監査する
3. 監査で明らかにされたあらゆる問題を解決する
4. 適切にレビューを実施し、完遂する
5. オープンソースの使用についての内部承認を得る
6. ソフトウェア目録へオープンソースのものを登録する
7. 製品の関連文書にオープンソースの使用状況を反映する
8. 頒布に先立ちすべてのステップに対する検証を行う
9. ソースコードを頒布し、頒布に関連した最終検証を行う

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](https://www.linux.com/publications/open-source-compliance-enterprise) the free e-book “Open Source Compliance in the Enterprise”, published by the Linux Foundation.

このプロセスからのアウトプットは、公開可能なオープンソースの部品表（BoM：Bill of Materials）、書面による申し入れ（Written offer）、著作権、ライセンス、帰属表示など、部品表にあるコンポーネントの法的義務を履行していることを示すものとなります。オープンソース コンプライアンス プロセスの詳細については、The Linux Foundationから公開されているフリーの電子書籍「[Open Source Compliance in the Enterprise](https://www.linux.com/publications/open-source-compliance-enterprise)」を参考にしてみてください。

9.1.2 スタッフ（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.

大企業におけるオープンソース コンプライアンスチームは、オープンソース コンプライアンスを確実にするという目標をもつ、さまざまな個人で構成される分野横断的なグループとなります。中核となるチーム（Core team）はしばしば「オープンソース レビューボード（Open Source Review Board: OSRB）」と呼ばれ、エンジニアリングや製品チームからの代表者と一人、それ以上の法律の専門家やコンプライアンス オフィサーで構成されます。また、ドキュメント、サプライチェーン、経営企画、情報システム、地域対応といった継続的にコンプライアンスの取り組みに貢献する複数の部門に亘るさまざまな個人によって拡張チーム（Extended team）が形成されます。ただ、小規模の企業やスタートアップにおいては、一人のエンジニアリング マネージャと一人の法律専門家というシンプルな構成もありえるでしょう。どういった構成になるかはすべての会社で違ってきます。

9.1.3 トレーニング（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.

教育は、コンプライアンス プログラムで本質的に重要な構成要素となります。教育によって従業員に対しオープンソース ソフトウェアの使用を統制しているポリシーについてきちんとした理解をもつことを促進していくことができます。オープンソースとコンプライアンスのトレーニングを提供することのゴールは、オープンソースに関するポリシーと戦略への理解を底上げし、オープンソースのライセンスについての事実と課題について共通した理解を構築することにあるのです。またプロダクトやソフトウェアのポートフォリオや製品にオープンソース ソフトウェアを組みこむことのビジネス上、法令上のリスクについてもその理解がおよぶべきでしょう。

9.1.4 ツールの活用（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.

オープンソース コンプライアンス チームは、ソースコードの監査の自動化、オープンソース コードの発見、そのライセンスの特定のためにツールを頻繁に用います。これらのツールとしては、コンプライアンス プロジェクト管理のためのもの、ソフトウェア インベントリのためのものやソースコードやライセンスを特定するためものなどが挙げられます。

9.2 「コンプライアンス」の状態にある（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.

オープンソース ソフトウェアを含む製品を出荷した場合、意図的であるかどうかによらず、それらソフトウェア コンポーネントを統制する各種ライセンスを順守している必要があります。こういったことを踏まえるとコードの中身を網羅された部品表（BoM）として知ることに重きを置くことによってコンプライアンスは実施しやすいものになるのです。

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.

コンプライアンスの状態にある、というのはそう単純な話ではなくライセンスやコードの構造に応じて製品ごとに変わってくるものです。ハイレベルでは、コンプライアンスにある状態とは、以下を意味しています；

1. すべてのオープンソース ソフトウェアを追跡している
2. 製品として出荷したイメージファイルにあるソフトウェアに対しオープンソースの部品表（BoM）として取りまとめている
3. オープンソース ライセンスの義務を履行している
4. ソフトウェアのアップデートを発行する毎に同じプロセス繰り返されている
5. コンプライアンスに関すると問い合わせに対し真摯にかつ迅速に対応している

9.3 セキュリティのために最新版を使用する（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.

