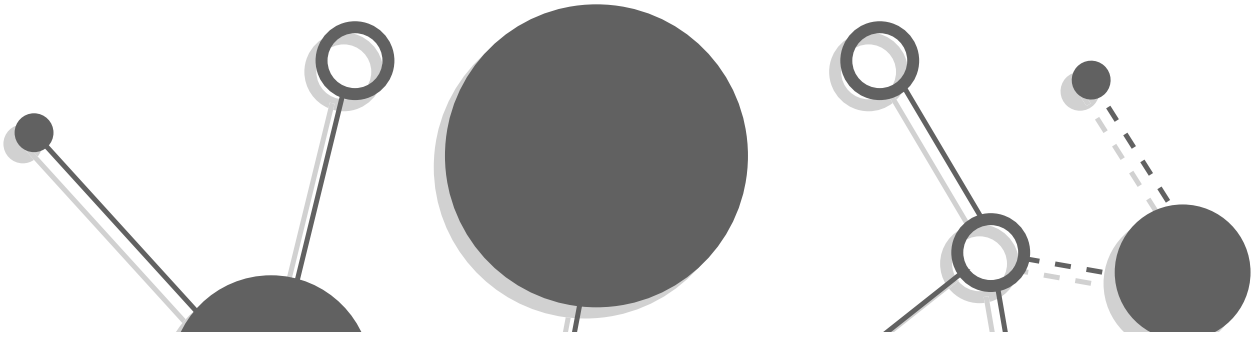
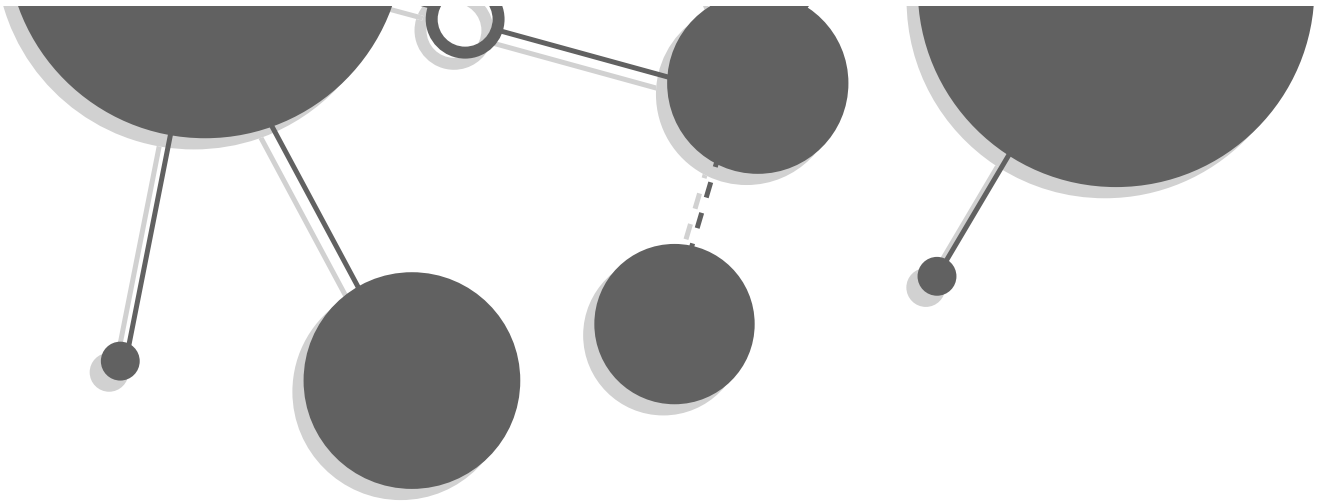


M&A取引における
オープンソース
監査
必須となるその基礎知識



THE LINUX FOUNDATION

イブラヒム ハダド(Ibrahim Haddad), Ph.D.

M&A取引におけるオープンソース監査

必須となるその基礎知識

イブラヒム ハダド(Ibrahim Haddad), Ph.D. 著

本書は、企業の合併・買収(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を提供するものです。また、買収企業、買収対象双方においてオープンソース コンプライアンスへの備えを強化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ていきます。

免責事項:本書は著者の法務専門職とは異なる経歴・知見に基づくものであり、法的アドバイスを提供するものではありません。また、本書は、著者自身の見解を示したものであり、現在もしくは過去の著者所属企業の見解を反映したものではありません。

THE LINUX FOUNDATION

目次

1 はじめに

私たちはソフトウェアによって定義された時代に生きています。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。その恩恵は、社外エンジニアリングリソースを有効活用することによるTime to Marketの短縮から、イノベーションの加速など多岐へ亘っているのです。

こういった考え方は、企業の合併・買収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。買収企業が買収対象企業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、精査）のプロセスは、合併・買収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で出くわすことがごく一般的になってきているオープンソース ソフトウェアには、プロプライエタリ ソフトウェアとは異なる検証課題があるのです。

