

OSS教育資料　(サプライチェーンリスクマネジメント･バージョン)

この資料は、各企業がOSSのサプライチェーンリスクマネジメントを行う際に、ライセンスの調査、ライセンス義務の履行、SBOM、セキュリティリスク対応を習得する上での参考教材です。各社で教育コンテンツを作成する際の材料として自由に複製、改変してご活用下さい。

　本資料は、 [Creative Commons CC0 1.0 Universal](http://creativecommons.org/publicdomain/zero/1.0/legalcode) ライセンスの下でリリースされており、複製、改変、  
配布にあたっての制限はありません。

　　【作成元：[OpenChain Japan Work Group education sg](https://openchain-project.github.io/OpenChain-JWG/subgroups/education/)】

Disclaimer（免責事項）

* 本資料は、日本国内でOSSを活用する場合を前提としており、 education sgのメンバの経験に基づいて説明を記載しています。厳密な法解釈や海外での利用など、判断に迷う場合は専門家にご相談ください。
* 本資料に記載した内容については、作成者、提供元は一切の責任を負いませんので、ご承知のうえご利用ください。
* 本資料は、 OpenChainの仕様書準拠や認定取得を保証するものではありません。

コンテンツ

１．OSSライセンスの調査

２．ライセンス義務の履行

３．SBOM

４．OSSのセキュリティリスクへの対応

[1 OSSライセンスの調査 2](#_7x16ojmz3axp)

[2 ライセンス義務の履行 8](#_sihqp3bqdr39)

[3 SBOM 16](#_brns4glw48x0)

[4 OSSのセキュリティリスクへの対応 18](#_4dqwmz9clk5h)

# OSSライセンスの調査

* 1. 導入Q&A  
     ちょっと考えてみましょう。自社でOSS Aを利用しています。インターネット上のWebサイトの記載は、『 OSS Aのライセンスはライセンス　X』となっています。そのOSSのVersion 2を配布する場合、ライセンス　Xで配布して良いでしょうか？答えは、必ず利用するVersion 2のライセンスを確認して、対応するライセンスで配布する必要があります。  
       
     

下記の節で、ライセンスの調査に関する事項について確認していきましょう。

* 1. ライセンスの調査  
     　OSSを利用する際には、ライセンスの調査を行い、利用時にライセンスを遵守する必要があります。OSSの選定時には、先ずダウンロードファイルの調査を行い、そのOSSのライセンスを網羅的に調査する必要があります。  
     　ライセンスの調査について、下記を実例として参考に紹介します。

(a)先ず、対象となるOSSをダウンロードします。  
ダウンロードする前に、「ダウンロードする時や利用するときはこのライセンスに従うように」等と記載がある場合は、そのライセンスに従う必要があるので、記載事項をライセンス情報として確認します。

(b)(a)でダウンロードしたファイルがアーカイブされたファイル(拡張子が  
zip/jar/tar/tar.gz/tgz/xz/deb/rpm等 のファイル)である場合は、これを解凍します。

(c)(a)でダウンロードしたファイルが実行可能形式や Windows Installer 形式のファイル(拡張子が exe,msi のファイル)である場合は、これをインストールします。  
インストーラを起動し、表示されるライセンス本文やソフトウェア利用許諾契約等をライセンス情報として確認します。

(d)ファイルの名称に下記の文字列が含まれるファイルを検索します。検索されたファイルにライセンス情報の記載があるかを確認します。  
license/copying/copyright/eula/about.html/pom/manifest-readme/notice/legal/licence(※)等

(e)ダウンロードファイルの全てのファイルの中身に対して、下記の文字列が含まれているかを検索します。これらの文字列が含まれているファイルに、ライセンス情報の記載があるかを確認します。  
　license/epl/gpl/asl/artistic/legal/notice/third/licence(※)等

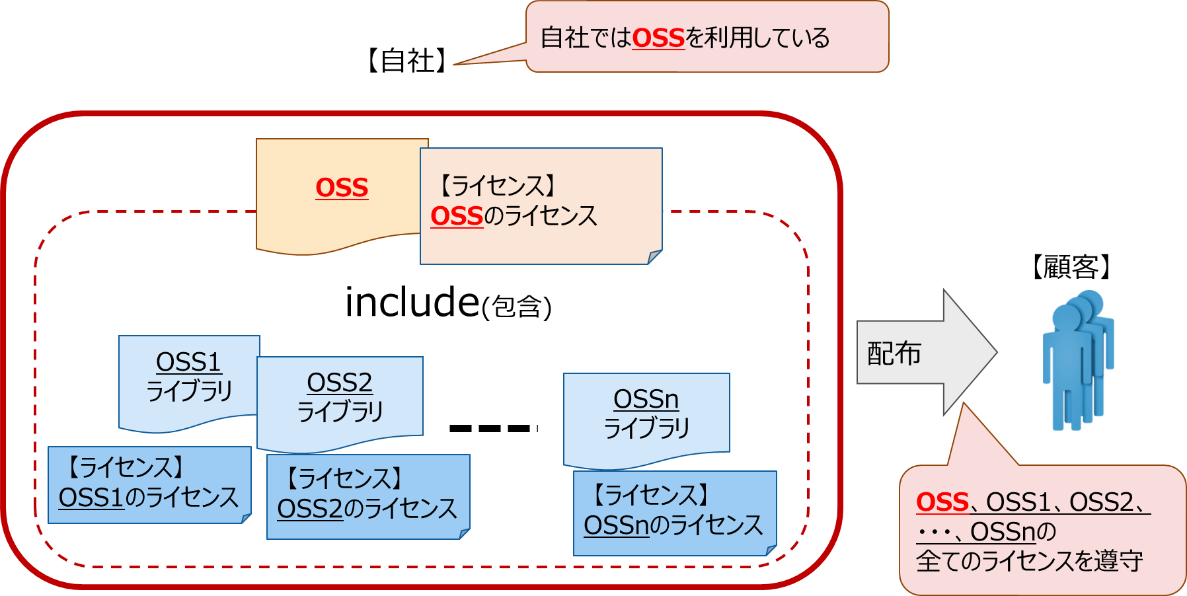
(f)(b)でアーカイブされたファイルの場合は、アーカイブされたファイル中の META INF フォルダ内の MANIFEST.MF/pom.xml 等の定義ファイルに、ライセンス情報が含まれていないかを確認します。

(g)ダウンロードサイトと公式サイト、プロジェクトサイトやベンダサイト、が別の場合には、下記を確認します。  
公式サイト、ベンダサイト上のLicense/Licensing/Readme/Copying/Copyright/Eula/Legal/about.html/notice/Thirdparty/Thirdparties/Licence(※)  
等の文字列を探し、ライセンス本文やライセンスへのリンクを調査します。

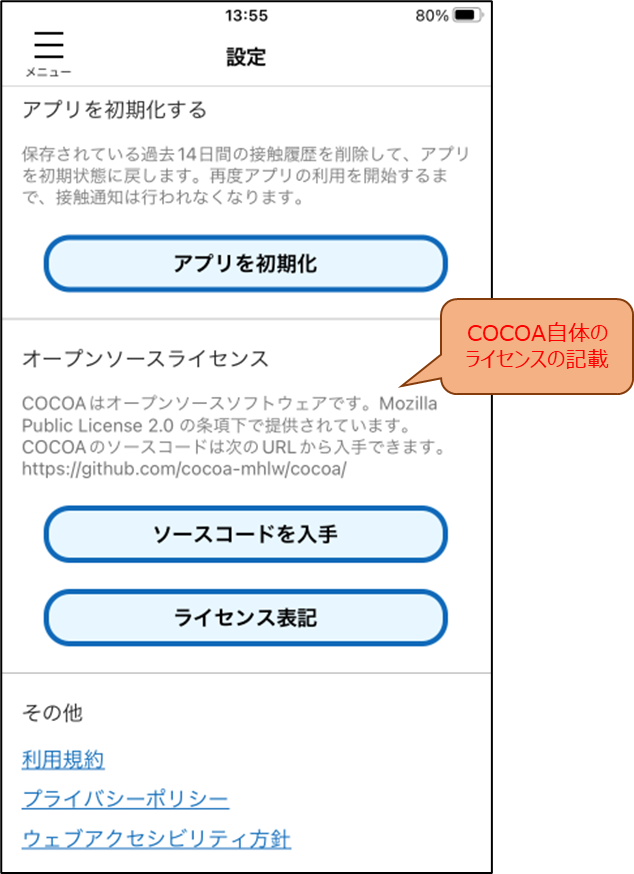
　上記の実施例の(a)～(f)の作業で、ソースコードファイル中のライセンス情報を網羅的に確認する事が基本となります。(g)の作業は、あくまでも補助的な作業と考えた方がよいと思います。(a)～(f)の作業で調査したライセンスと、(g)の作業で調査したライセンスが異なる場合、前者のライセンスと後者のライセンスが異なる理由を明確にし、その理由によりどちらを選択するかを判断し、いづれかを選択して使用した方がよいと思われます。

　OSSのオブジェクトコードやロードモジュールを利用する場合でも、対応するソースコードをダウンロードして、ライセンスの調査を行う事が重要です。そこで、オブジェクトコードやロードモジュールのバージョンと、ソースコードのバージョンが一致している事を確認しておく必要があります。また、ロードモジュールには、ライブラリ等が追加されている場合もあります。追加されているライブラリ等のライセンスも確認が必要です。

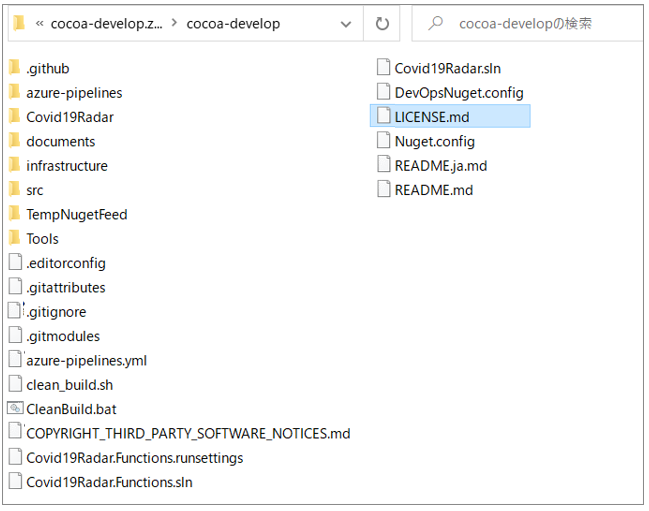
　上記の調査でライセンス情報が見つからない場合には、コミュニティの開発者に確認が必要でしょう。それでもライセンス情報が明確にならない場合は、そのOSSは利用禁止にする必要があります。  
(※: licenceは、licenseと同様の意味で使用される場合があります。)