包括的なコンプライアンスプログラムの一つのメリットとして、オープンソース ソフトウェアで安全ではないバージョンを見つけ、それを置き換えやすくできることがあります。今は大抵のソースコード スキャンツールが古いソフトウェア コンポーネントで明らかになったセキュリティ脆弱性に対しフラグを付ける機能を提供しています。オープンソース コンポーネントをアップグレードする際に考えるべき大事なことの一つは、前のバージョンと同じライセンスが維持されていることを常に確認することです。オープンソース プロジェクトでは、メジャーリリースのタイミングでライセンスが変更されることがあります。セキュリティ上の問題があるバージョンを使う状況を回避するには、企業がオープンソース プロジェクトのコミュニティに積極的に関わって（エンゲージして）いくことが望まれます。使用しているオープンソース プロジェクトすべてでアクティブに活動することは合理的、現実的ではないので、もっとも重要なコンポーネントを特定した上で一定レベルの優先度付けを行うことが必要となります。エンゲージメントのレベルは多岐に亘り、メーリング リストへの登録、技術的議論への参加や、バグ修正や小さめの機能へのコントリビューション、といったレベルから、さらには主要となるコントリビューションまでさまざまです。少なくとも、特定のオープンソース プロジェクトに取り組んでいる企業内の開発者がメーリング リストを閲覧し、セキュリティの問題や修正の公開のレポートに目を配っておくことは非常に有益なものとなります。

9.4 コンプライアンスの取り組みを測る（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](https://www.openchainproject.org/conformance)” status. This is done by filling out a series of questions either [online](https://certification.openchainproject.org/) or [manually](https://wiki.linuxfoundation.org/_media/openchain/openchain_conformance_conformance_check_1.1.pdf). 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.

組織の規模によらず、もっとも簡単でかつ効果的な最初のステップはOpenChainプロジェクトに関わり、「[OpenChain適合](https://www.openchainproject.org/conformance)（OpenChain Conformant）」のステータスを得ることです。[オンライン](https://certification.openchainproject.org/)もしくは[机上で](https://wiki.linuxfoundation.org/_media/openchain/openchain_conformance_conformance_check_1.1.pdf)一連の質問を埋めることでこの作業は実施できます。OpenChain適合に使われる質問が、組織の作ったオープンソース ソフトウェアのコンプライアンスに対するプロセスやポリシーをチェックする助けにもなります。OpenChainはISO9001に似た産業の標準であり、適切なプロセス、ポリシーが個々の組織までに実装されている全体像を描くことに焦点を当てています。OpenChain適合していることは、オープンソース コンプライアンスのプロセスやポリシーが存在していることを示すものであり、サプライヤや顧客の求めに応じて、詳細情報も提供できることも意味します。OpenChainは、グローバルなサプライチェーンをまたがる組織間の信頼を築き上げるために設計されているのです。

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.

The Linux Foundationの「[自己査定用チェックリスト（Self-Assessment Checklist）](https://www.linuxfoundation.org/blog/self-assessment-checklist-a-measuring-stick-for-open-compliance-efforts/)」はオープンソースコンプライアンスプログラムがうまくさせる有効要素に加え、コンプライアンスのベストプラクティスについても触れた幅広いチェックリストです。企業はこれを社内で、自分たちのコンプライアンスをベストプラクティスと比較して評価する自己管理のために用いることが奨励されています。

9.5 教育する（Educate）

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.

オープンソースとコンプライアンスに関するトレーニングのゴールは、オープンソース ポリシーと戦略についての意識を高めること、そしてオープンソースのライセンスの問題と実態についての共通的な理解を築きあげることです。トレーニングが、製品にオープンソースを組み入れることに関するビジネス面、法令面での義務にまでおよぶこともあります。また、トレーニングは、組織のコンプライアンス ポリシーやプロセスを社外アピールや宣伝、社内におけるコンプライアンス カルチャー醸成のための手段として機能することもあります。

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.

トレーニングには形式ばったフォーマルな手法と、そうでないインフォーマルな手法があります。形式的なものには、講師主導のトレーニング コースで、従業員がコースを修了するために知識試験を通過しなければなりません。非形式的なものとしては、Webinarや弁当持参のセミナー（Brown bag seminar）、そして新規採用者向けの従業員オリエンテーションの一部に組み込まれたプレゼンテーションなどが挙げられます。

10. 買収企業として監査に備える（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.

監査を依頼する前の段階および監査結果を受領した後の段階で、買収する側の企業としてするべき意思決定やとるべきアクションがあります。

10.1 ニーズに合わせ適切な監査モデル・監査人を選択する（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.

前に挙げたように、使われうる主だった監査の手法には3つあり、自社の業務のパラメータを考えながら、どれが自社の具体的な状況と合っているのか、ということは決める必要があります。

10.2 何に留意をすべきかを知る（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.

コードの複雑さによっては、ソースコード監査レポートが膨大の量の情報を提供することがあります。ここでは、ライセンスやユースケースのどれを重要なものとみなすべきかを明確にすることが重要になってきます。

10.3 適切な質問をする（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?