本書では、企業の合併・買収（M&A）取引におけるオープンソース ソフトウェアの監査プロセスの全体像について触れていきます。

2 共通的なオープンソース使用シナリオ

オープンソースのデューデリジェンスの話に入る前に、買収対象の開発プロセスでオープンソース ソフトウェアが取り込まれうるさまざまな局面を理解することが有益です。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にも当てはまります。交通違反キップを切られた際に、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、複数の提供元のソフトウェアが使用される可能性のあるさまざまな局面を理解しておくことは賢明なことなのです。最もありがちなオープンソース ソフトウェア使用シナリオとして、取り込み(Incorporation)、リンク(Linking)、および、改変(Modification)があります。

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プロプライエタリ コードやサード パーティ 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・報告手法に影響を与える可能性があります。オープンソース監査のサービス提供者と関わる際に、彼らがどうやってオープンソースのコードを発見し、捕捉するか、そのアプローチを理解することがしばしば助けとなるのです。

2.1 取り込む(Incorporation)

開発者がソフトウェア製品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(これはスニペットと言われることもあります)を使用する場合があります。そういったシナリオは許容できるもので、取り込まれるオープンソースのコードのライセンスや、それを取り込むソフトウェア コンポーネントのライセンスに依存したライセンス リスクもないものかもしれません。しかし、コピーされたオープンソースのコードがプロプライエタリのコードベースのライセンスと相入れないような場合など、取り込みが問題を生じることもあるのです(図1)。

オープンソース ライセンスは企業の法的責任や、自社コードのプロプライエタリ性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は、サ

ードパーティのソフトウェアと同様のプロセスの下で追跡され、申告され、社内で承認されるべきものといえるでしょう。

ソースコード監査は、申告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、買収後の好ましくないサプライズを回避するために設計されます。申告されない取り込みは、買収対象がオープンソース 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、コントラクタやインターンといった長期に亘る記録管理を行わない期間作業者に依存し続けてきたりする場合に起こる可能性が高くなります。

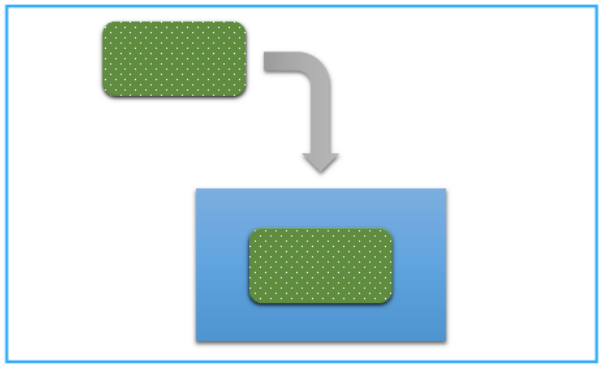


図1:オープンソース コード(緑色ドット)の別コード体系(青色)への取り込み(Incorporation)

人間の目でソースコードを見た場合、この取り込みがはっきりとわからないことが多いのですが、スニペットを発見し、突合せを行う機能のあるソースコード スキャンツールによって取り込みを容易に発見できる場合があります。

2.2 リンクする(Linking)

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、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ます(図2)。このシナリオに対応する用語は静的リンク(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)といったものを含めいくつかあります。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

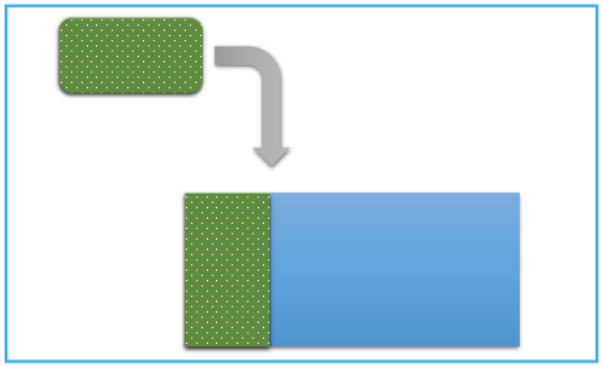


図2:オープンソース コード(緑色ドット)の別のコード体系(青色)へのリンク(Linking)

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。

リンクが取り込み(Incorporation)と異なるのは、ソースコードが一体化した形で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時点(静的リンク)、あるいは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)のいずれかの時点で生じます。

2.3 改変(Modification)

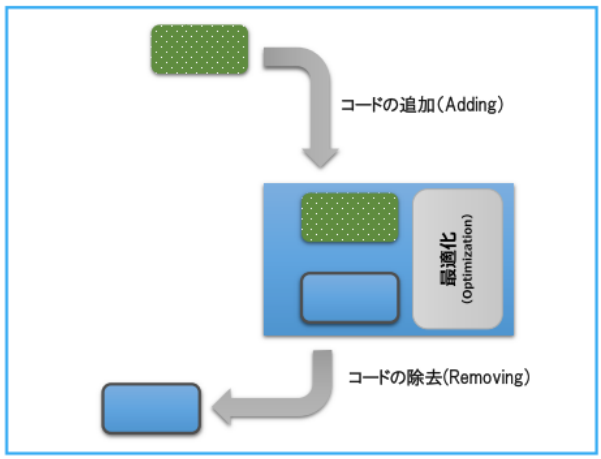


図3:オープンソースのコードへの開発者によって適用された改変(緑色・ドット)