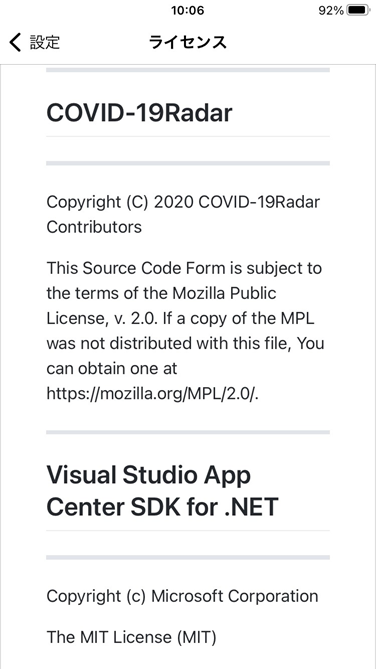
* 1. ライブラリ等のライセンスも確認  
     　近年、OSSも(特にOSやフレームワーク等で)大規模化が進んでおり、他のOSSのライブラリ等を含んでいるものが多々あります。そこで、1.2項のライセンスの調査を確実に行い、網羅的にそのOSSに含まれている他のOSSのライブラリ等とそれらのライセンスを調査する必要があります。  
     

コンテナイメージでの利用を行う場合にも、コンテナの中に含まれるOSS、OSSのライブラリ等とそれらのライセンスを調査する必要があります。

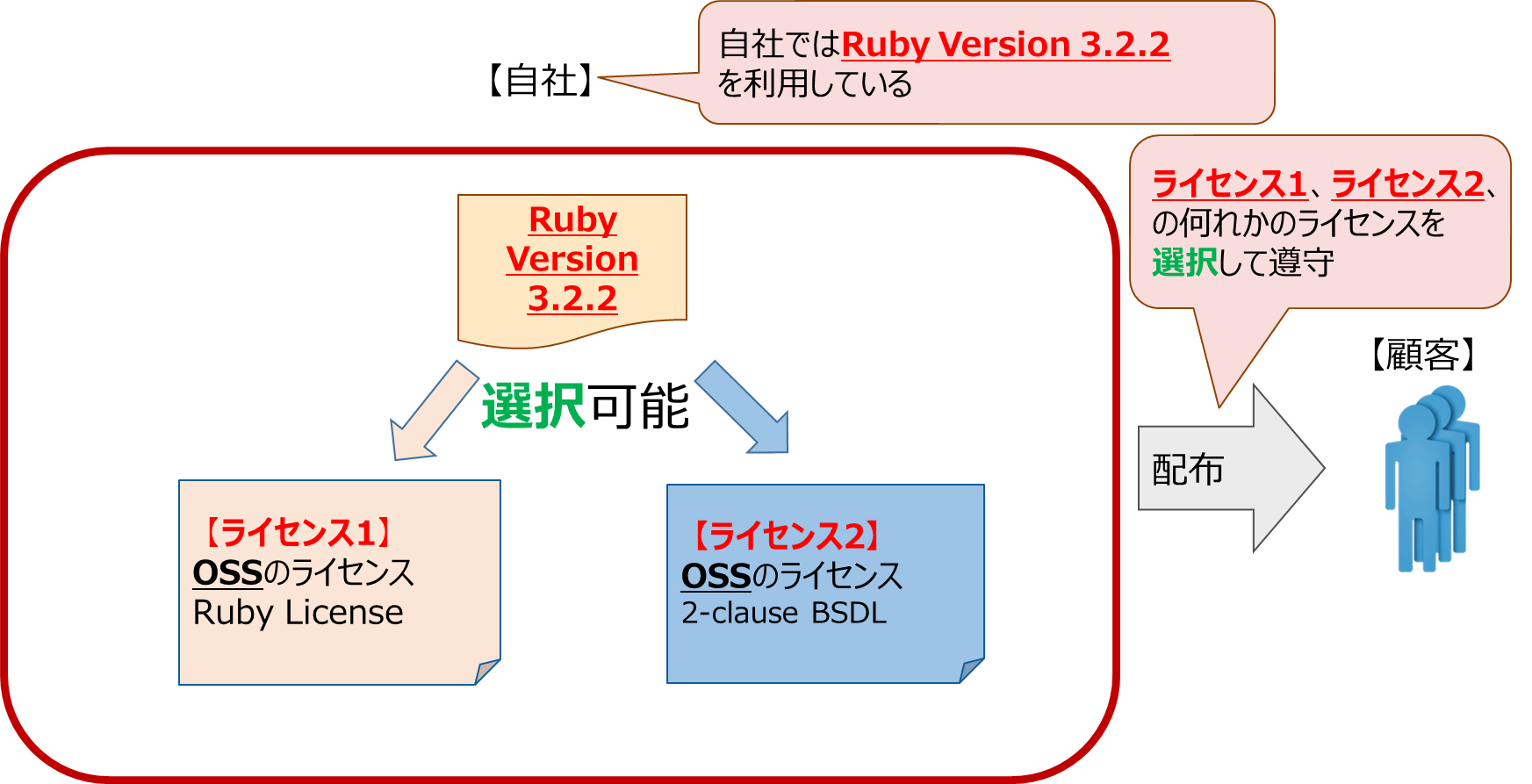
* 1. ライセンスの記載例
     1. OSSをオブジェクトコード、ロードモジュールで利用する場合には、例えば配布するアプリケーションやプログラム上に、ライセンスの記載を行う場合があります。  
        下記は、COCOAのアプリケーションの例を示しています。  
          
        
     2. OSSをソースコードで利用する場合は、ファイル群の中にライセンスが記載されていることがあります。

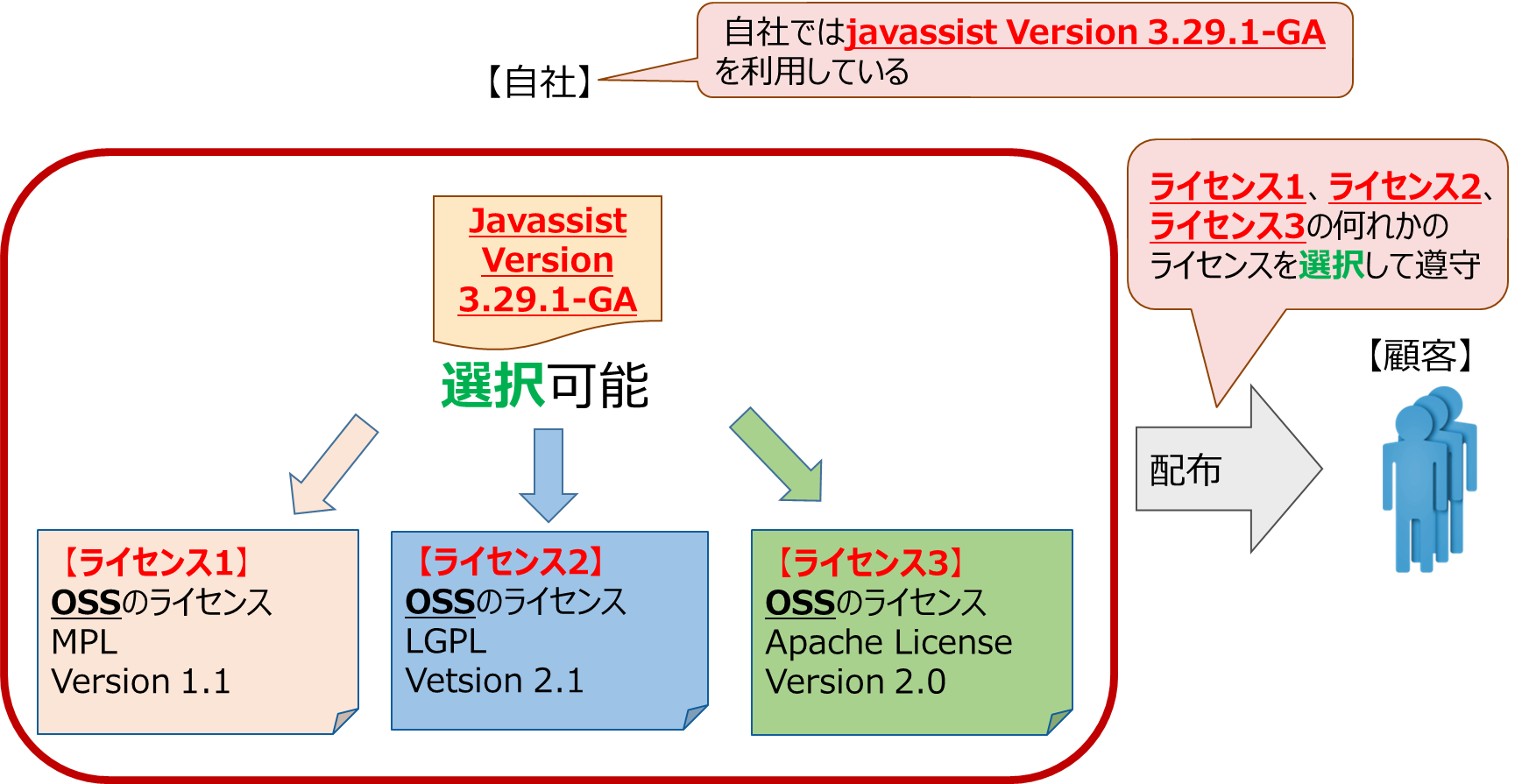
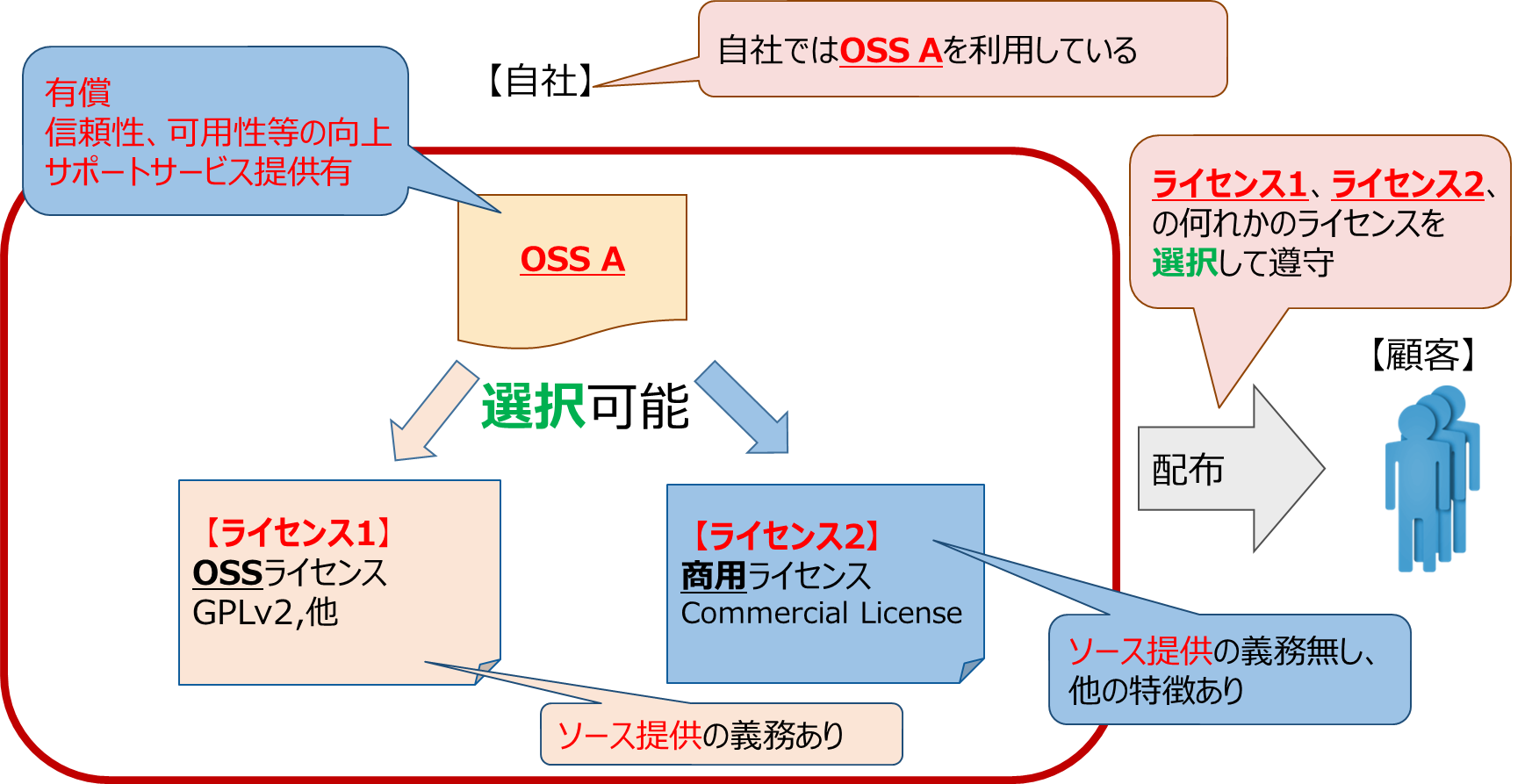
下記は、新型コロナウィルス接触確認アプリ「COCOA」の、Web サイト上でライセンスを表示している例です。  
（図中水色枠部分、[https://github.com/cocoa-mhlw/cocoa）。](https://github.com/cocoa-mhlw/cocoa%EF%BC%89%E3%80%82)



