

OSS Educational Materials (Supply Chain Risk Management Version)

This material is a reference material for each company to learn about license investigation, license obligation fulfillment, SBOM, and security risk response when conducting OSS supply chain risk management. Please feel free to copy and modify it as a material for each company to create educational content.

This material is released under the [Creative Commons CC0 1.0 Universal](http://creativecommons.org/publicdomain/zero/1.0/legalcode) License with no restrictions on copying, modifying or distributing it.

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

Disclaimer

* This document assumes that OSS is used in Japan, and is based on the experience of members of education sg. Please consult an expert if you have difficulty making a decision, such as strict legal interpretation or overseas use.
* Please be aware that neither the author nor the supplier assumes any responsibility for the contents of this document.
* This document does not guarantee OpenChain's specification compliance or certification.

Content

1. Investigating OSS Licensing

2. Fulfilling Licensing Obligations

3. SBOM

4. Addressing OSS Security Risks

[1 OSS License Investigation](#_7x16ojmz3axp) [2](#_7x16ojmz3axp)

[2 Performance of License Obligations](#_sihqp3bqdr39) [8](#_sihqp3bqdr39)

[3 SBOM](#_brns4glw48x0) [16](#_brns4glw48x0)

[4 Response to OSS Security Risks](#_4dqwmz9clk5h) [18](#_4dqwmz9clk5h)

# Investigating OSS Licenses

* 1. Introductory Q & A Let's think about this for a moment. We use OSS A ourselves. The website on the Internet says "License X for OSS A." If I distribute version 2 of OSS, can I distribute it under license X? The answer is that you must check the version 2 license that you use and distribute it under the corresponding license.

In the following sections, you will learn about license research.

* 1. License research When you use OSS, you must conduct license research and comply with the license at the time of use. When selecting OSS, it is necessary to first examine the download files and then examine the OSS licenses comprehensively. The following is an example of license examination.

(a)First, download the applicable OSS. Before downloading, if it says "Follow this license when downloading or using," you need to follow that license, so check the information as license information.

(b)If the file downloaded in (a) is an archived file (Files with extensions such as zip/jar/tar/tar.gz/tgz/xz/deb/rpm), unzip it.

(c)If the file downloaded in (a) is an executable or Windows Installer format file (Files with extensions exe and msi), install it. Start the installer and check the license text and the software license agreement as license information.

(d)Search for files whose names include the following characters: Check whether license information is included in the searched file.license/copying/copyright/eula/about.html/pom/manifest-readme/notice/legal/license (\*), etc.

(e)The contents of all files in the download file are searched for the following strings: The files containing these strings are checked for license information. license/epl/gpl/asl/artistic/legal/notice/third/license (\*), etc.

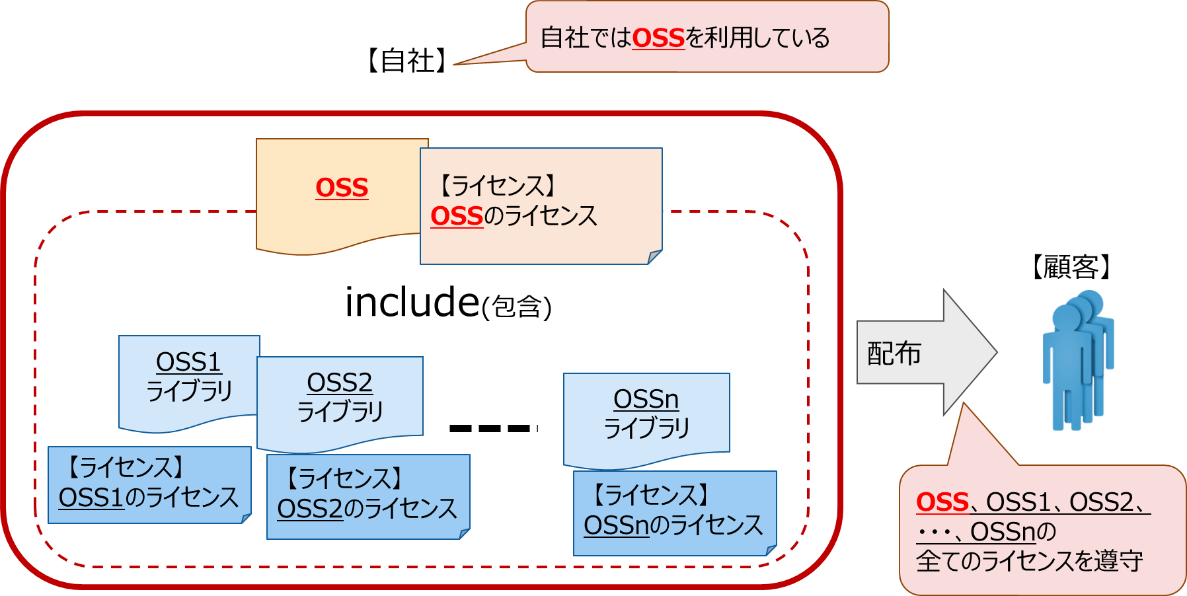
(f)For the file archived in (b), check whether the definition file such as MANIFEST.MF/pom.xml in the META INF folder in the archived file contains license information.

(g)If the download site is different from the official site, the project site, or the vendor site, check the following: Look for the License/Licensing/Readme/Copying/Copyright/Eula/Legal/about.html/notice/Thirdparty/Thirdparties/License (\*) string on the official site or the vendor site, and examine the license body and the link to the license.