オープンソース監査レポートは、買収対象のソースコードと関連するライセンスについての大量の情報を提示することになります。しかし、コンプライアンスに関連する明確な説明や立証を得る上では、更なる調査が要求されるようなデータがたくさんあるのです。本節では、自社にとって必要なことが何か、買収先と取り組むべき問題は何か、といったことを枠組みとしてとらえるためのスタート地点として以下の質問集を挙げてみます。

* 買収元や買収対象の知財を危険にさらすようなライセンスのコードを買収対象が使っていないか？
* よくわからない起源、ライセンスのコードのスニペットはないか？
* 買収対象のオープンソース コンプライアンスの実務は、十分にもまれていて、包括的なものであるか？
* 買収対象は、自身が使っているオープンソース コンポーネントの既知の脆弱性を追跡しているか？
* 製品が流通しているとき、買収対象の企業がオープンソース ライセンスの義務を履行するためにすべての必要な資料（書面での申し入れ、必要な告知/通知/表示、ソースコードなど当てはまるもの）を提供しているか？
* 買収対象企業のコンプライアンス プロセスが、製品リリース計画に基づく開発スピードと合ったものになっているか？
* 買収対象に、社内外のソースコードを求める要請すべてに、タイムリーな形で対応することができるプロセスがあるか？

10.4 取引実行前の段階で解決すべき項目を特定する（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.

取引にもよりますが、オープンソース監査がライセンスやコンプライアンスの実務が、買収元にとって受け入れられない実例が明らかになる場合があります。その段階で買収元は、そういった実例を軽減することを契約締結の条件として要求することが可能です。たとえば、買収対象の企業が、「Aライセンス」というライセンス下にあるコンポーネントのコードを使っていて、一方で買収元の企業が「Aライセンス」の使用を禁止する、厳格なポリシーを持っているようなケースです。こういった状況は、両社が議論して、可能性のある解決策を導き出すことが必要になってきます。

10.5 買収後のコンプライアンス改善計画を策定する（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.

大企業が小さいスタートアップを買収し、子会社として運営し続けるようなときこういった話が大事になってきます。このシナリオでは買収元はしばしば買収先に対し、秩序だったコンプライアンス ポリシーやプロセスを確立することを支援し、実務者たちに向けたトレーニングを実施し、そして継続的指導・支援を提供します。

11.コンプライアンス関連で推奨される開発の実務（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.

オープンソース ライセンスのコンプライアンス活動を支援する開発業務の実務を確立する上での推奨事項の詳細については、いくつかの文書が執筆されています。この節では、その中でも最も重要となるものに簡単に焦点を当てます。それら従うことでオープンソース ソフトウェアとともに仕事をしていく時に遭遇する共通的なコンプライアンス問題の多くを排除することが期待できます。

11.1 実務として推奨されること（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.
* 製品のレポジトリにコードをコミットする前に、オープンソース ソフトウェアを使用するための承認を求める
* あるオープンソース ライブラリのコードのライセンスが企業のポリシーとして事前に承認されていない場合、プロプライエタリ コードをライブラリにリンクする、もしくはその逆のことをする前の段階で承認を求める
* 適用された変更について、変更日時、変更者、一行程度の内容説明といった変更ログ（Changelog）をすべてのファイルについてアップデートする
* 作成しているすべてのコードとオープンソース ソフトウェアの間のインターフェースを文書化する。これにより他社が、（ソフトウェア間の）相互作用を理解し、コンプライアンス上の関係性を明確にすることを促進する
* ソースコード パッケージのライセンスが記載されているWebページをPDFで保管する。これによりパッケージをダウンロードした際のプロジェクトの状態を文書化することができる
* パッケージの変更されていない状態でのコピーをライセンス情報と合わせてバックアップしておく
* オープンソース ソフトウェアのコンポーネントをアップグレードする際に、ライセンスが同じものかどうかを確認する。ライセンスはバージョン間で変わることがある
* ソースコード パッケージにあるライセンスがプロジェクトWebサイトで記述されているものと合っているかどうかを確認する。差異がある場合、プロジェクトにコンタクトし明確にする.

11.2 間違いを回避する（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.
* すでにあるライセンス情報・著作権情報の削除や配置換えをしないこと。こういった情報はすべて完全な状態にしておくこと
* オープンソース コンポーネントの名称を変更しないこと
* 事前承認なくオープンソース コードをプロプライエタリ、もしくはサードパーティのコードにコピー/ペーストしないこと（その逆もしかり）
* 事前承認なくオープンソース、もしくはサード パーティのソースコードを製品のソース ツリーにコミットしないこと
* 事前承認なく外部からのライセンスが異なるコードを混合させないこと
* 社外の人々と、コンプライアンスの実務について議論しないこと

12. 結論（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.