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一般的なシナリオで(図3)あり、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バグ修正(Fixing)、最適化(Optimizing)、あるいは変更(Making change)を実施する
- コードの削除(Deleting)または除去(Removing)

2.4 開発ツールに関する留意事項

開発ツールの中にはこういった作業を開発者に気づかれず実施してしまうものがあることを知っておくことが重要です。たとえば開発者は、開発プロセスの一部を自動的に行ってくれる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ソフトウェア開発キット(SDKs: Software Development Kit)などがあります。こういったツールは、開発者の作成物がビルドされる際、前述の処理を提供するために開発ツールのコードの一部をその作成物に注入しなければなりません。特に、生み出された作成物がしばしば静的リンクされているということを考慮すると、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

個々のM&A取引はいずれも異なるものではありますが、買収に伴ってオープンソースの義務を引き継ぐ、そのインパクトを検証する必要性は普遍的なものといえます。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。さらにオープンソース監査は、コンプライアンス課題や買収対象におけるエンジニアリングの実務について重要な洞察を与えてくれるものでもあります。

3.1 なぜオープンソース監査を行うのか？

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す。オープンソース ソフトウェアがあることで買収対象企業の資産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します。一つの例がGNU General Public License (GNU GPL)で、派生物もしくは結合物を同じライセンスの下で利用できるようにすることを要求します。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するものもあります
- 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、製品リコールや悪評など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？

一つの共通的な疑問として、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。その疑問への答えは企業によって、買収の目的によって、またソースコードのサイズによって異なります。たとえば、小規模な買収の場合、買収対象企業からオープンソースの部品表(Bill of Material, BoM)の提供を受けることができれば、それをレビューし、エンジニアリング 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だけでよしとする企業もあります。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引き

継ぐライセンス義務からくる、明確にされていない責任があるのかどうかを明らかにするのに有益なものとなるのです。

3.3 インプットとアウトプット

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4）。プロセスのインプットは、買収取引に関係するソフトウェア 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。プロセスの出力側、つまり主となるアウトプットは詳細に亘るオープンソースの部品表であり、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、それらの起源およびライセンス
- プロプライエタリもしくはサードパーティ ソフトウェアで使用された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

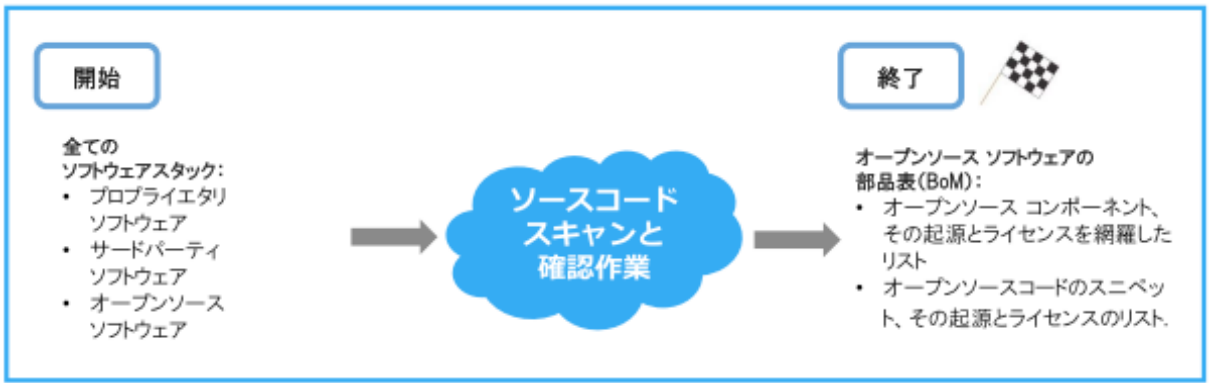


図4：デューデリジェンスプロセスのインプットとアウトプット

4 監査業務のスコープを評価する

監査の規模、スコープそしてコストは合併・買収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、監査人はコードベースのサイズとその特徴、そしてプロジェクトの緊急性について基本的な理解をもつ必要があります。

監査人が挙げ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のか、ということも質問するでしょう。監査対象のファイルの拡張子を知るのはときとして、監査人にとっても有益なことなのです。

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。

監査費用の議論は規模やスコープに基づき監査プロセスの中でも早期に起こるため、買収企業が前述のような情報のすべて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。ただし、追加の情報が見積もりの精度を上げてくれることがあります。監査人が作業スコープ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を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

オープンソース監査を実施する際に活用するツールの機能で買収企業にとって有意義な価値をもたらすものがあります。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、これによって手作業を最小にすることができます。