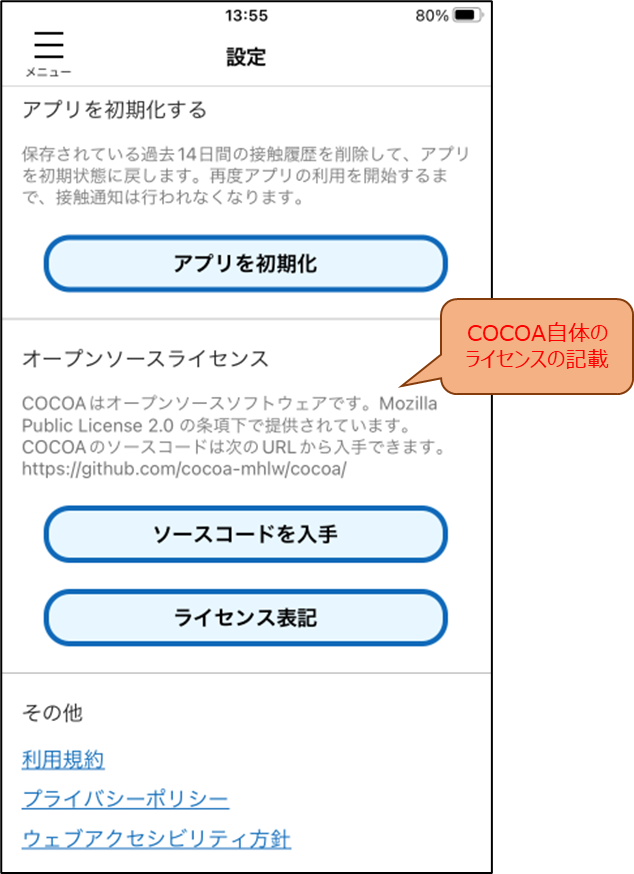
The basic task in (a) through (f) of the above example is to check the license information in the source code file comprehensively. (g)It is better to consider the task in (a) through (f) as a supplementary task. If the license investigated in (a) through (f) differs from the license investigated in (g), it is better to clarify the reason why the former license differs from the latter license, determine which one to choose for that reason, and select one to use.

Even when using OSS object code or load modules, it is important to download the corresponding source code and investigate the license. Therefore, you need to make sure that the version of the object code or load module matches the version of the source code. In addition, the load module may have additional libraries. You also need to check the licenses of the additional libraries.

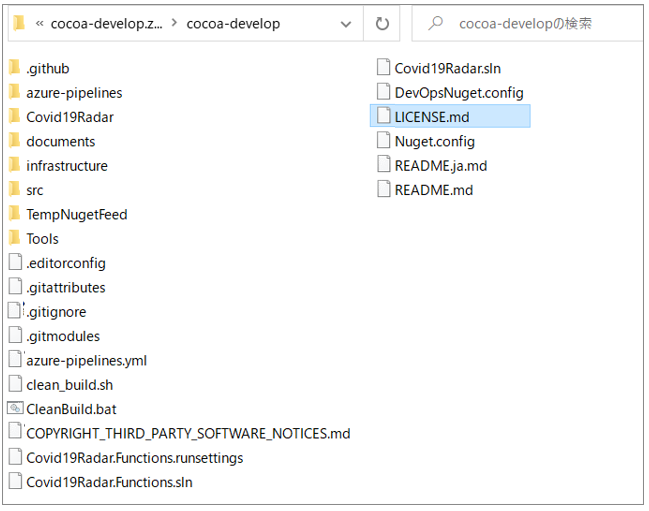
If you cannot find the license information in the above survey, you will need to check with the community developers. If the license information is still not clear, the OSS should be disabled. (\*: license may be used in the same sense as license.)

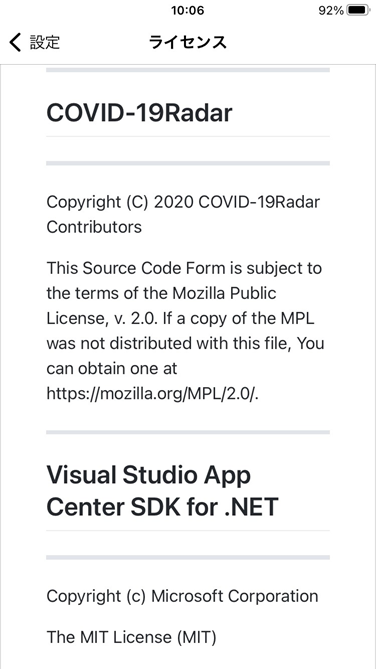
* 1. Check licenses for libraries, etc. In recent years, OSS has also been increasing in scale (especially for operating systems, frameworks, etc.), and many of them include libraries of other OSS. Therefore, it is necessary to investigate the licenses of Section 1.2, and investigate the libraries of other OSS included in the OSS and their licenses comprehensively.

When using container images, it is also necessary to investigate the OSS and libraries of OSS included in the container and their licenses.

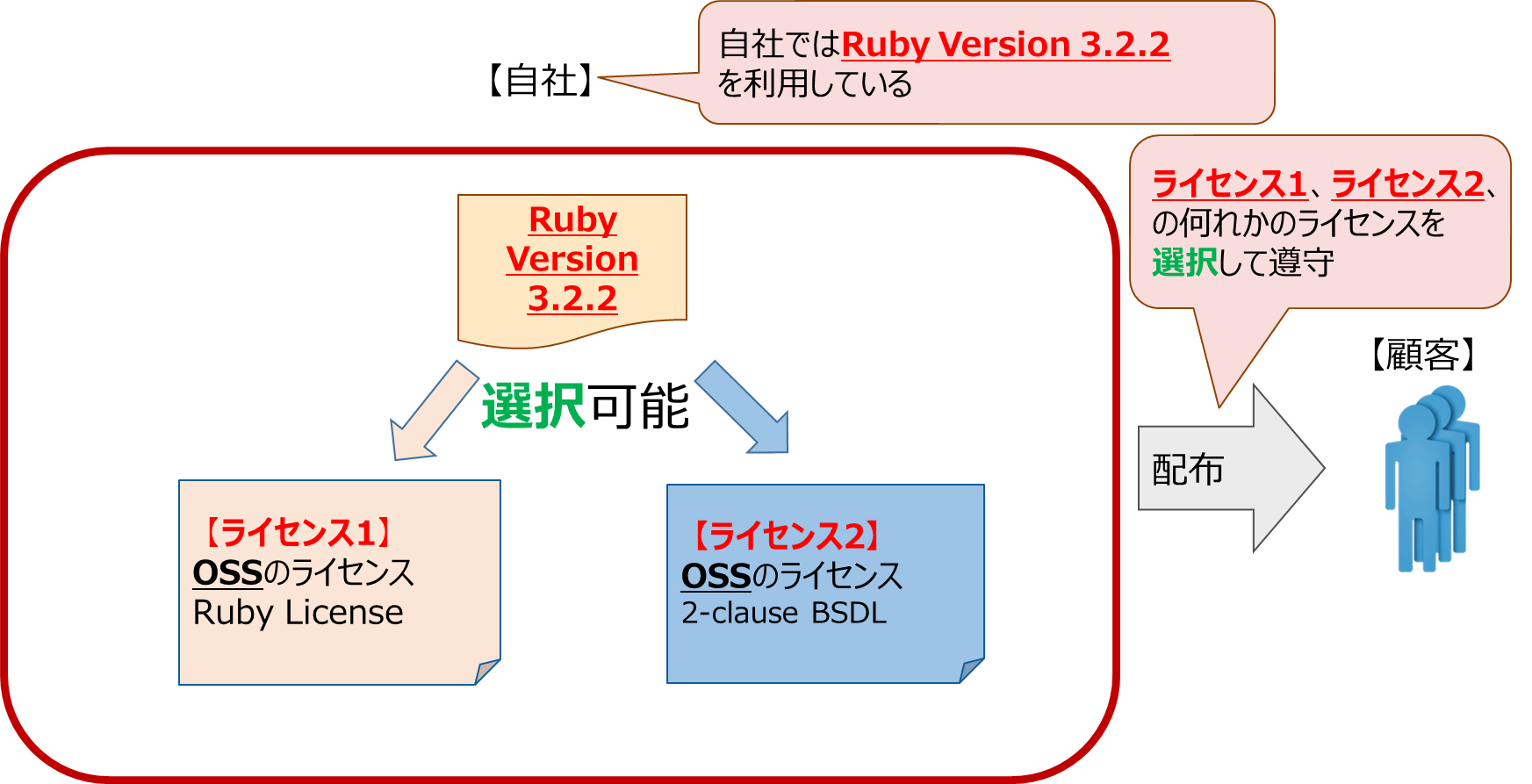
* 1. Examples of licenses
     1. When OSS is used in object code and load modules, for example, the license may be written in the application or program to be distributed. The following is an example of a COCOA application.
     2. If you use OSS in source code, you may find a license in the files.