オープンソースのデューデリジェンスは一般的に、M&A取引を成功裏に完了させるために必要な長いタスクリストの中の一つにすぎません。しかし、そうはいってもソフトウェアの中核的な役割と潜在的な知財リスクを考えると、全面的にデューデリジェンスを実践していくのは重要なことです。オープンソースのデューデリジェンスは、長期にわたるプロセスに感じられるかもしれませんが、特に両社が準備をしていてかつ対応の早いコンプライアンス サービスプロバイダと一緒に取り組む場合は、しばしば迅速にこれを完了させることができるのです。

How can you be prepared?

どうやれば準備しておくことができるのでしょうか？

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

買収対象であれば、開発プロセスとビジネス プロセスが以下のような点に確実に対応することで、適切なオープンソース コンプライアンスの実務を維持することができます。

* 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.
* 社内、社外のソフトウェアすべてについて起源とライセンスを明確にする
* 開発プロセスの中でオープンソースのコンポーネント、スニペットを追跡する
* ビルドの中で新規採用、アップデートされるコードに対しソースコード レビューを実施する
* 製品が出荷されるとき、ソフトウェアがアップデートされるときにライセンスの義務を履行する
* 従業員に対しオープンソース コンプライアンスのトレーニングを提供する

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

買収元であれば、探すべきものを知り、迅速に問題に取り組むためのスキルを手近にもっておくことが必要でしょう。

* 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
* 買収対象の企業とともに、用いるべき妥当な監査手法とその監査に関与するサードパーティの企業を決定する。ただし、ブラインド テストを実施できないところもあれば、DIY監査に対応していないところもあり、またスニペットまでは発見できないところもあるので留意が必要となる
* 可能であれば、監査を複数社からの入札とし、監査サービス プロバイダについて学ぶ。単純に費用の話だけではなく、買収元としての取り組みを支援する、きちんとしたアウトプットを出せるかどうかも重要となる。各社の入札を公平に比較するための社内専門家を確保した上で、各社の入札内容が以下監査パラメータすべてを含んでいることを確認する
  + 監査手法、インプットとアウトプット
  + 問題が生じた際の議論の迅速化のための買収元と買収対象に対する一次窓口担当者
  + スケジュールと物流（実地訪問が絡む場合は特に）
  + 機密事項に関するパラメータ
  + コードの脆弱性とバージョン コントロールに関する分析
  + 費用、通常プロセスと簡易版プロセス

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.

オープンソース コンプライアンスは継続して行われるプロセスであり、到着地点ではありません。よいオープンソース コンプライアンスの実務を維持していくことで、企業は、起こりうる買収、売却、はたまた製品・サービスのリリースといったソフトウェアの持ち主が変わる、あらゆるシナリオに対し用意ができるようになります。こういった理由からオープンソース コンプライアンス プログラムの構築、改善へ投資することが企業に高く推奨されているのです。

###

参考文献（Resources）

Open Source Compliance in the Enterprise（企業におけるオープンソース コンプライアンス）

Published by The Linux Foundation, [Open Source Compliance in the Enterprise](https://www.linux.com/publications/open-source-compliance-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.

The Linux Foundationで公開されている「[Open Source Compliance in the Enterprise](https://www.linux.com/publications/open-source-compliance-enterprise)（企業におけるオープンソース コンプライアンス）」は、実践ガイドとして企業が製品やサービスにオープンソースをうまく活用し、法的責任を果たす形でオープンソース コミュニティに参加するする方法について触れています。

Practical GPL Compliance （実践GPLコンプライアンス）

Published by The Linux Foundation, [Practical GPL Compliance](https://www.linuxfoundation.org/news-media/research/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.

The Linux Foundationで公開されている「[Practical GPL Compliance](https://www.linuxfoundation.org/news-media/research/practical-gpl-compliance)（実践GPLコンプライアンス）」は、スタートアップや小規模事業者、そして技術者向けのコンプライアンス ガイドで、特にGNU General Public License (GPL)の各バージョンを順守することに焦点を当てています。

OpenChain Curriculum （OpenChain カリキュラム）

The [OpenChain Curriculum](https://wiki.linuxfoundation.org/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.

「[OpenChain Curriculum](https://wiki.linuxfoundation.org/openchain/curriculum)（OpenChainカリキュラム）は、組織がOpenChain Specification（OpenChain仕様書）のトレーニングやプロセスとしての要求事項へ準拠することを支援するために設計されています。一般的なオープンソースのトレーニングでも使用可能ですし、パブリックドメインのライセンスであることから部分的にでもすべてでも制約なく組織内外で再利用することができます。

Compliance Basics for Developers （開発者向けコンプライアンスの基礎）

A [free open source compliance course](https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/compliance-basics-for-developers) from the Linux Foundation targeted for developers.