監査の手法には以下の3つがあります。

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実施し、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。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。買収対象企業もしくは買収企業が自分自身で大半の監査作業を実際に行います。監査企業からは、監査ツール、サポート、さらには、監査結果に対する無作為的な検証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

私がこの手法を「伝統的(Traditional)」と呼ぶのは、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スキャンを実施します。

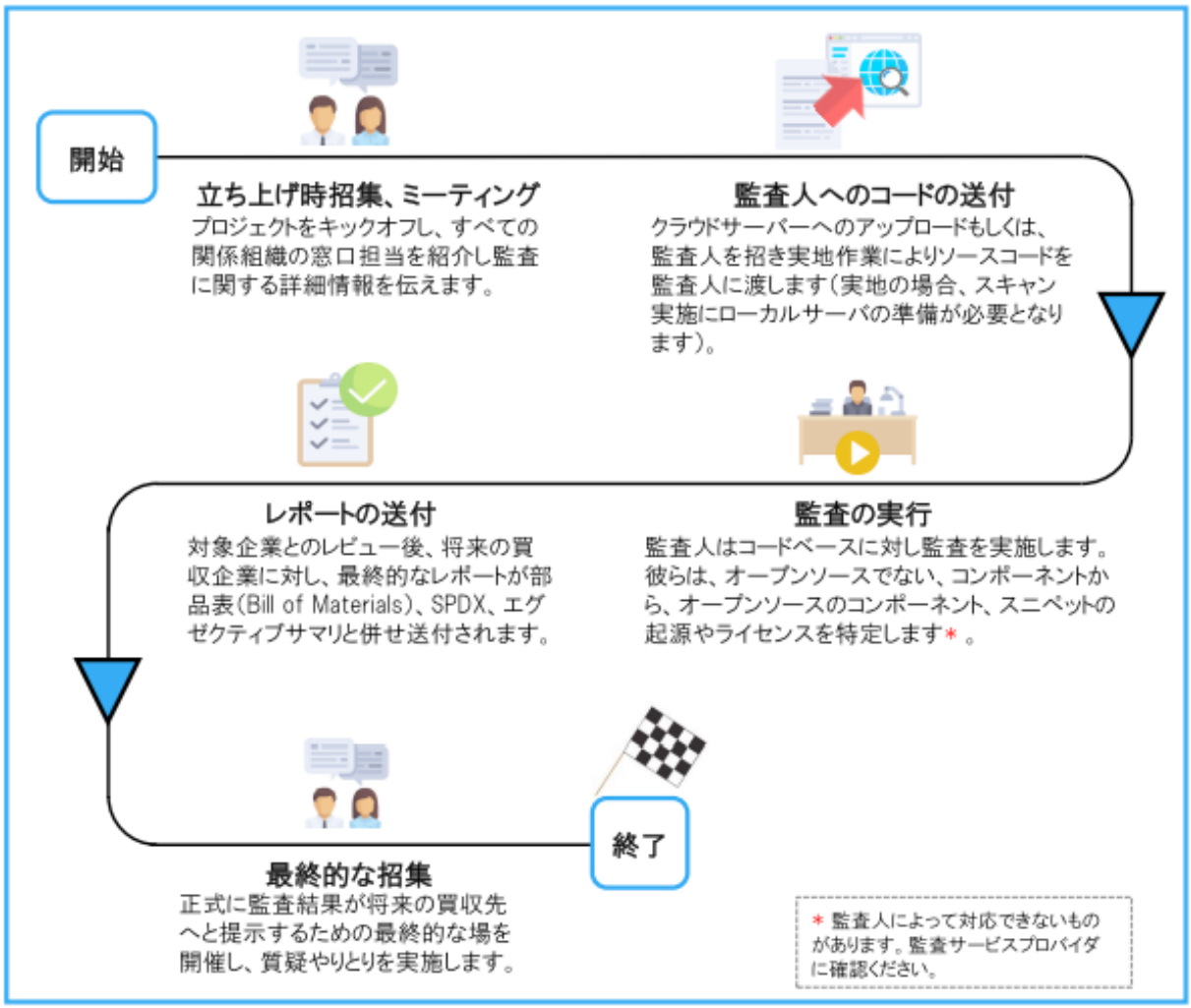


図5:M&A取引における伝統的な監査手法

図5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質問状を送付する
- 買収対象は、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5、6、7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の開始を想定しています。)

- セキュアなクラウド経由のアップロード、もしくは実地訪問に基づく監査で監査人に対し対象企業のコードへのアクセス権が与えられる
- 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、監査人とともに結果をレビューし、質疑のやり取りを実施する

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複数企業からの入札を集め、要求に合った最良の入札者を選択することもできます。このモデルでは、買収対象企業はコードを監査人へ送付し、監査人が実地に訪れることに協力的でなければなりません。

5.2 ブラインド監査

ブラインド監査は、ストックホルムを拠点としたFOSSID AB社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。(ここではFOSSID ABが会社名で、FOSSIDがツール名としています。)