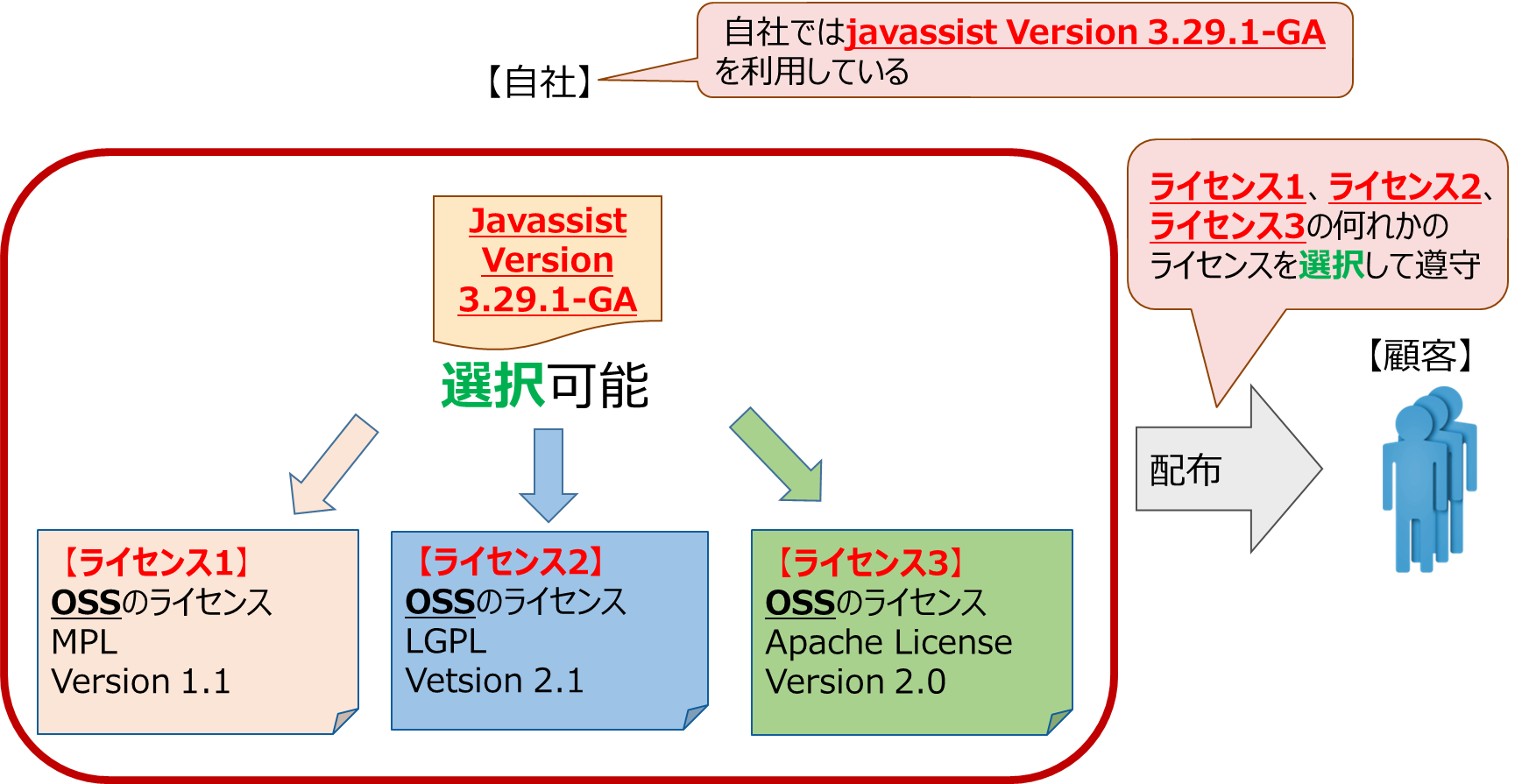
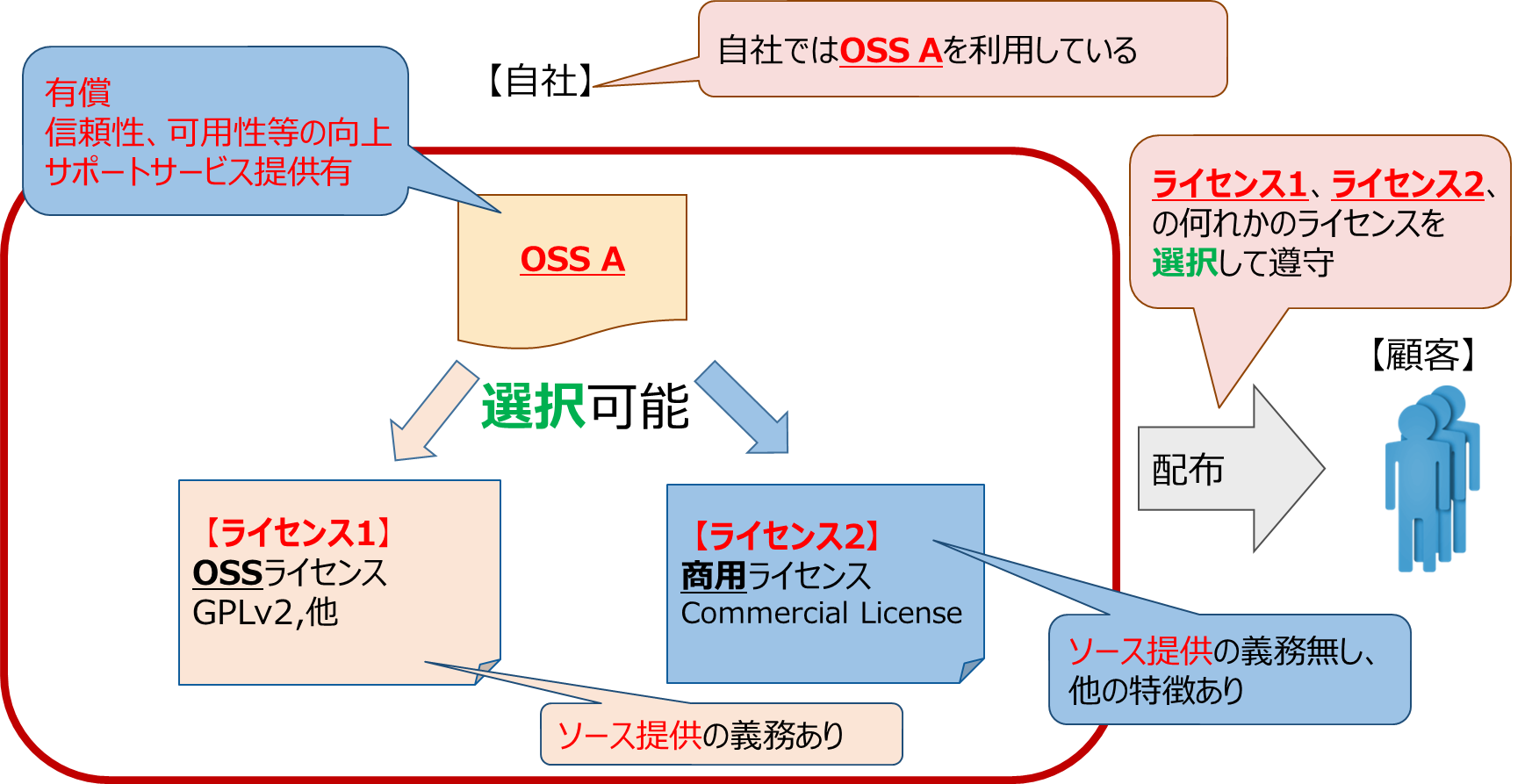
The following is an example of displaying a license on the website of the coronavirus contact confirmation application "COCOA." (Light blue frame in figure [https://github.com/cocoa-mhlw/cocoa).](https://github.com/cocoa-mhlw/cocoa%EF%BC%89%E3%80%82)