The Linux Foundationによる開発者を対象とした[無償のオープンソース コンプライアンスコース](https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/compliance-basics-for-developers)です。

Software Package Data Exchange (SPDX)

[SPDX](https://spdx.org/) is a set of standard format for communicating the components, licenses and copyrights of software packages.

[SPDX](https://spdx.org/)は、ソフトウェア パッケージのコンポーネント、ライセンスおよび著作権を伝達していくための一連の標準フォーマットになります。

オープンソース コンプライアンス ソリューションを提供している商用プロバイダー一覧（List of Commercial Providers\* of Open Source Compliance Solutions ）

* [Black Duck Software](https://www.blackducksoftware.com/)
* [Flexera Software](https://www.flexerasoftware.com/)
* [FOSSA](http://fossa.io/)
* [FOSSID AB](http://www.fossid.com)
* [nexB](https://www.nexb.com/)
* [Protecode](http://www.protecode.com/) (Synopsys)
* [Rogue Wave Software](https://www.roguewave.com/)
* [WhiteSource Software](https://www.whitesourcesoftware.com/)

オープンソース コンプライアンスツール（Open Source Compliance Tools\*）

* [FOSSology](https://www.fossology.org/) is an open source license compliance software system and toolkit.
* [Binary Analysis Tool](http://www.binaryanalysis.org/) is an open source tool that assist in compliance activities. It examines binary code looking for compliance issues.
* [FOSSology](https://www.fossology.org/) はライセンス コンプライアンスのためのオープンソースのソフトウェアシステムとツールキットです。
* [Binary Analysis Tool](http://www.binaryanalysis.org/) は、コンプライアンスのための活動を支援するオープンソースのツールです。バイナリ コードを試験し、コンプライアンスの問題を探索します。

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

\* 事前のお詫び：このリストに記載したプロバイダやツールに不備がありましたらお詫びいたします。またその際にはお手数ですが適切なリストとして追加いたしますので以下に電子メールでご連絡ください（ただし英語で）[*info@linuxfoundation.org*](mailto:info@linuxfoundation.org)

謝辞（Acknowledgments）

[OpenChain project](https://www.openchainproject.org/), 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.

[OpenChanプロジェクト](https://www.openchainproject.org/)へ。2章で紹介した図使用し、改良することを許可してくれたことに対して。

Brian Warner（Samsung、Open Source Strategy and Engineeringシニアマネージャ）へ。編集、構成対応に対して。

Phil Odence（Black Duck Software、VP&GM）へ。M&A監査プロセスについての議論に対して。

Jon Aldama（FOSSID AB、製品対応VP）へ。ブラインド監査、DIY監査プロセスについての議論に対して。

Jose L. Lopez (SamsungNEXT、シニア コーポレート カウンシル)、David Marr (Qualcomm、法令対応VP)、Nithya Ruff (Comcast、Open Source Practiceシニアダイレクタ), and Gil Yehuda (Oath、Open Sourceシニアダイレクタ)へ。レビューや貴重なフィードバックを通じて詳細情報が不足してことを気づかせてくれたこと、本書を改善してくれたことに対して。

Shane Coughlan（Openchain、Program Manager）へ。彼のレビューとOpenChainについての記述が正確であることを確認してくれたことに対して。

著者について（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.

Web: <http://www.ibrahimatlinux.com/>

Twitter: [@IbrahimAtLinux](https://twitter.com/ibrahimatlinux)

Ibrahim Haddad (Ph.D.) はSamsung Research America のR&D部門のVPおよびOpen Source Group（OSG）の長として一翼を担っています。現在、Open Connectivity FoundationのVice PresidentおよびSamsung Electronicsを代表したDirector on Boardの役を務めています。また彼は、企業が製品・サービスでオープンソースを活用し、法的に責任のある形でオープンソース コミュニティに参加するための最前の方法についての実践ガイドである「Open Source Compliance in the Enterprise（企業におけるオープンソース コンプライアンス）」の著者でもあります。

Web: <http://www.ibrahimatlinux.com/>

Twitter: [@IbrahimAtLinux](https://twitter.com/ibrahimatlinux)

1. Please note that FOSSID AB refers to the company and FOSSID refers to the tool. [↑](#footnote-ref-1)
2. FOSSID ABが会社名で、FOSSIDはツールの名前です。 [↑](#footnote-ref-2)