FOSSID AB社のプロプライエタリな技術を用いることで、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が可能になります。図6に、FOSSID AB社が用いる、M&A取引においてソースコードの機密を保つ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、買収企業が事前に十分配慮することで監査人にも買収対象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。著者の認識する範囲ではありますが、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。

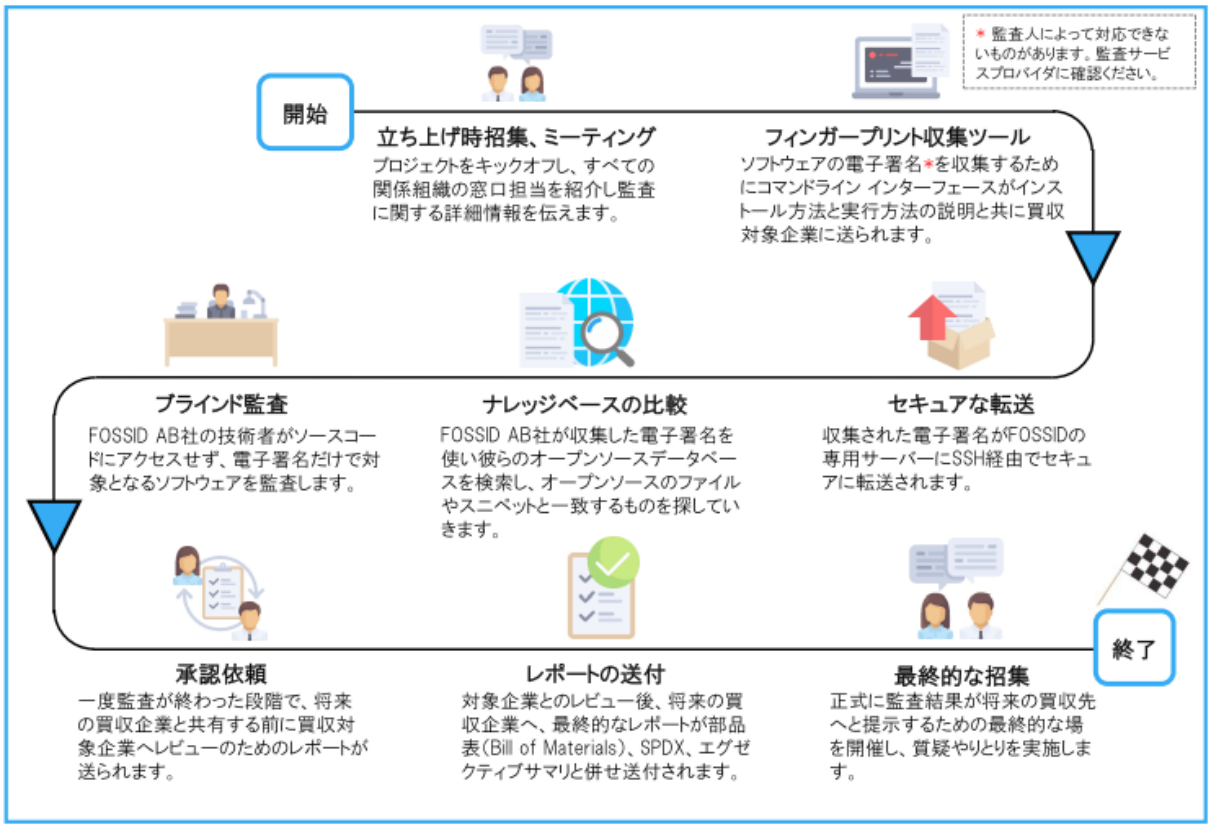


図6:M&A取引を想定したFOSSIDを用いたブラインド監査

5.3 DIY監査

Do-It-Yourself(DIY)監査は買収企業もしくは買収対象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 ツールへのアクセスを提供します。すべてのナレッジベースやレポート機能にアクセスすることで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是正手続きを提案する十分な知見がある従業員をもつ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はこの手法により迅速に費用効率を上げることができます。さらなる監査の完全性確保を目的として、監査ツールのサービスプロバイダが発見事項の検証を実施し、独立した形で認定を行う手法もあります。

図7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。たとえば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ときにすぐに監査を開始できる点が挙げられます。時間の短縮の可能性、社外のコストの削減といったこともあります。コンプライアンスの問題がどんなものでも、直接コードにアクセスし、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。FOSSID AB社におけるDIYサービスの一部として、対象企業で監査されるべきものと示されたファイルのX%(このX%は見積もりの合意の一部として決定されます)に対し無作為抽出検証が提供されています。