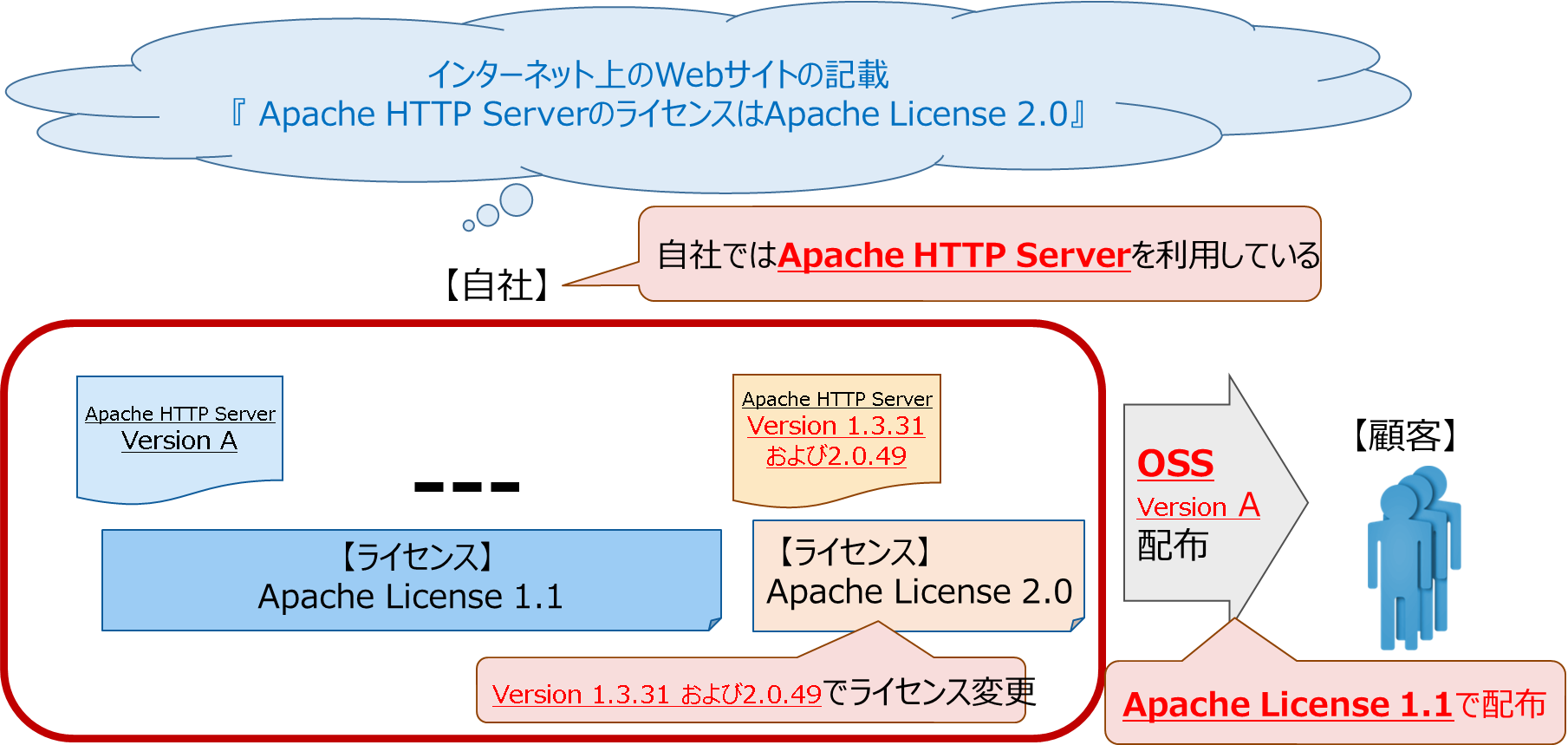
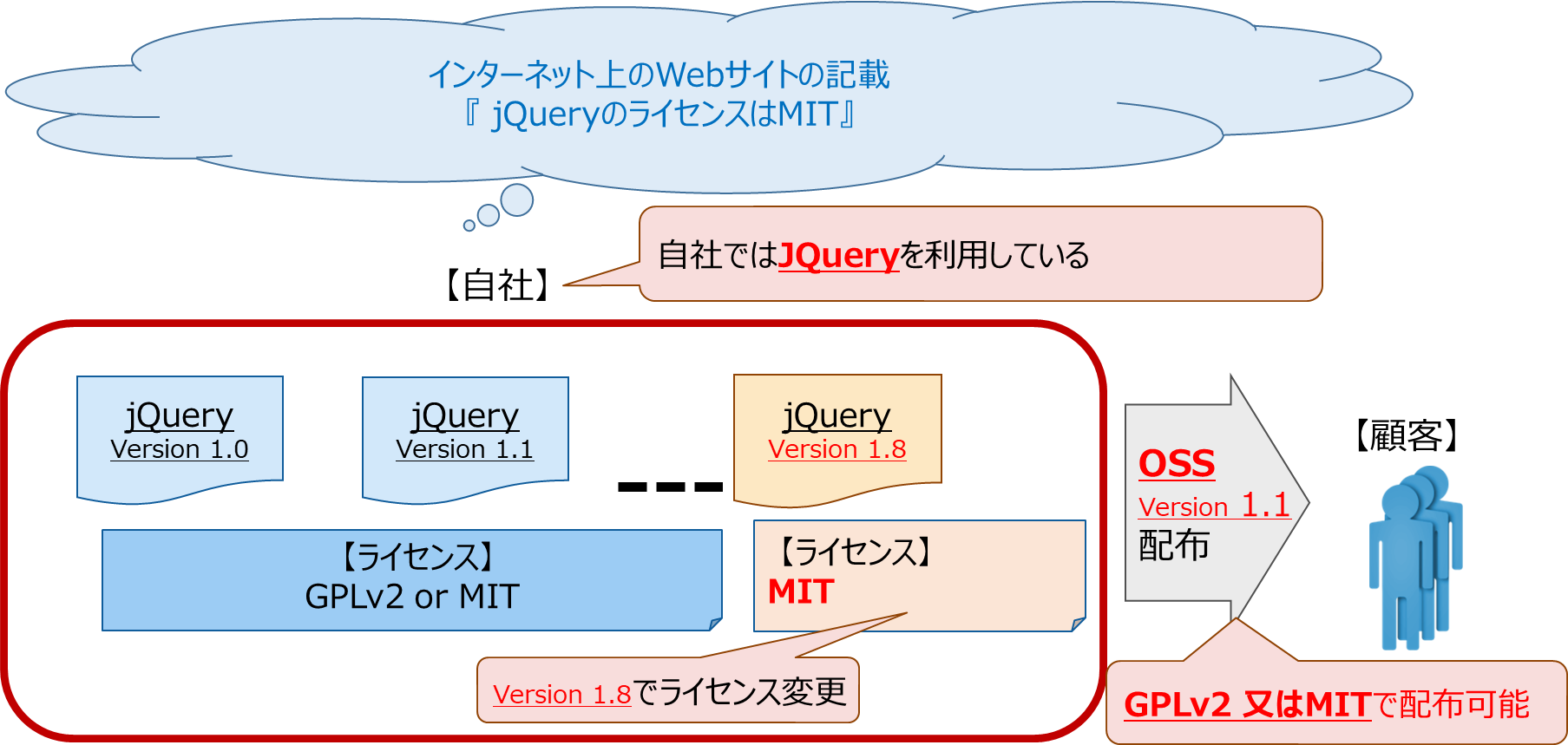