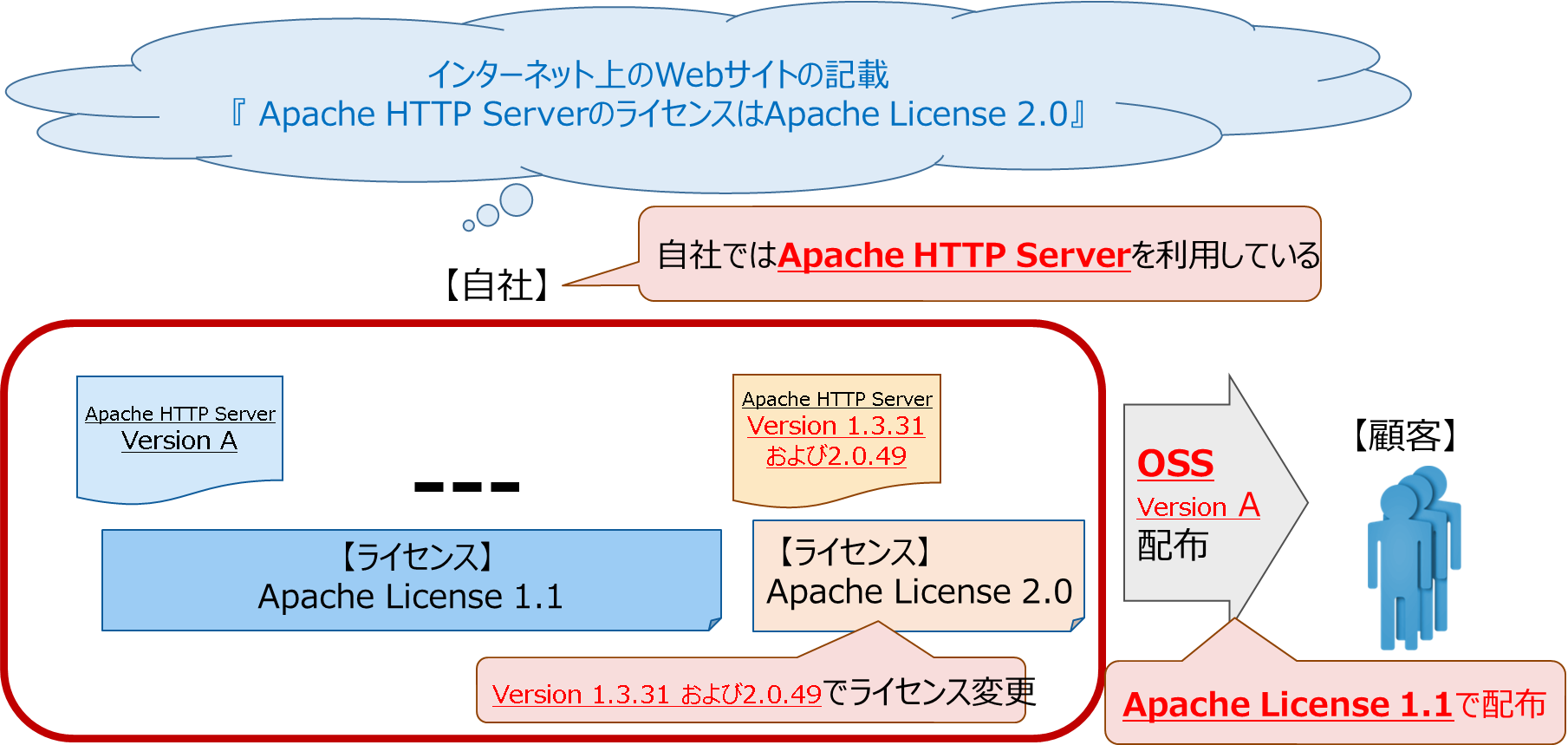
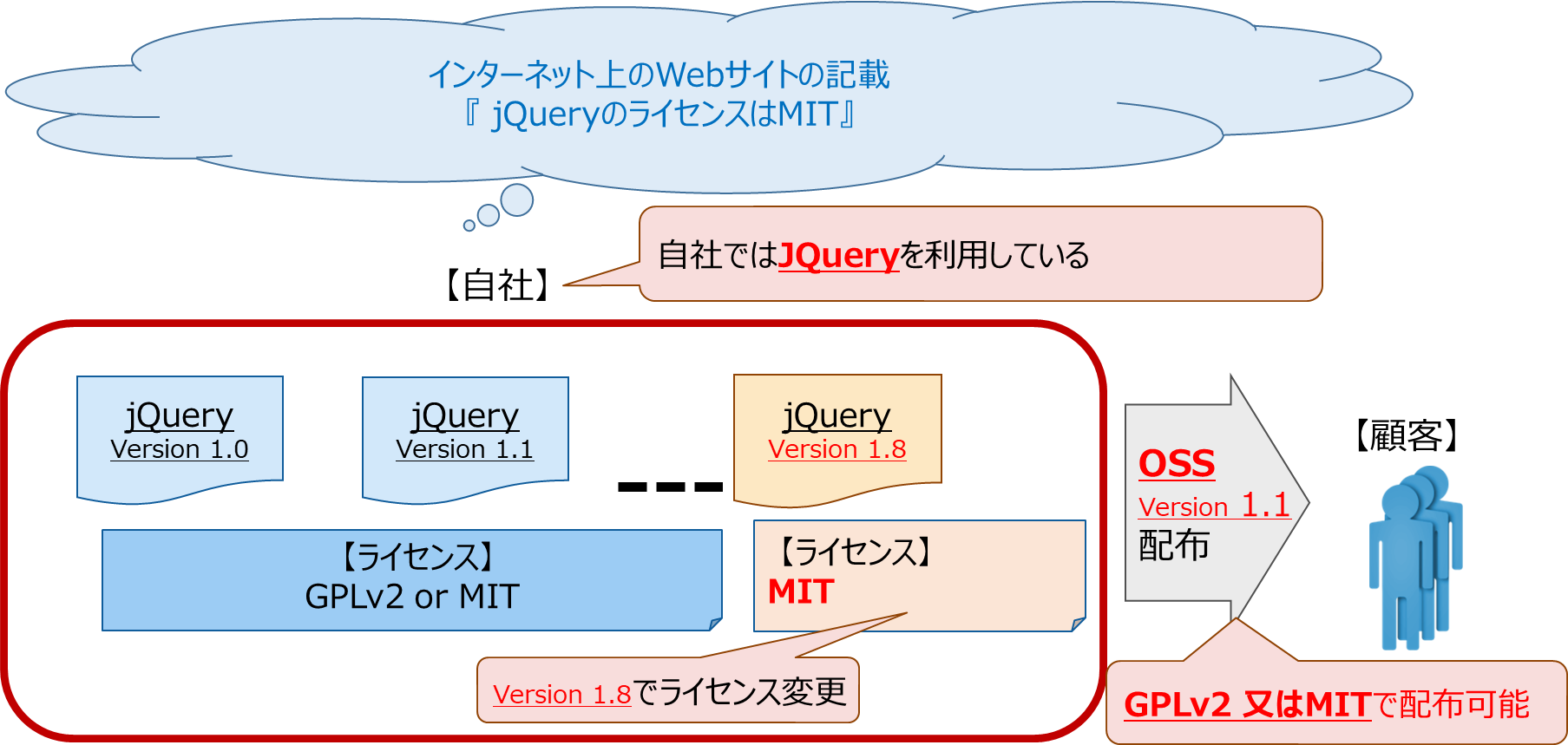