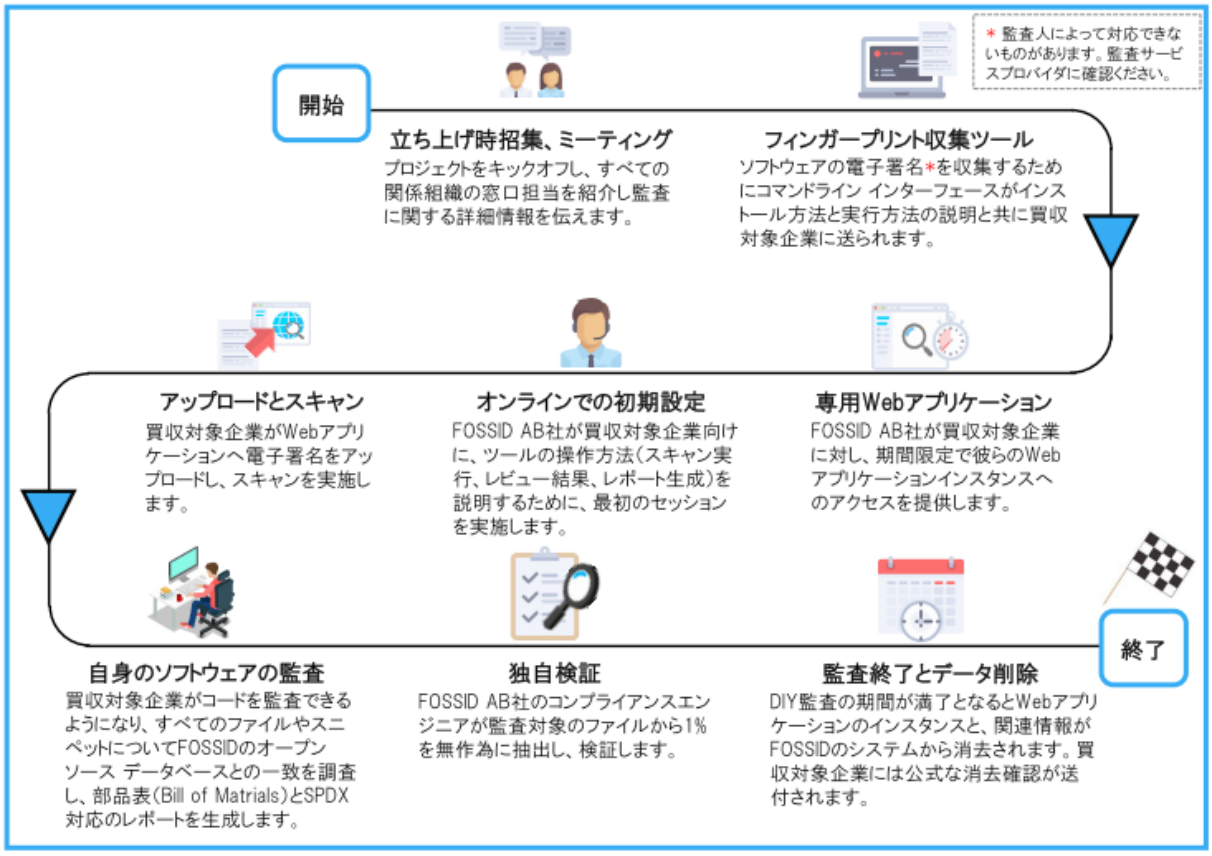


図7:M&A取引を想定したFOSSIDを用いるDIY監査プロセス

6 最終レポートに関する留意事項

多くの監査ツールは、存在する可能性のあ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、多くが問題ではなかったということがありますがこういったノイズとしてで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。このためレポートは当初長いものとなる場合があるので、時間をかけてでも事前に準備し、真の問題を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。

通常Software Package Data Exchange (SPDX) に準拠したレポートが要求に応じ提供されることにも留意しましょう。したがって、監査サービス提供者にそのようなレポートを用意してもらいたいときには、それを事前に要求することが必要となるでしょう。

7 セキュリティとバージョン管理

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経年劣化するもの、というのは一般的に受け入れられていることです。そしてオープンソースであれ、それ以外のものであれ、すべての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は、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ことになります。脆弱性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の範疇ではありませんが、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マッピングするようなサービスが合わせてソースコード スキャンのサービス企業から提供されることがあります。

8 買収前、買収後の改善

この段階で買収企業は、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 ライセンスの義務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もっているべきです。こういった情報は、さまざまなコンプライアンス上の問題に対する是正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いくつかの選択肢が考えられます。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。

そのコードが必須のもの、以前に頒布されていたものだとすれば、残される選択肢はそのコードをコンプライアンス状態にすることしかありません。それぞれの選択肢で要するコストは対象企業の買収価格を決定する際に考慮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取り込むのに誰が関わったのかを特定し、その人たちに是正作業に参加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資料や知識を彼らがもっている可能性があるからです。

9 買収対象企業として監査に備える

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。しかし買収企業が関心を見せたときに初めてその準備を始める、といったことだとするとそうとはいえないでしょう。ここで示す活動は、日常のビジネスや開発と密接に関係しています。その目的は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。その企業が企業取引の対象になったときに、これらの取り組みは、好ましくないサプライズのリスクを最小にしてくれる点で大きな助けとなってくれます。