* 1. Cases where multiple licenses are listed Some OSS allows users to select a desired license from among multiple licenses. A license that can be selected from two licenses is called a dual license, and a license that can be selected from three licenses is called a triple license, or a multiple license.
     1. An example of a dual license is Ruby. See the following URL: <https://www.ruby-lang.org/en/about/license.txt>



* + 1. An example of a triple license is javassist. Please refer to the following URL: <http://www.javassist.org/>  
         
       
    2. In some cases, you can choose between commercial and OSS licenses. For example, a software development vendor might add features to OSS developed by the OSS community to improve reliability, availability, and so on, and then provide them under a commercial license. In this case, it is generally provided for a fee, and the software development vendor might provide support services. In another example, you might be able to choose between an OSS license with a source obligation and a commercial license without a source obligation.  
       
  1. Misleading Web-Based Licenses (Version)

Depending on the version of OSS, the license may vary. Web sites on the Internet often list the latest version of the license, and if you determine that the same license applies to an older version of OSS, you may distribute OSS under the wrong license. Be sure to check the license of the version you want to distribute.

* + 1. Example 1: When Apache HTTP Server became versions 1.3.31 and 2.0.49, the Apache License was also 1.1⇒2.0.
    2. Example 2: Prior to version 1.8, jQuery was a dual license with a choice between GPLv2 and MIT, but in version 1.8 it was unified to MIT.
  1. OSS, Listing Licenses
     1. Researching and verifying OSS licenses, such as 1.2~1.6, is also important for fulfilling the licensing obligations when using OSS. It is also necessary to create the SBOM in Chapter 3. Therefore, it is a good idea to summarize the results of the license investigation by making a list of the following items. (1)OSS name (2) OSS version (3) OSS download URL (4) License name (5) Copyright information
     2. By listing OSS and licenses, you can make very effective use of OSS compliance when reviewing and approving it internally.

# fulfillment of licensing obligations

* 1. Introductory Q & A In this chapter, let's first look at OSS-based product development and work to comply with OSS licensing through the following examples.

Company A has decided to include OSS "X" in its products under development. X is a very famous OSS in a certain field, and everyone knows its name, characteristics and developers. Therefore, Company A decided to add a single word "We use OSS" in part of the product documentation, and omit other detailed expressions. Can we say that Company A's response at this time was sufficient to comply with the OSS license?