* 1. 複数のライセンスが記載されているケース  
     　OSSの中には、複数のライセンスの中から所望のライセンスを選択して利用できるものがあります。２つのライセンスから選択可能なものはデュアルライセンス、３つのライセンスから選択可能なものはトリプルライセンス、またはマルチライセンスと呼ばれています。
     1. デュアルライセンスの例としてはRubyがあります。下記URLを参照ください。  
        <https://www.ruby-lang.org/en/about/license.txt>



* + 1. トリプルライセンスの例としてはjavassistがあります。下記URLを参照ください。  
       <http://www.javassist.org/>  
         
       
    2. ライセンスの選択については、商用ライセンスとOSSライセンスから選択するケースがあります。  
       例えば、OSSのコミュニティで開発されているOSSに対して、ソフト開発ベンダーが、信頼性、可用性等を向上する目的で機能追加し、商用ライセンスで提供するケースがあります。この場合、一般的に有償で提供され、ソフト開発ベンダーによるサポートサービスが提供される場合があります。  
       また、別の例では、ソース提供義務のあるOSSライセンスと、ソース提供の義務の無い商用ライセンスを選択できるようになっている場合があります。  
         
       
  1. Web掲載のライセンスで誤認(バージョン)

OSSは、そのバージョンによって、ライセンスが変わっている場合があります。インターネット上のWebサイトには最新のバージョンのライセンスが記載されている事が多く、バージョンの古いOSSにも同じライセンスが適用されると判断してしまうと、OSSを配布する場合に、誤ったライセンスで配布してしまうおそれがあります。必ず、配布するバージョンのライセンスを確認しましょう。

* + 1. 事例１：Apache HTTP Serverは、Version 1.3.31 および2.0.49になる際に、Apache Licenseも1.1⇒2.0となりました。  
         
       
    2. 事例2: jQueryは、Version 1.8より以前は、GPLv2とMITのどちらかを選択するデュアルライセンスでしたが、Version 1.8でMITに統一されました。  
         
       
  1. OSS、ライセンスのリスト化
     1. 1.2～1.6のようにOSSのライセンスを調査して確認することは、OSSを利用する際にそのライセンスの義務を履行するためにも重要になります。また、第３章のSBOMを作成するためにも必要になります。  
        そこで、ライセンス調査の結果は、下記のような項目についてリストにして纏めておくと良いでしょう。  
        (1)OSS名称  
        (2)OSSのバージョン  
        (3)OSSのダウンロードURL  
        (4)ライセンス名称  
        (5)著作権等の表示
     2. OSS、ライセンスのリスト化を行っておくと、社内でOSSのコンプライアンスについて検討、承認を行う際に、大変有効に活用できます。

# ライセンス義務の履行

* 1. 導入Q&A  
     　本章に際して、はじめにOSSを利用した製品開発とOSSライセンスを遵守するための作業について、下記の例を通して考えてみましょう。

　A社は現在開発中の製品にOSS「X」を搭載することにしました。Xはある分野において非常に有名なOSSで、その名前や特徴、開発者について誰でも知っています。そのためA社では、製品のドキュメントの一部に「OSS『X』を利用しています」と一言だけ添え、その他の詳細な表現は省くことにしました。このときのA社の対応はOSSライセンスの遵守作業として果たして十分だったと言えるでしょうか。

　残念ながら話はそう単純ではなく、OSS「X」のライセンス条件によってはライセンス違反とみなされる可能性があるため、「十分に遵守された状態」とは言えません。こうした事態を避けるためには、はじめにOSSライセンスをきちんと理解し、その条件を遵守する必要があります。本章ではOSSライセンスの代表的な遵守作業とそのポイントについて確認します。

* 1. ライセンス遵守作業の概要  
     　下記の3点はOSSライセンスの遵守作業の中でも、特に代表的なものです。2.3節以降で、それぞれの具体的な作業内容について解説します。
* 必須文言（ライセンス文書等）の掲載（2.3項）
* ソースコードの提供（2.4項）
* 製品に付帯する契約書（エンドユーザライセンス等）への追記（2.5項）

　また解説の便宜上、ライセンス条件の制約の強さを5段階に類型化した「OSSライセンスのレベル」を用語として取り扱います。各レベルの詳細については「OSS教育資料、サプライチェーンリスクマネジメント・バージョン　ライセンス基礎」第2章をご参照ください。

Table X.　ライセンス条件の制約の強さ（５レベル）

|  |  |
| --- | --- |
| レベル | ライセンス条件の概要 |
| レベル１ | OSS配布時、ライセンス文書を添付する |
| レベル２ | OSS配布時、レベル１に加えて、著作権関連情報等をドキュメントに記載する |
| レベル３ | OSS配布時、レベル２に加えて、対象OSS のソースコードを提供する |
| レベル４ | OSS配布時、レベル３に加えて、対象OSS と他のソフトウェアを組み合わせてひとつの著作物となった全体のソースコードを提供する |
| レベル５ | レベル４に加えて、サービス利用（SaaS/ASP等）の場合でも、サーバへのアクセス者へ、レベル４と同様のソースコードを提供する |

* 1. 必須文言（ライセンス文書等）の掲載  
     　大半のOSSライセンスは、OSSの著作権情報やライセンス条文、免責条項など、OSSに関する基本的な情報も受領者に提供するよう求めています。OSSライセンスによっては、ライセンス条文を製品のドキュメントに掲載するのみで良いものや、ライセンス条文に加えて謝辞の掲載を求めるもの、広告媒体上での謝辞の掲載を必要とするものもあります。

下記は主な掲載方法の例です。具体的な条件がライセンスに詳述されている場合もあるため、実際に検討する際は各ライセンスの条文をきちんと確認することが重要です。

* 製品の取扱説明書やエンドユーザライセンスなどのドキュメントに含める
* （製品自体に情報表示機能がある場合）オンラインヘルプやオプションメニューなど、ユーザが使用する機能の中で表示する
* （HTML形式など、ソースコードのまま第三者に提供する場合）ソースコードに予め埋め込まれた著作権表示を削除せずに残しておく

ひとつ注意点として、掲載場所は「OSSの受領者にとって自然に見る場所」で行う必要があります。

例えば「取扱説明書やエンドユーザライセンスが紙の冊子であるにも関わらず、何の案内も無いままに、OSSに関する著作権情報やライセンス条文のみを専用のCD-ROMで保管する」といった方法は、あまり好ましくありません。OSSの受領者が意図的に「製品に含まれるOSSの情報を調べよう」としない限り、目に触れることがないからです。もし他の製品ドキュメントと分離して提示する場合は、その旨を明確に案内する文書を追加するなど、OSSの受領者にとってわかりやすい形となるよう配慮しましょう。

以下にライセンス条文の掲載、謝辞の掲載、広告媒体での謝辞の掲載について説明します。

1. 製品のドキュメントにライセンス条文を掲載する  
   　OSSライセンスの条文は多くの場合、「COPYING」や「LICENSE」という名前のテキストファイルに格納されています。下記はレベル１のOSSライセンスであるMIT LICENSEのテンプレートです。上から順に著作権情報、ライセンス条件、免責条項についてそれぞれ記載されています。  
   * ライセンス条文の多くは著作権情報やライセンス条件（許諾される行為と制約事項、遵守すべき条件）、免責条項が英文で記載されており、大抵はこれをそのままOSS受領者に提供することでライセンス条件を満たすことができます。
   * ライセンス条文を掲載する際は**未改変のまま保持する**ことが重要です。ライセンス条文を変更せず、和訳や解釈を加えることなく、英文原文をそのまま掲載するように留意してください。

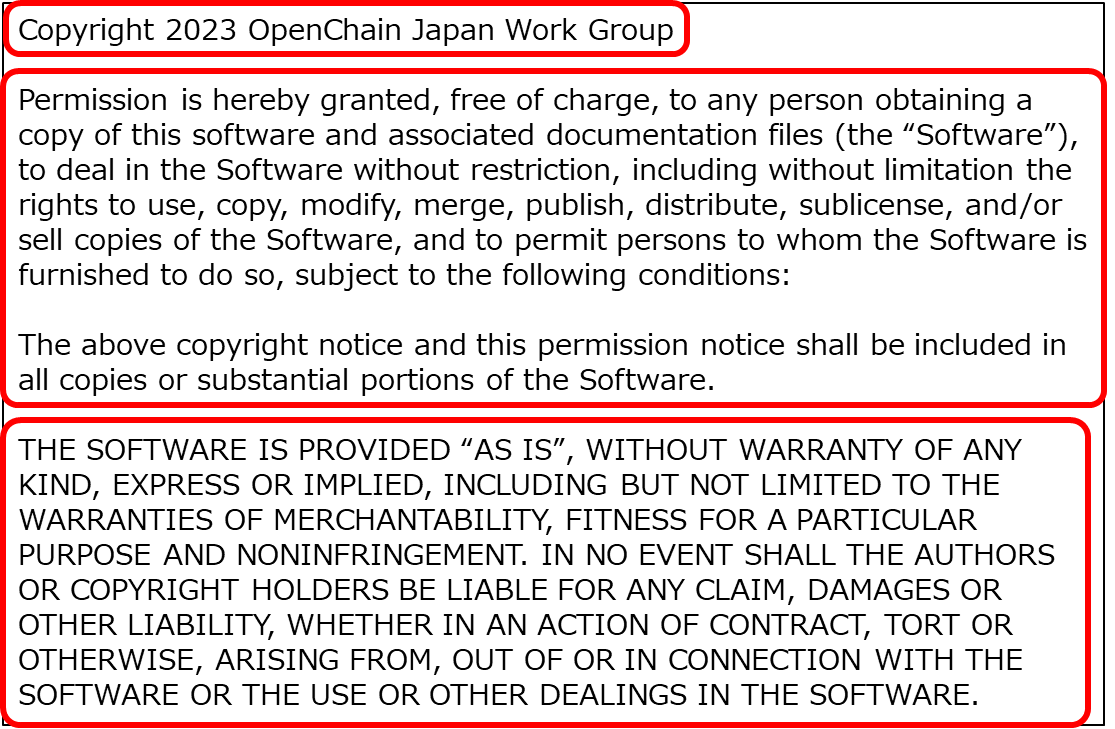


Fig X. レベル１のOSSライセンス（MIT LICENSE）

1. 謝辞を掲載する  
   　レベル２のOSSライセンスの中には謝辞（Acknowledgements）の掲載を求めるものがあります。多くの場合、謝辞は「このソフトウェアには開発プロジェクトAによる成果物が含まれています」という趣旨の一文であり、ライセンス条文にある謝辞の一文をそのまま掲載します。