9.1 コードの中身を知る

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。すべてのソフトウェア コンポーネントについて、起源やライセンス情報などを伴いつつそれらを網羅した目録を保持していく必要があります。目録には、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を記載します。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する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

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

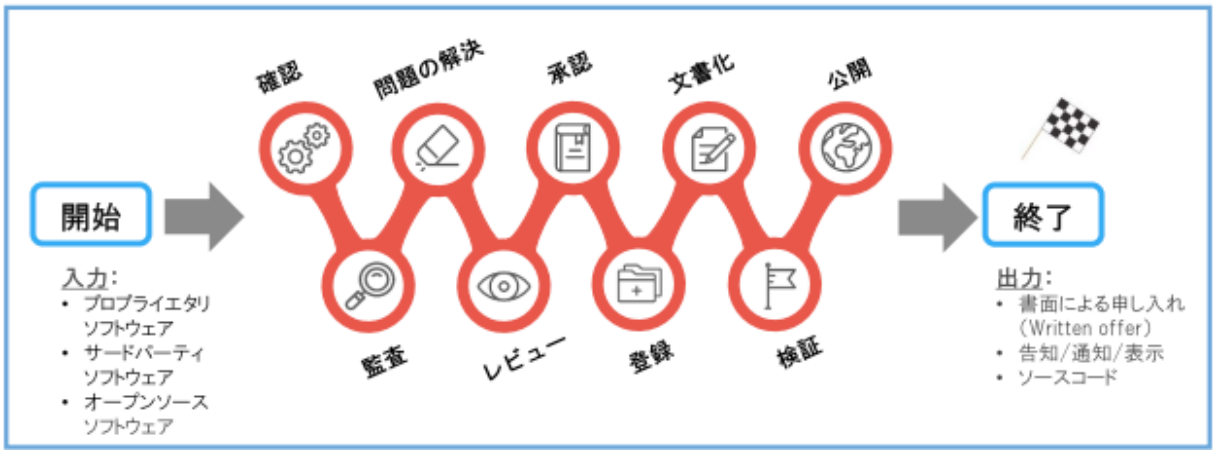


図8:オープンソース コンプライアンス プロセスの開始からから終了まで(サンプル)

アの使用、コントリビューション、監査、頒布といったさまざまな側面から統制していくことになります。

図8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. 適切なレビューを実施し、これを完遂する
5. オープンソースの使用についての内部承認を得る
6. ソフトウェア目録にオープンソース ソフトウェアを登録する
7. 製品の関連文書にオープンソース ソフトウェア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、頒布に関する最終検証を行う

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)ですが、それに書面による申し入れ(Written offer)、著作権、ライセンス、帰属表示の告知文など、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を伴います。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」を参考にしてみてください。

9.1.2 スタッフ

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、一人以上の法務専門家、およびコンプライアンス 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、ローカライゼーションなど、コンプライアンスの取り組みに継続的な貢献をする、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。しかし、小規模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務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それぞれの会社で違ってきます。

9.1.3 トレーニング

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確実にすることができます。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、トレーニングは製品やソフトウェア ポートフォリオにオープンソース ソフトウェアを取り込むことのビジネス上、法務上のリスクについてもカバーすべきでしょう。

トレーニングとして形式的なもの、非形式的なもの両方を活用できます。形式的な手法には、コース修了のために従業員が確認試験に合格する必要がある、インストラクターが指導するトレーニングコースがあります。非形式的なものには、Webinerやブラウンバッグセミナー(昼食持ち込み可のセミナー)、そして採用した従業員に対するオリエンテーション セッションの一部として使うプレゼンテーションといった

ものがあります。

9.1.4 ツールの活用

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、ソフトウェア目録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

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。これはコードの中に何があるかを整った部品表（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が実施しやすいものになるからです。

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. オープンソース ソフトウェアのすべての使用を追跡している
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ごとに同じプロセスを繰り返している
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

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、安全でないバージョンのオープンソース ソフトウェアを搭載した製品を見つけ、それを置き換えるのが容易になるということが挙げられます。最近で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ためにも、企業はオープンソース プロジェクトのコミュニティに積極的に参加することが望まれます。使用しているオープンソース プロジェクトのすべてにおいてアクティブに活動することは合理的、現実的ではないため、最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することが必要となります。参加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、さらには大規模な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、セキュリティの脆弱性や修正対応に関するレポートに目を配っておくことは有益なものといえます。

9.4 コンプライアンスの取り組みを測る

組織の規模によらず、最も簡単でかつ効果的な最初のステップはOpenChainプロジェクトに参加し、「OpenChain適合（OpenChain Conformant）」のステータスを得ることです。オンラインもしくは机上で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は、組織のオープンソース ソフトウェア 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に似た産業標準で、全体像を描くことに焦点を当てていますが、個々の組織にまでいたる適切なプロセス、ポリシーの実装も伴っています。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に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の「**自己査定用チェックリスト(Self-Assessment Checklist)**」はオープンソースコンプライアンスプログラムを成功させるために必要な要素に加え、コンプライアンスのベストプラクティスを提供した、幅広いチェックリストです。企業はこれを社内の自主運営型のチェックリストとして使用し、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ことが奨励されています。