Unfortunately, the story is not so simple, and depending on the license conditions of OSS "X," it may be considered as a violation of the license, so we cannot say that it was "fully complied with." To avoid this situation, you need to first understand the OSS license and comply with its conditions. This chapter reviews typical OSS license compliance tasks and their points, using embedded system products such as TVs and smartphones as examples. [\*]

* 1. Overview of License Compliance Tasks The following three points are particularly representative of OSS license compliance tasks. Section 2.3 and subsequent sections describe specific tasks.
* Inclusion of mandatory wording (e.g., license documents) (Section 2.3)
* Provision of source code (Section 2.4)
* Addendum to the contract attached to the product (e.g. end-user license) (Section 2.5) [\*] [\*] [\*]

In addition, for the convenience of explanation, the term "OSS license level" is used to categorize the strength of restriction of license conditions into 5 levels. For details on each level, refer to "OSS education materials, supply chain risk management version licensing basis" Chapter 2.

Table X. License condition constraint strength (5 levels)

|  |  |
| --- | --- |
| Level | Overview of License Terms |
| Level 1 | Attach license documents when distributing OSS |
| Level 2 | When OSS is distributed, in addition to Level 1, copyright-related information is included in the document. |
| Level 3 | Provide source code for target OSS in addition to level 2 during OSS distribution |
| Level 4 | At the time of OSS distribution, in addition to Level 3, the entire source code that combines the target OSS and other software into a single copyrighted work is provided. |
| Level 5 | In addition to Level 4, in the case of service usage (SaaS/ASP, etc.), the same source code as Level 4 is provided to the server accessors. |

* 1. Listing mandatory language (such as license documents) Most OSS licenses also require recipients to provide basic OSS information, such as OSS copyright information, license text, and disclaimers. Some OSS licenses require only the license text to be included in the product documentation, some require an acknowledgment in addition to the license text, and some require an acknowledgment to be included in advertising media.

The following are examples of the primary methods of inclusion. Specific terms and conditions may be detailed in the license, so it is important to check the text of each license when considering it.

* Include in documentation such as product manuals and end-user licenses
* (if the product itself has an information display function) Display in user functions such as online help and option menus
* (Providing the source code to a third party in HTML format) Leave the copyright notice pre-embedded in the source code without removing it

One caveat is that the posting location should be "the natural place for OSS recipients."

For example, "Store only OSS copyright information and license text on a dedicated CD-ROM without any guidance, even though the user manual and end-user license are paper booklets" is not a good idea. Unless the OSS recipient is intentionally trying to "find out what OSS information is in the product," they won't see it. If the document is presented separately from other product documents, consider adding a document that clearly guides the OSS recipient.

Following is a description of the license text, acknowledgments, and acknowledgments in advertising.

1. Include license text in product documentation OSS license text is often stored in text files named "COPYING" or "LICENSE." The following is a template for MIT LICENSE, a Level 1 OSS license. Copyright information, license terms, and disclaimers are listed from top to bottom.
   * Many of the license terms contain copyright information, license terms (Permitted acts and restrictions, conditions to be observed), and disclaimers in English, which can usually be provided directly to OSS recipients to satisfy the license terms. [\*]
   * It is important to **keep the license terms unaltered** when posting them. Be careful not to change the license terms, and to post the original English text without Japanese translation or interpretation.

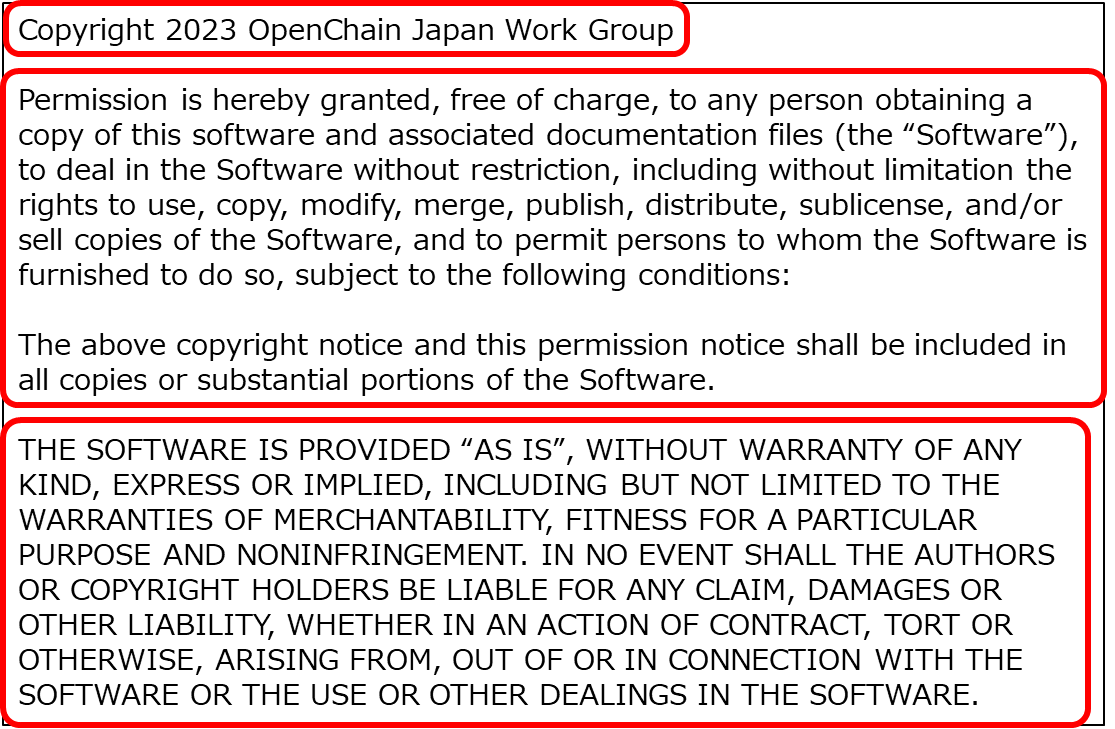


Fig X. Level 1 OSS License (MIT LICENSE)

1. Post Acknowledgements Some Level 2 OSS licenses require you to post Acknowledgements. In many cases, an Acknowledgement will be a sentence to the effect of "This software contains artifacts from Development Project A," and will carry the Acknowledgements as they appear in the license text.

Below is an example of a license that requires an Acknowledgement to be posted, and a software that actually posted an Acknowledgement under that license. The Libjpeg License requires an Acknowledgement (”this software is based in part on the work of the Independent JPEG Group”) to be posted in the product documentation, while Firefox, a web browser that uses the Libjpeg License OSS, posts an Acknowledgement in Other Required Notices.