　下記は謝辞の掲載を求めるライセンスと、そのライセンスに基づき実際に謝辞を掲載したソフトウェアの事例です。Libjpeg Licenseは謝辞の一文（”this software is based in part on the work of the Independent JPEG Group”）を製品付帯文書に掲載するよう求めており、それに対してLibjpeg LicenseのOSSを利用するWebブラウザ「Firefox」では謝辞の一文を”Other Required Notices”に掲載しています。

テキスト, 手紙

自動的に生成された説明

Fig X. 謝辞の掲載を求めるライセンス例  
（Libjpeg License, [https://jpegclub.org/reference/libjpeg-license/）](https://jpegclub.org/reference/libjpeg-license/%EF%BC%89)



Fig X. 謝辞の記載例（Web ブラウザ「Firefox」、about:license#other-notices）

1. 謝辞を広告媒体に掲載する  
   　レベル２の一部のOSSライセンスには、「このソフトウェアの機能または使用について言及するすべての広告媒体にも謝辞を掲載」するよう求める条項があります。例えばOSSを搭載した製品について販売促進を目的としたWebページ上でOSSの機能について記載する場合、ライセンスの求める謝辞も掲載する必要があります。

　下記は広告媒体での謝辞の掲載を求めるライセンスと、そのライセンスに基づき謝辞を広告媒体上に掲載したソフトウェアの例です。自社Webサイトの製品紹介ページ上でOSSの機能について触れているために、謝辞の一文（”This product includes software developed by the University of California, Berkeley and its contributors.”）を記載しています。

テキスト

自動的に生成された説明

Fig X. 謝辞を広告媒体へ掲載するよう要求するライセンス条項の例  
（BSD 4-Clause Original (Old) License）

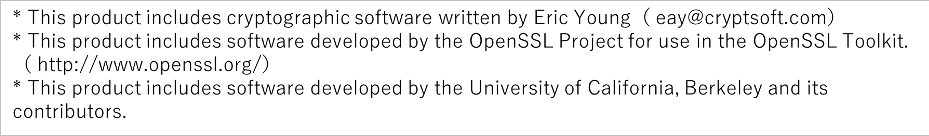


Fig X. 通信系組み込み製品における自社Webサイトの製品紹介ページでの掲載例

* 1. ソースコードの提供  
     　OSSをバイナリコードの形式で製品に含める際、ソースコードとその関連情報もOSSの受領者に対して提供するよう求めるライセンスがあります。これは受領者に配慮した条件で、例えば「受領者がOSSを改変し、製品に搭載されたOSSを改変版に差し替える」際の技術的な選択肢を残すためのものです。

　特にソースコードの関連情報として、製品搭載されたOSSのバイナリコードを完全に再現するための設定情報やそのバイナリコードを実行するための設定情報も提供するよう求めるライセンスもあるため、必ず条文を確認するようにしましょう。

* レベル３のOSSライセンスではOSSそれ自体のソースコードを提供します。OSSを改変した場合は改変後のソースコードを提供します。
* レベル４のOSSライセンスでは、レベル３の対応に加えて、OSSライセンスの定める他のプログラムのソースコードも提供します。
  + 1. 提供するソースコードの準備  
       　ライセンスの求めるソースコード一式を用意します。ライセンスによっては単にソースコードだけでなく、製品に含めたOSSのバイナリコードを再現し、実行できるようにするために必要となる全ての情報も用意する必要があります。
* 製品搭載されたOSSのバイナリコードを再現、実行するための情報も提供するよう求めるライセンスの場合、そのバイナリコードを過不足なく再現できるよう、ソースコードに加えてバイナリコード生成時の情報も用意します。コンパイルオプションや各種設定パラメータ (一般にmakeファイル一式)がこれに相当します。なお、その際に使用したコンパイラなどの「バイナリコードを再現するための環境」それ自体を提供する必要はありません。
* 改変したソースコードを提供する場合はその改変箇所を告示する等、一般的なコーディングルールやマナーに配慮するとよいでしょう。ただしライセンスによっては告示方法に条件を設けているケースもあるため、ライセンスを確認するようにしましょう。
* ソースコードを敢えて紙媒体で配布する等、バイナリコードの再現と実行を意図的に妨げる行為は避けましょう。ライセンス条件に違反する可能性があります。また、例えばOSSにもともと含まれている他人の著作権表示を自社のものに差し替える等、著作権を侵害する行為も避けましょう。
  + 1. ソースコードの提供方法の選択  
       　OSSライセンスの中には、ソースコードの提供方法について複数の選択肢を提示しているものがあります。そのようなケースでは最も適した提供方法を選択し、準備を進めると良いでしょう。

　一方、注意点としてソースコードの提供方法を限定しているOSSライセンスもあります。そのため実際に検討を行う際はライセンス条件を確認のうえ、必要に応じて担当部門（例：法務、知財、OSPOなど）と連携を図るようにしましょう。

　本節ではソースコードの提供が必要となるレベル３以上のOSSを製品に適用するケースを例に挙げて解説します。ソースコードの代表的な提供方法は下記の３点です。

方法1. 製品に含めるOSSのバイナリコードとともに、必要なソースコード一式も同梱する

方法2. 受領者のリクエストに応じてソースコード一式を提供する旨の文書を同梱する

方法3. Webサイト上でソースコード一式を提供し、そのURL等を記載した文書も同梱する

1. 製品に含めるOSSのバイナリコードとともに、必要なソースコード一式も同梱する  
   　下図のように、両者を製品本体または製品の電子媒体に直接同梱する方法です。

* 製品出荷時にバイナリコードとソースコードを同梱するため、その時点の正確な情報が確実に同梱されるという利点があります。また後述の方法２、方法３と比較して、ソースコードを提供するための体制の構築や維持も不要になります。
* 一方で注意したいのは、製品本体のバージョンアップの際はバージョンアップ版に組み込んだOSSのバイナリコードと一致したソースコードがきちんと同梱されているか、適宜見直す必要がある点です。また製品本体や製品のドキュメントの容量を圧迫するため、場合によっては媒体を追加する費用がかかるという点もあります。
* 製品リリース前には、必要なソースコード一式が適切に同梱されているか、最終的な出荷物件を確認しましょう。例えば製品出荷に至るまでの一連の工程で、バイナリコードと対応したソースコードを開発部門が下流工程の部門に対してきちんと提供できていないケースや、出荷部門の判断でソースコードが出荷対象から除外されてしまうケースなどが考えられます。同梱漏れにならないよう、各工程の担当部門間で密に連携を図るようにしましょう。

ダイアグラム

自動的に生成された説明

Fig X. ソースコードを同梱する三つの形態

1. 受領者のリクエストに応じてソースコード一式を提供する旨の文書を同梱する  
   　「受領者のリクエストに応じてソースコードを提供する」旨とその依頼方法（例：問い合わせ窓口）に関する案内を、製品本体または製品ドキュメント（紙媒体や電子媒体）に同梱する方法です。  
   　この場合、案内に基づき受領者が製品開発元に対して「製品搭載されているOSSのソースコードを提供してほしい」と依頼し、それに応じる形でソースコードを提供することになります。

* 受領者からのリクエストがあってはじめて対応する形となるため、コスト面での利点があります。受領者全てがリクエストするとは限らないことに加えて、製品同梱時の媒体コストも外部化できるためです。半面、受領者からのリクエストに速やかに対応できる体制を構築し、常に維持していく必要があります。
* 受領者が製品を使用する過程で自然に目を通すものであれば、案内の掲載場所は紙媒体でも電子媒体でもかまいません。製品に付帯する取扱説明書やエンドユーザライセンスがあればそこに記載してもよいですし、製品が何らかの表示機能を有しているならオンラインヘルプ等でも問題ありません。
* OSSライセンスによってはソースコードの提供期間について言及しているケースがあります（例：OSSを含む製品出荷後から少なくとも３年間有効など）。
* ソースコードを格納した媒体（例：CD-ROMやDVDなど）を提供する際は、送付に必要な実費を請求することも可能です。郵送費や媒体費を鑑み、妥当な価格を設定しましょう。
* 誤ってバージョン違いのソースコード一式を提供してしまわないように、ソースコード一式は製品のバージョンごとに用意しておくと良いでしょう。
* 初めから媒体を用意しておく必要はありません。あらかじめ用意しておいても、あるいはソースコードの提供依頼があった際に初めて作成しても問題ありません。
* ソースコード一式は、makeコマンド等で一括再現可能であれば、tgz形式やzip形式などの圧縮形式で提供してもかまいません。
* ユーザからの問い合わせを受ける可能性のある部門（例：ユーザサポート部門）と事前に手順をすり合わせておきましょう。たとえ製品に同梱した案内に窓口を記載していたとしても、思わぬ形で問い合わせを受ける可能性があります。適切な窓口にエスカレートされるよう関係部門にも手順を確認しておくとよいでしょう。

ダイアグラム が含まれている画像

自動的に生成された説明

Fig X. ソースコード一式を提供する旨の文書を同梱する

1. Webサイト上でソースコード一式を提供する旨の文書を同梱する

　「Webサイト上でソースコード一式を提供する」旨とその入手方法（例：URL、その他アクセスに必要な情報）に関する案内を、製品本体または製品ドキュメント（紙媒体や電子媒体）に同梱する方法です。