10 買収企業として監査に備える

あなたが買収企業だとしたら、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、アクションをとり意思決定をしなければなりません。

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

前述のとおり監査では主に3つの手法を用いることができますが、そのどれが自社の具体的状況と合っているのかは決める必要があるでしょう。

10.2 何に留意すべきかを知る

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な情報を提供することがあります。どのライセンスやユースケースが大事だとみなす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

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、コンプライアンスとしての懸念事項を明確化し、確認するためには多くのデータ点でさらなる調査が求められるでしょう。本節では、買収企業にとって必要なことが何か、買収対象とともに取り組むべき問題は何か、といった枠組みを作るためのスタート地点としての質問集を以下に提示します。

- 買収企業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、十分成熟しており、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場合、買収対象企業はオープンソース ライセンスの義務を履行するために必要なすべての資料（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？
- 買収対象企業は、ソースコード要求に対しタイムリーな形で対応することができるプロセスを用意しているか？

10.4 買収取引実行前の段階で解決すべき項目を特定する

取引にもよりますが、オープンソース監査がライセンス実務やコンプライアンス実務について、買収企業にとって受け入れられない事実を明らかにするケースがあります。そういったときに買収企業は、そういった事実の影響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、一方で買収企業が「Aライセンス」のソースコードの使用を禁止する、厳格なポリシーをも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、可能な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是正計画を策定する

コンプライアンス是正計画を策定することは、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際に重要になってきます。このシナリオでは買収企業はしばしば買収対象に対し、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、買収企業で使われているトレーニングを提供し、そして継続的指導・支援を行います。

11 コンプライアンスに関し推奨される開発実務

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。それらに従うことによって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できるでしょう。

11.1 推奨プラクティス

- 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
- 当該オープンソース 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 コードをオープンソースの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
- 実施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
- 開発する、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。これにより他者が(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懸念事項を明確にすることができる
- 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として保存できる
- 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 パッケージに記述された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

- もともとあったライセンス情報・著作権情報の削除や修正をしないこと。こういった情報はすべて元通りの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自社製品のソース ツリーにコミットしないこと
- 事前承認なく異なったライセンスのもとで外部から入手したコードをマージしたり、混合させたりしないこと
- 社外の人々と、コンプライアンスのプラクティスについて議論しないこと

12 結論

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、デューデリジェンス全般で重要な側面を有しています。オープンソースのデューデリジェンスは、長期に亘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。

どうすれば準備しておくことができるのでしょうか？

買収対象企業であれば、開発プロセスとビジネス プロセスが以下のような点を確実に実行することで、適切なオープンソース コンプライアンスのプラクティスを維持することができます。

- 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 ソフトウェアのコンポーネント、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、あるいは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

買収企業であれば、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。

- 買収対象企業とともに、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、ブラインド監査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。このステップは単純に費用の話だけではなく、買収企業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迅速化のための買収企業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

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す。よいオープンソース コンプライアンスのプラクティス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

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)

The Linux Foundationで公開されている「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)」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)

The Linux Foundationで公開されている「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。本書の目的は、共通に抱える課題に実践的な情報提供をすることにあります。

OpenChain Curriculum (OpenChain カリキュラム)

「OpenChain Curriculum(OpenChainカリキュラム)」は、組織がOpenChain Specification(OpenChain仕様書)で定義されたトレーニングやプロセス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)

The Linux Foundationが提供する、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です。

Software Package Data Exchange (SPDX)

SPDX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*

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

オープンソース コンプライアンスツール*

- FOSSologyはオープンソース 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およびツールキットです
- Binary Analysis Tool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検出します。

* リストに挙げたプロバイダやツールについて記載漏れがあるかもしれませんが、ご容赦下さい。

謝辞

OpenChain プロジェクト 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、校正対応をしてくれたことに対して。

Phil Odence(Black Duck Software、VP&GM)へ。M&A監査プロセスについての議論をしてくれたことに対して。

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をしてくれたことに対して。

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), そして 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プロジェクト、Program Director)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

Ibrahim Haddad (Ph.D.) はSamsung Research America のR&D部門のVPでありOpen Source Group (OSG)の長でもあります。彼はSamsungにおけるオープンソース戦略の監督、その実行、社内外の研究開発におけるコラボレーションやM&AやCVC(コーポレート ベンチャー キャピタル)活動の支援などを担当し、オープンソース関連団体においてSamsungを代表する立場を担っています。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善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

THE LINUX FOUNDATION

The Linux Foundation は、Linux の普及促進、保護、ならびに標準化に取り組み、Linux/OSS がクローズドなプラットフォームに対抗するのに必要とされる統合されたリソースとサービスを提供します。

The Linux Foundation およびその他の活動については、http://www.linuxfoundation.org を参照してください。