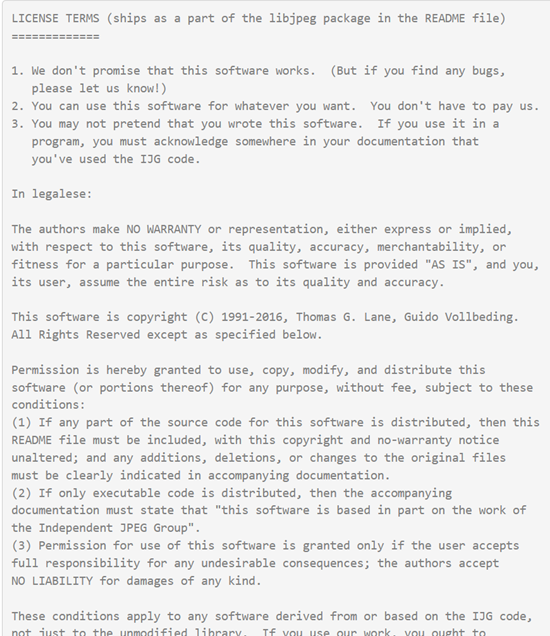


Fig X. Example license requesting acknowledgment (Libjpeg License, [https://jpegclub.org/reference/libjpeg-license/)](https://jpegclub.org/reference/libjpeg-license/%EF%BC%89)



Fig X. Example of acknowledgment (Firefox, about: license#other-notices)

1. Posting acknowledgments in advertising media Some Level 2 OSS licenses include a provision that requires acknowledgments to also be posted in all advertising media that mention the functionality or use of this software. For example, if you write about OSS features on a promotional Web page for an OSS-based product, you must also include an acknowledgment that the license requires.

The following is an example of a license that requires an acknowledgment in an advertising medium and software that uses that license to include an acknowledgment in an advertising medium. Because OSS features are mentioned on the product introduction page of the company's website, a sentence of acknowledgment (This product includes software developed by the University of California, Berkeley and its contributors.) is included.

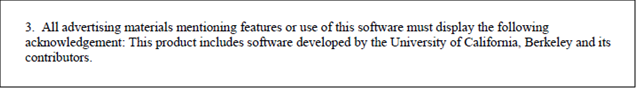


Fig X. Example of a license agreement that requires an acknowledgment to be posted in an advertising medium (BSD 4-Clause Original (Old) License)

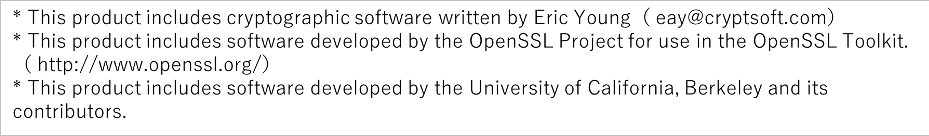


Fig X. Example of a product introduction page on a company's website for a communications embedded product

* 1. Providing source code When OSS is included in a product in the form of binary code, there is a license that requires OSS recipients to provide the source code and related information as well. This is a condition that is sensitive to the recipient, for example, to preserve technical options when "Recipient modifies OSS and replaces OSS in the product with a modified version."

Be sure to check the text, especially since some licenses require you to provide information related to the source code, including configuration information to fully reproduce the OSS binary code in the product and configuration information to run the binary code.

* Level 3 OSS licenses provide the source code for OSS itself. If you modify OSS, you provide the modified source code.
* Level 4 OSS licenses provide level 3 support, as well as source code for other programs defined by the OSS license.
  + 1. Preparing the Source Code You provide the complete source code required by the license. Some licenses require that you provide not only the source code, but also all the information necessary to reproduce and execute the OSS binary code that you include in the product.
* If your license requires that you also provide information to reproduce and execute the OSS binary code that is included in the product, then you should provide the information at the time the binary code was generated, in addition to the source code, so that the binary code can be reproduced properly. Compilation options and various configuration parameters (generally a set of make files) are equivalent. Note that you do not need to provide an "environment for reproducing binary code" per se, such as the compiler you used.
* If you do provide modified source code, you should pay attention to general coding rules and etiquette, such as announcing the changes. However, some licenses have conditions on the notification method, so you should check the license.
* Avoid intentionally preventing the reproduction and execution of binary code, such as distributing source code on paper. This may violate the license terms. Also, avoid violating copyrights, for example, by replacing someone else's original OSS copyright notice with your own.
  + 1. Choosing how to provide source code Some OSS licenses offer multiple options for providing source code. In such cases, you may want to choose the most appropriate delivery method and proceed.

On the other hand, there are OSS licenses that limit how source code is delivered. Therefore, you should review the license terms and work with your department (Examples: Legal, IP, OSPO, etc.) if necessary.

This section provides an example of applying Level 3 or higher OSS, which requires the provision of source code, to embedded system products. There are three typical ways to provide source code:

Method 1. Include the OSS binary code to be included in the product, as well as a complete set of required source code

Method 2. Include documentation that a set of source code will be provided at the request of the recipient

Method 3. Provide a set of source code on the website and include a document containing the URL, etc.

1. Include the required set of source code along with the OSS binary code to be included in the product. As shown in the figure below, you can ship both directly to the product itself or the electronic media of the product.

* Because the binary code and source code are included when the product is shipped, you have the advantage of ensuring that accurate information is included at that time. In addition, there is no need to establish or maintain a system for providing source code, as compared to Method 2 and Method 3 described below.
* On the other hand, it is important to note that when upgrading the product itself, it is necessary to make sure that the source code that matches the OSS binary code included in the upgraded version is included in the package. There is also the cost of adding additional media in some cases because it takes up more space on the product itself and on the product documentation.
* Before the product is released, make sure that the complete source code required is properly included and that the final shipment is available. For example, the development department may not be able to properly provide the source code corresponding to the binary code to the downstream department in the process leading up to the product shipment, or the shipping department may decide to exclude the source code from the shipment. To avoid omissions, the departments in charge of each process should work closely together.