* Webサイト上で受領者が自由にダウンロードできる形となるため、授受に要するタイムラグやコストを低く抑えることが期待できます。一方、当然ながらWebサイトの運用ノウハウを要求されるために、社内の体制構築やインフラ整備等の追加投資が必要となる可能性があります。またダウンロードできない等、何かあった際の連絡先は設けておいた方が良いでしょう。
* 案内の掲載場所は、受領者が製品を使用する過程で自然に目を通すものであれば、紙媒体でも電子媒体でもかまいません。製品に付帯する取扱説明書やエンドユーザライセンスがあればそこに記載してもよいですし、製品の特性として何らかの表示機能を有していればそちらに記載する形でも問題ありません。
* OSSライセンスによってはソースコードの提供期間について言及しているケースがあります（例：OSSを含む製品出荷後から少なくとも３年間有効など）。
* ソースコードの提供窓口はOSSの開発コミュニティではなく、バイナリコードの提供者である製品開発元が担います。仮に受領者に対して、製品開発元の管理が及ばないWebサイトをソースコードの提供窓口として案内してしまった場合、その後の意図しないサイトの閉鎖や移転によって「ソースコードの提供」というライセンス条件を満たせなくなってしまう可能性があるためです。製品搭載したバージョンのOSSを製品開発元できちんと保持し、OSSの受領者に適切に提供できるよう、体制を整えておきましょう。
* ソースコードはフォルダ構造を維持したまま提供しましょう。「バイナリコードの再現と実行を意図的に妨げている」と捉えられないように、過度のファイル分割や公開先の分散は避けましょう。
  + Webブラウザ上でソースコードをそのまま確認できる状態になっている必要はありません。
  + makeコマンド等で一括再現可能であればtgz形式やzip形式などの圧縮形式で提供してもかまいません。
* 製品に含まれるOSSを変更した際は、Webサイトで提供しているソースコードも忘れずに更新しましょう。
  + OSSを最新版に差し替えた、あるいは改変を加えた場合は、製品に含まれるOSSと一致させるためにWeb上で提供しているソースコードも適宜差し替えましょう。
  + ただし最新バージョンだけではなく旧バージョンの製品も市場に残るような場合、最新バージョンと同様に旧バージョンのソースコードも引き続きWeb上で提供する必要があります。
* Webサイト上にアクセス制限を特に設けない場合、OSSの受領者以外の誰でもダウンロード可能になります。掲載する情報は必ず確認しましょう。
* 下図はWebサイト上でOSSのソースコードを提供した新型コロナウィルス接触確認アプリ「COCOA」におけるソースコードの提供例です。
  + 「COCOA」は新型コロナウィルスの感染拡大防止を目的として開発された、スマートフォン上で動作可能なアプリケーションです。
  + 「COVID-19Radar」のライセンスに従って、「COCOA」のソースリポジトリ上で「COVID-19Radar」のソースコードを提供しています（図中赤枠部分、<https://github.com/cocoa-mhlw/cocoa>）。

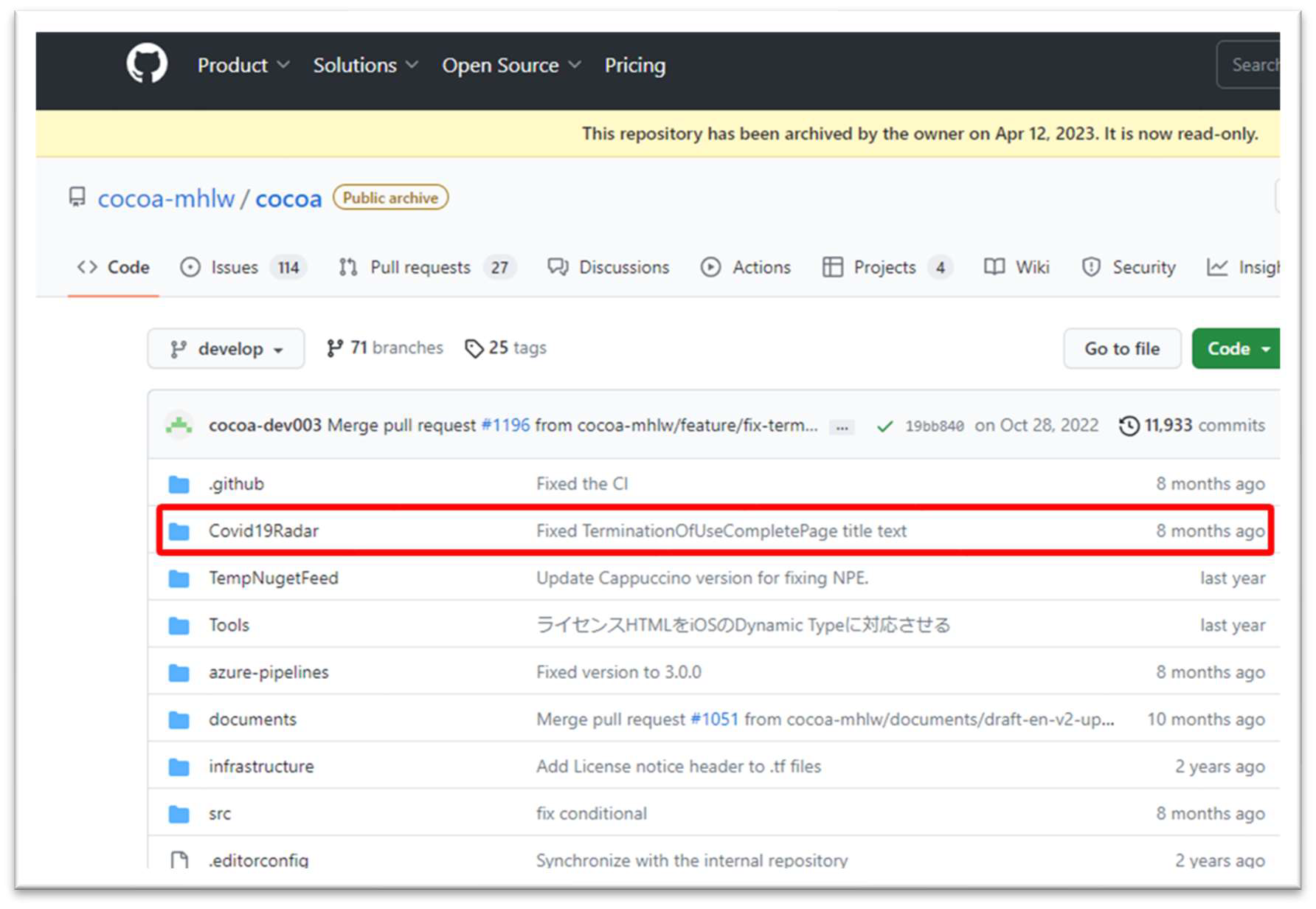


Fig X. 新型コロナウィルス接触確認アプリ「COCOA」のソースリポジトリ  
（赤枠部分にOSS「COVID-19Radar」のソースコードを公開）

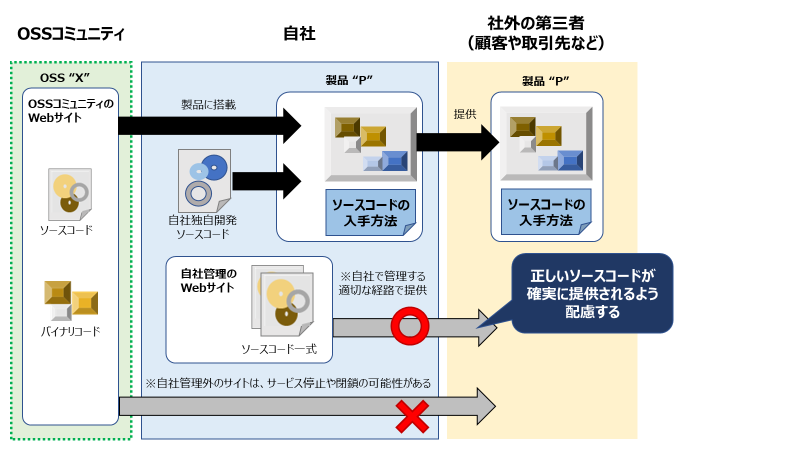


Fig X. ソースコードの提供者はバイナリコードの提供者

* 1. 製品に付帯する契約書（エンドユーザライセンス等）への追記  
     　製品を販売する際、製品の使用条件を定めた契約書としてエンドユーザライセンス（End-User License Agreement, EULA）を提示し、購入者に同意を求めることがあります。これは権利の帰属や製品の使用条件をあらかじめ明示しておくことで、双方の意図しない権利侵害や事故、損害を防ぐための一般的な措置です。

　しかしOSSを製品に組み込んだ場合、従来のエンドユーザライセンスの条件とOSSライセンスの条件との間に矛盾が生まれてしまうことがあります。例えば、OSSライセンスに「OSSライセンスの条件以外に追加の制限を課してはならない」といった条件や「OSSとリンクしたプログラムのリバースエンジニアリングを許諾せよ」といった条件が含まれるケースがこれに該当します。

　こうしたケースでは「OSSについては、エンドユーザライセンスの条件よりも、OSSライセンスの条件が優先する」旨をエンドユーザライセンス側に記載する必要があります。この場合、矛盾する条件についてはOSSライセンスの条件が適用され、それ以外の部分についてはエンドユーザライセンスおよびOSSライセンスの双方の条件が適用されることになります。

　そのためOSSを製品に組み込む際は、エンドユーザライセンスの条件がOSSライセンスの条件に矛盾しないか確認するよう、エンドユーザライセンスの担当部門や法務部門等と連携して、対応を検討することが重要です。

# SBOM

* 1. ソフトウェアの構成を知る

ここまで、オープンソースソフトウェア(OSS) においてライセンスコンプライアンスが重要だということを見てきました。

1. 自分が利用しようとする利用しようとするOSS:
2. そのOSSが開発者以外のOSSを含む場合: OS
3. そのOSSが他のOSSを利用する場合: O
   1. SBOMを取得する方法

SBOMがあれば、利用するソフトウェアのライセンスや著作権などの情報を把握や、関連する脆弱性情報と紐づけることができます。SBOMを取得する手段には、次の手段が考えられます。取引先や事業環境に応じて適切に手段を選択することが重要です。

1. ソフトウェアの提供元から入手する：
2. 手作業で解析する：
3. ソフトウェSCA (Software Composition Analysis：ソフトウェア構成解析)のツールやソリューションを利用する：
   1. SBOMで扱う管理項目  
      　　※ ソフトウェアコンポーネントを特定できることとしてminimum elements などを概説か。加えて、

* 1. SBOMのフォーマット

先に挙げたMinimum Elementsでは、SPDX、CycloneDX、SWIDのどれかを採用し、かつ、いずれのフォーマットにも相互に変換できることが必要だとしています。ここでは、オープンスタンダードのSBOMフォーマットについて紹介します。

* + 1. SPDX：
    2. SPDX Lite：SPDXのサブセットとして仕様化されています (ISO/IEC 5962及び SPDX v2.3では Annex.G)。パッケージのレベルで管理することと、Excelなどのスプレッドシートで読み書きできるフォーマットとして利用できることを想定し、SPDXのフルセットよりも取り扱う管理情報を絞ったものにしています。取引によっては、SPDX Liteまたはこれをカスタマイズしたものを利用することがあるでしょう。
    3. CycloneDX：Open Web Application Security Project（OWASP)が
  1. SBOMを管理するためのベストプラクティス
     1. SBOM管理のポリシーとプロセスを定める: ISO/IEC 5230 (OpenChain Specificartion) 、とくに§3.3 “Open source content review and approval” に照らして、SBOMの作成と管理に関して、ポリシーとプロセスを定め、文書化し、エビデンスとして管理するようにしましょう。
     2. ソフトウェアライフサイクルに応じて更新し、管理する:　ソフトウェアの構成に変更がある時、都度、SBOMを更新するようにしましょう。たとえば、ソースコードの状態、ビルド用に準備を整えている状態、ビルドしたバイナリの状態、リリースして運用後にソフトウェア更新をした場合、などで、構成が変わることがあります。SBOMは、利用している状態を反映して最新のものになっていることが重要です。
     3. SCAツール等や、構成管理システム: OSS自体の大規模化や、間接的に依存するOSSが多い場合などがあります。ツールなどを利用して効率よくSBOMを作成しましょう。ソフトウェア開発環境によってはCI/CDと連携させることも検討すると良いでしょう。また、SBOMはいつでも確認できるように、構成管理システムを用意するなど、集約しておくと良いでしょう。
     4. e-ラーニング、トレーニング: ルール、プロセス、ツール等の使い方、エビデンスの管理など、開発者及びSBOM管理の運用者にトレーニングを実施しましょう。SBOM管理が当たり前の状態となることで、コンプライアンスや脆弱性に係る問題発生の予防や、問題発生時の速やかな対処に繋がります。
     5. 法的要請への対応、業界や取引先への対応、SPDX Lite の利用：法的な要請によりSBOM及びそれと関連付けて必要となる管理情報が定まる場合があります。また、業界や取引先によっては、そうした法的要請への対応に加えて独自で必要とする管理情報を設定している場合もあります、できるだけ早めに情報収集し、必要に応じて取引先と協議し、対応を進めるようにしましょう。また、SPDX Lite が扱う情報で十分の場合もあれば、これを変更して利用する場合もあるでしょう。
     6. オープンソースコミュニティへの参加：SBOM管理については、様々なオープンソースコミュニティで情報交換がなされています。SBOMのフォーマットであるSPDXやCycloneDXはオープンスタンダードのため、それぞれのオープンソースコミュニティでその仕様策定や管理が議論されています。また、オープンソースコンプライアンスのためのツールを開発するコミュニティでもそうした活動が見られます。さらに、SBOMの相互利用性を検証するために、SBOMやツールのコミュニティが共同し、Plugfestを開催する場合があります。こうしたコミュニティに参加することで、共通の課題の発見や、コミュニティと共に課題解決に取り組むことができます。
  2. ケーススタディ、事例

経済産業省が、SBOMの運用に関する事例、

* + 1. オープンソースソフトウェアの利活用及びそのセキュリティ確保に向けた管理手法（2022年5月拡充版）  
       <https://www.meti.go.jp/press/2022/05/20220510001/20220510001.html>
    2. 参考資料　SBOM（Software Bill of Materials）の導入に関する手引（案）<https://www.meti.go.jp/shingikai/mono_info_service/sangyo_cyber/wg_seido/wg_bunyaodan/software/009.html>
  1. 発展的な話題

※NTIA, CISA, EU CRA, EO14028, 経済安全保障推進法案、などなど？

※SBOM Attestation。OpenSSF におけるSBOM関連の取組。

※各ツールの紹介？

# OSSのセキュリティリスクへの対応

* 1. はじめに

　現代のソフトウェア開発において重要な役割を果たすOSS（オープンソースソフトウェア）ですが、ライセンスに対しての理解に加え、セキュリティに対しても正しい理解をする必要があります。2021年には脆弱性が発見されたログライブラリ「Apache Log4j」が社会的な問題となりました。またNode.js (NPM) および Python (PyPI)の[パッケージリポジトリを標的とした攻撃も報告](https://jpn.nec.com/cybersecurity/blog/211217/index.html)されています。OSSの脆弱性を悪用した攻撃は増加しており、こういった背景から、企業や組織は今まで以上のOSSのセキュリティ対策を求められてきています。

OSSには同じような機能を持つものや、同じコンポーネントの異なるバージョンによって、脆弱性の数や深刻度、脆弱性の修正速度など、異なるセキュリティレベルが存在することがあります。このため、企業は使用目的に応じた適切なOSSコンポーネントとバージョンを選択する必要があります。また、OSSには元々同じコンポーネントだったものが、複数のパブリッシャーから提供されることで、セキュリティレベルが異なってくる場合もあります。な脆弱性は常に発見・報告される可能性があるため、最新情報を収集し、適切な対応策を講じる必要があります。

本章ではOSSのセキュリティリスクについて分析し、リスクに対する有効な対策を考えます。またOSSのセキュリティリスクへの理解を深めるためのレポートや有用なガイドラインも紹介します。想定読者は、開発者やシステム管理者、セキュリティ担当者、OSS推進担当者などです。OSSのセキュリティの重要性を再確認し、リスクに対処するためのヒントにしていただけると幸いです。

* 1. OSSのセキュリティリスクの種類

オープンソースソフトウェア(OSS)多くの人々が開発に携わっていますが、その質は様々です。扱うにあたってどのようなセキュリティのリスクがあるか、考えてみましょう。

1. 脆弱性のあるバージョンの採用：過去にリリースされた古いバージョンのOSSはセキュリティの問題がある可能性があります。新しい脆弱性が発見された場合、OSSプロジェクトは通常、脆弱性を修正した最新バージョンリリースします。古いバージョンのOSSを使用する場合、攻撃者は修正されていない脆弱性を悪用して攻撃することができます。さらに、古いバージョンは、脆弱なプロトコルを使用している場合があり、この点においても最新のセキュリティ基準に適合しないことがあります。
2. パッケージ・ライブラリの脆弱性：使用しようとするOSSには、それらが依存するパッケージやライブラリがある場合があります。依存関係のあるソフトウェアのバージョンが古く、脆弱性がある場合はa. で挙げたリスクと同様にアプリケーションやシステムに影響を与える可能性があります。
3. パッチの未適用： OSSプロジェクトが脆弱性に対する修正パッチをリリースした場合、Linuxディストリビューションなどのリポジトリ管理者はOSSプロジェクトがリリースした修正パッチをリポジトリへ配信します。システムの管理者は配信された修正パッチを適用しなければ、脆弱性の脅威に晒され続けます。
4. ソースコードが公開されているためのリスク：OSSのソースコードが公開されているため、悪意のある者が脆弱性を探し出し、それを悪用する可能性があることには注意が必要です。しかし、OSSは多数の人々によって監視され、改善されることが多いため、協力的なコミュニティの存在がそのリスクを軽減することができます。この点はメリットとリスクが表裏一体となっています。公開されているコードはユーザーが多いほど脆弱性が早期に発見される可能性が高くなり、また修正される可能性も高くなる傾向があります。
5. 開発プロジェクトのサポート: OSSは開発プロジェクトによって開発が行われるため、バグや脆弱性の修正対応は開発プロジェクトが行います。ユーザーコミュニティや企業などが脆弱性の対応支援を行う活動もあり、第三者から修正プログラムが提供される場合があります。しかし、活発なコミュニティでない場合や、セキュリティリスクの解決を優先しないコミュニティの場合は、問題の解決が遅れる可能性があり、プロプライエタリ・ソフトウェアよりも修正に要する平均時間が高いと報告するレポート(\*)もあります。

(\*)State of Open Source Security 2022. https://snyk.io/reports/open-source-security/

OSSプロジェクトの脆弱性修正にかかる時間が増加したとの調査結果. <https://www.conversion.co.jp/tecblog/20220805>

OSSは上述したリスクや傾向からプロプライエタリ・ソフトウェアなど非オープンソースソフトウェアと比較されることがありますが、どちらが安全か決めることはできません。どのようなソフトウェアもセキュリティに対するリスクを持ちますが、リスクを正しく評価し、適切な対策を講じることが重要です。

* 1. OSSのセキュリティリスクを発見する方法

既知の脆弱性を発見する方法として、以下の手段があります。これらの方法を組み合わせ、組織やプロジェクトの状況に応じたベストな脆弱性の収集方法を決定しましょう。

1. セキュリティリスク情報データベース：CVEやJVNなど情報セキュリテイ情報を発信するデータベースには、OSSの脆弱性を含む多くセキュリティリスク情報が掲載されています。

参照できる関連情報公開元

* 1. CVE(Common Vulnerabilities and Exposures):<https://www.cve.org/>
  2. IPA (情報処理推進機構)脆弱性対策HP:<http://www.ipa.go.jp/security/vuln/>
  3. JVN(Japan Vulnerability Notes) 脆弱性対策情報データベース:<http://jvndb.jvn.jp/index.html>
  4. JPCERT(Japan Computer Emergency Response Center) CC:<https://www.jpcert.or.jp/vh/top.html>
  5. OSV(Open Source Vulnerabilities):<https://osv.dev/>
  6. Known Exploited Vulnerabilities catalog: <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>

1. セキュリティ情報配信サービスの利用：セキュリティベンダーが発見した脆弱性情報を公開するサービスを確認することで、既知の脆弱性を知ることができます。サービスによっては付加価値として追加の脆弱性対策が案内される場合があります。
2. セキュリティグループコミュニティへの参加：企業や法人からなるセキュリティグループへの参加を通じて、OSSのセキュリティに関する動向、攻撃手口、最新の情報の入手し、関連業種の他会員と解決策を模索することができます。
3. ツールを用いた脆弱性スキャン：セキュリティベンダーが提供するリスク解析ツールや、OSSとして公開されているツールを用いてスキャンすることで、既知の脆弱性を発見できます。。
   1. OSSのセキュリティリスク対応  
      　脆弱性が発見された際はリリースされた修正パッチを迅速に適用することが重要です。

すべての脆弱性へ対応を行うことが難しい場合は、公開されているCVSS(※)の評価結果を用いて脆弱性のシステムへの影響度合いをはかり、環境への影響を踏まえたトリアージが有効です。トリアージとは組織の顧客、経営層、開発者などのステークホルダとのコミュニケーションのうえ、対応する脆弱性の優先度付を行うことです。早急な対応が必要とされる緊急時は指摘されたすべてのリスクへの対応が現実的に困難な場合があり、許容出来ないリスクを明確化することが必要です。

なお修正パッチの適用が難しい場合や、パッチ自体がリリースがされない場合は、影響のある機能を一時的に停止する対応や、セキュリティに問題のない代替となるライブラリへの置き換えを検討する必要があります。

※CVSSスコアは、脆弱性の深刻度を0~10.0の範囲で評価した指標です。  
IPA 共通脆弱性評価システムCVSS概説. <https://www.ipa.go.jp/security/vuln/CVSS.html>  
IPA共通脆弱性評価システムCVSS概説によれば、基本評価基準に加え、脆弱性が現在利用可能かどうかを評価するために使用される「現状評価基準」、脆弱性が組織の環境にどの程度影響を及ぼすかを評価するために使用される「環境評価基準」の要素を活用することで組織に応じた効率的な脆弱性対応の優先度を決定することが出来ます。指標の例として、Access Vectorは攻撃がネットワーク経由で行われるか、ローカルからの攻撃が可能かを示す指標です。これにより、攻撃可能範囲や特権が必要かなど、様々な脅威を確認することができます。脆弱性の脅威を正しく理解するためには、CVSS総合スコアにだけ注目せず各要素も考慮することが重要です。

ただし、CVSSスコアが高くなくても、すでにその脆弱性に対して攻撃が行われている場合には、早急に対応を検討する必要があります。この場合、CISA（米国の社会基盤安全保障庁）が公開している『[Known Exploited Vulnerabilities Catalog (KEV)](https://www.cisa.gov/known-exploited-vulnerabilities-catalog)』を参考にすることができます。このサイトには、既に悪用が観測されている危険な脆弱性が公開されており、こちらも参考にすることが有効です。（米国政府機関の使用システムはDue Dateまでにパッチを当てる義務があります）

* 1. OSSのセキュリティリスク対策・予防  
     　いくつかの方法で事前にOSSのセキュリティリスクへ対策・予防することが可能です。

1. 環境のバージョン最新化：使用するOSSコンポーネントを定期的に最新バージョンへ置き換え、セキュリティアップデートを適用する。
2. 定期的な脆弱性のスキャン：脆弱性スキャンツールを使用して、対応可能な既知の脆弱性を検出する。

ソフトウェアのセキュリティインシデントに対応する組織、プロセスの設計については、JPCERT/CCから日本語版が公開されている「PSIRT Services Framework」、「PSIRT Maturity Document」が非常に参考になります。このフレームワーク、モデルはOSSを使った製品にも適用できます。

PSIRT Services Framework：<https://www.jpcert.or.jp/research/psirtSF.html>

* 1. OSSのセキュリティリスクを管理するためのベストプラクティス  
     　以下にOSSのセキュリティリスクを管理するためのベストプラクティスを整理します。
     1. 正規のパッケージを利用

OSSは公式のリポジトリやパッケージマネージャーからインストールすることを推奨します。悪意のあるパッケージ登録者がいる場合、ダウンロードしたパッケージには悪意のあるアルゴリズムが内包されている可能性があります。公式の認証がされているパッケージディストリビューターからインストールすることで、安全性を確保することができます。

* + 1. サポートライフサイクルの理解  
       採用するOSSのサポートライフサイクルを理解することでソフトウェアの適切なバージョン選択、関連パッケージの採用、セキュリティアップデートの計画ができ、OSSの選択に関わるリスクを軽減することができます。サポートは最新のメジャーバージョンから過去数バージョンが対象となることがありますが、基本的には最新バージョンの採用が推奨されます。OSSプロジェクトによっては、サポートポリシーやソフトウェアライフサイクルは公式Webページに記載していることがあるため、これらの情報を確認してみましょう。

* + 1. ソフトウェアの脆弱性評価  
       脆弱性データベースや脆弱性情報サイトなどから得られるCVSSスコアを活用し、OSSの脆弱性評価を行うことで、ソフトウェアに存在する脆弱性の対応を講じることができます。
    2. 脆弱性スキャンツール

セキュリティベンダーが提供するリスク解析ツールや、OSSとして公開されているツールを用いてスキャンを活用して、脆弱性を発見することができます。脆弱性を把握、管理することで、OSSのセキュリティリスクの軽減ができます。

* + 1. 脆弱性管理ガイド、プロセスを策定する  
       OSSを使用する企業や組織は、使用するソフトウェアに対するセキュリティポリシーやガイドライン、対応プロセスを策定し、運用することでリスクを管理することができます。プロセスについてはPSIRT Services Frameworkや[[OpenChain Security Assurance Specification](https://www.openchainproject.org/security-assurance)] が参考になります。(→4.8)
    2. e-ラーニング、トレーニング  
       OSSを使用する開発者や運用者に対して、セキュリティトレーニングを実施することで、OSSの採用に際しての正しいリスクの理解、脆弱性発見後の早期に対処する能力を養うことができます。
    3. SBOMの活用  
       SBOM(Software Bill of Materials)とはソフトウェアのコンポーネント、バージョン、OSSライセンス、脆弱性などの情報を記載したドキュメントのことを指します。SBOMを活用することで前述したコンポーネントの特定工数を短縮することができ、脆弱性などOSSセキュリティリスクを軽減することが期待されています。  
       ※SBOMはOSSライセンスに関する問題に対しても有効です。SBOMはMachine Readableなファイル形式であり、一般的に使用するためには、SBOMを読み取り・分析するためのツールが必要になります。詳細は3章で説明・解説がされています。
  1. 動向、レポート、ケーススタディ
     1. [OpenSSF 2022 アニュアル レポート](https://www.linuxfoundation.jp/publications/2023/02/openssf-2022-annual-report-jp/)  
        OpenSSF（Open Source Security Foundation）が、2022年に公表した報告書です。このレポートは、OSSセキュリティ、コミュニティの動向、OpenSSFが実施する取り組みなどが掲載されています。
     2. [Synopsysオープンソース·セキュリティ＆リスク分析レポート(OSSRA)](https://www.synopsys.com/ja-jp/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html)

Synopsys社が公開しているOSSのリスク状況を分析したレポートです。M&Aに際して行われた監査の情報が基となっており、OSSのセキュリティだけではなく、ライセンスコンプライアンスの問題にも焦点が当てられています。

* + 1. [Snyk オープンソースソフトウェアにおけるセキュリティの現状（The State of Open Source Security）](https://prtimes.jp/main/html/rd/p/000000007.000092857.html)Snyk社が公開しているオープンソースセキュリティの状況を分析するレポートです。オープンソースプロジェクトの脆弱性、セキュリティトレンドなどOSSのセキュリティに関する課題についての情報を提供しています。
    2. [オープンソースソフトウェアの利活用及びそのセキュリティ確保に向けた管理手法（2022年5月拡充版）](https://www.meti.go.jp/press/2022/05/20220510001/20220510001.html)  
       経済産業省が策定したドキュメントで、オープンソースソフトウェア（OSS）を利用する企業・組織が、セキュリティリスクを軽減しながらOSSを活用する事例を事例集として提供しています。各企業がOSSを選定・評価する際のポイント、OSSの導入・運用・保守における適切な手順、またOSSのライセンスに関する考慮事項などが説明されています。
    3. [OpenChain Security Assurance Specification](https://www.openchainproject.org/security-assurance)  
       OpenChainが定義する、サプライチェーンにおけるソフトウェア品質、セキュリティ、およびコンプライアンス確保のための標準です。この標準は企業がリスク管理の効率化と信頼性の高いソフトウェア提供を目指すことを支援し、コンプライアンス追跡の簡素化や品質向上にも寄与します。

* 1. OSSセキュリティに役立てられるリソース（オンラインコミュニティ、トレーニングなど）
     1. [CIS Software Supply Chain Security Guide](https://www.cisecurity.org/insights/white-papers/cis-software-supply-chain-security-guide)  
        米国の非営利組織であるCIS（Center for Internet Security）が作成したガイドです。  
        ソフトウェアサプライチェーンのセキュリティに関するベストプラクティスを提供することを目的としています。

このガイドは、ソフトウェアサプライチェーンにおけるセキュリティの重要性に焦点を当て、ソフトウェア開発者、サプライヤー、顧客などの関係者に向けた実践的なアドバイス、また内容としてはサプライチェーン攻撃のリスクを軽減するための戦略や、セキュリティのためのフレームワーク、技術的なガイダンスが含まれています。

* + 1. [SLSA（Supply chain Levels for Software Artifacts）](https://slsa.dev/)  
       SLSAは、ソフトウェアサプライチェーンのセキュリティに関するフレームワークの１つです。Googleが開発し、2021年に公開されました。SLSAは、ソフトウェアサプライチェーン内での信頼性を高め、サプライチェーン攻撃から保護するためのベストプラクティスを提供しｒており、OSSの利用者や開発者が、ソフトウェアサプライチェーンの信頼性を高めるための指針としても利用できます。  
       SLSA Whitepaper: SLSAの概要や目的、各レベルの説明、実装例などが記載されています。  
       SLSA Implementation Guide: SLSAの実装についてのガイドです。SLSAを実装する際に参考にできます。  
       SLSA Scorecard: SLSAの各レベルに対する評価ツールです。自社のOSSのSLSAレベルを評価する際に利用できます。
    2. [PSIRT Services Framework 、PSIRT Maturity Document](https://www.jpcert.or.jp/research/psirtSF.html)  
       (4.4) OSSのセキュリティリスク対策・予防でも紹介したPSIRT Services Frameworkは、製品のセキュリティインシデントを処理するためのPSIRTチームのための指針を提供するフレームワークです。OSSを活用する製品においても、このフレームワークは役立てられます。一方でPSIRT Maturity Document は企業や組織のPSIRTの構成、プロセスの成熟度を評価する情報が提供されています。これらはJPCERT/CCから日本語版が公開されています。
  1. オープンソースソフトウェアのセキュリティに関連する実践的なヒント、Tipsなど
     1. 発見したOSSの脆弱性にどう対処すべきか。  
        OSSの脆弱性が発見された場合は、速やかにOSSの開発者やメンテナーに報告することが大切です。報告先について、OSSプロジェクトのWebサイトやリポジトリに報告に関するポリシーや手順が掲載されている場合があります。脆弱性が発見された場合は各OSSプロジェクトの方針を確認しましょう。

報告先が不明な場合は、[Zeroday Initiative](https://www.zerodayinitiative.com/advisories/disclosure_policy/)へ報告することができます。Zeroday InitiativeはTrend Microが運営する脆弱性情報の収集・報告・研究を行うプログラムです。報告された情報を基に、パッチが公開される前に脆弱性を修正するための情報をベンダーに提供します。

* 1. その他

改訂歴

|  |  |  |
| --- | --- | --- |
| No. | 改訂日(Version) | 改訂内容 |
| 1 | (V1.0) |  |
|  |  |  |