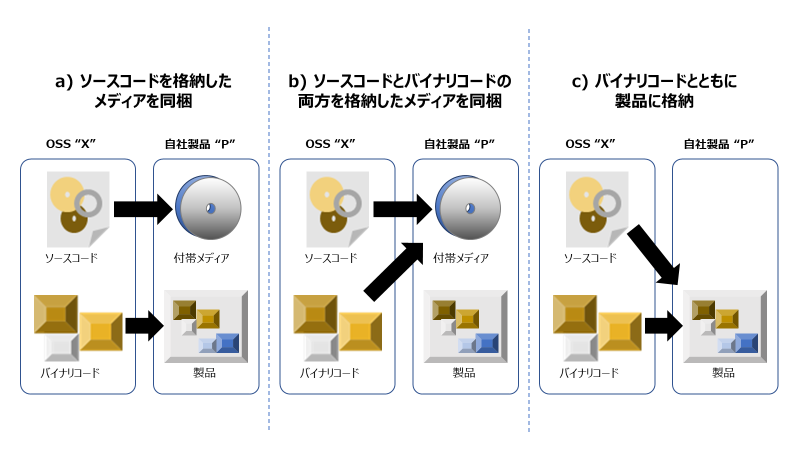


Fig X. Three forms of shipping source code [\*] [\*]

1. Packing a document stating that a complete set of source code is to be provided in response to a recipient's request A method of shipping a document stating that "Source code is to be provided in response to a recipient's request" and instructions on how to make the request (e.g., a contact) to the product itself or product documentation (paper or electronic media). In this case, based on the instructions, the recipient requests the product developer to provide the source code of the OSS included in the product, and the source code is to be provided accordingly.

* There is an advantage in terms of cost, because the recipient only responds to the request. This is because not all recipients may request it, and the cost of shipping the product can be externalized. On the other hand, it is necessary to establish and maintain a system that can respond promptly to requests from recipients.
* As long as the recipient sees the product naturally in the process of using the product, the place where the information is posted can be in print or electronic media. If there is an instruction manual or an end-user license that accompanies the product, it can be in there, or if the product has some display function, it can be in online help.
* Some OSS licenses mention the availability period of the source code (e.g., valid for at least 3 years after the product including OSS is shipped).
* When you provide a medium containing the source code (e.g., CD-ROM, DVD, etc.), you can also charge the actual cost of shipping it. Be sure to set a reasonable price for shipping and media costs.
* It's a good idea to have a separate set of source code for each version of your product so you don't accidentally provide a different set of source code.
* You don't have to have a medium in the first place. You can prepare it in advance or create it for the first time when the source code is requested.
* The entire source code can be provided in a compressed format such as tgz or zip, as long as it can be reproduced in bulk using the make command.
* Be sure to review the steps in advance with the department (e.g., user support department) that may be contacted by the user. Even if you include a point of contact in the instructions that ship with the product, you may receive an unexpected inquiry. It is a good idea to check with the relevant department to ensure that it is escalated to the appropriate point of contact.

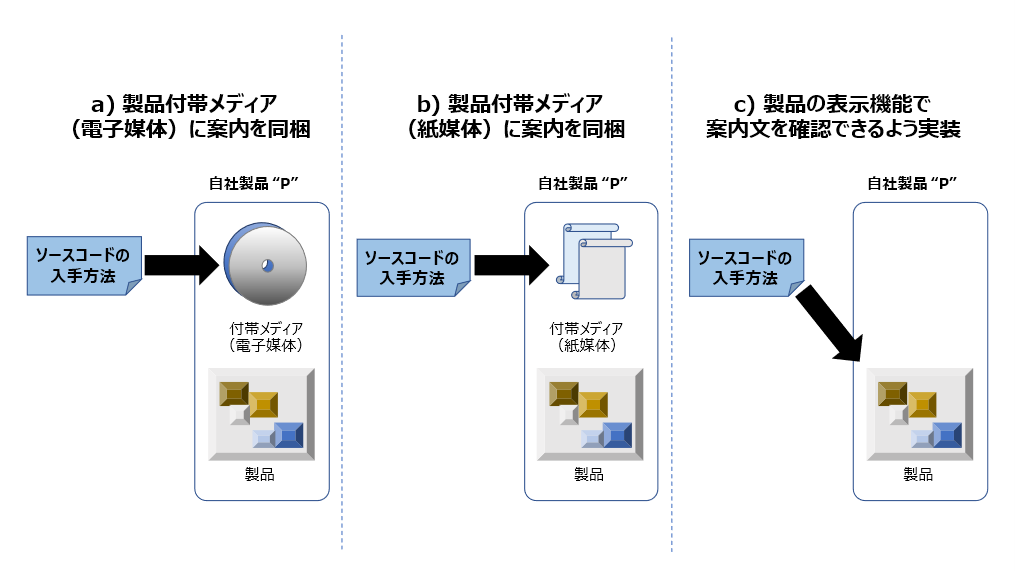


Fig X. Include documentation to provide a set of source code

1. Include documentation to provide a set of source code on the website

Include information about providing a set of source code on the website and how to obtain it (Example: URL, Other Information Required for Access) in the product itself or in the product documentation (paper or electronic).

* As the recipient can freely download it on the website, it is expected to reduce the time lag and cost of sending and receiving. On the other hand, of course, because of the demand for website operation know-how, additional investment such as internal structure construction and infrastructure development may be required. Also, it is better to have a contact number in case something happens, such as not being able to download.
* The location of the information can be in paper or electronic media, as long as the recipient sees it naturally in the course of using the product. It can be in the user's manual or end-user license that accompanies the product, or it can be in the form of a label if the product has some specific features. [\*] [\*]
* Some OSS licenses mention how long the source code will be available (e.g., valid for at least three years after the product ships, including OSS).
* The source code is not provided by the OSS development community, but by the product developer who provides the binary code. If the recipient is directed to a Web site that is not under the control of the product developer as the source code provider, the subsequent unintentional closure or relocation of the site may result in the failure to satisfy the "source code" licensing condition. Ensure that the product developer maintains the version of OSS that contains the product and has a system in place to ensure that the OSS recipient is properly served.
* Provide source code while maintaining the folder structure. Avoid excessive file splitting and distribution of destinations to avoid being perceived as "intentionally preventing the reproduction and execution of binary code."
  + You do not need to be able to view the source code verbatim in a web browser.
  + If you can reproduce the source code in bulk using a make command, you can provide it in compressed format such as tgz or zip.
* If you change the OSS included in the product, don't forget to update the source code provided on the website.
  + If you replace or modify the latest version of OSS, replace the source code provided on the web to match the OSS included in the product.
  + However, if the current version of the product remains in the market as well as the previous version, you must continue to provide the source code for the previous version on the web as well as the latest version.
* If you do not specifically restrict access to the website, it will be available for download to anyone other than the OSS recipient. Be sure to check the information that is posted. [\*]
* The figure below shows an example of the source code provided by the novel coronavirus contact confirmation app "COCOA," which provided the OSS source code on its website.
  + "COCOA" was developed to prevent the spread of the novel coronavirus and is an application that can run on smartphones.
  + Since "COCOA" was developed using "COVID-19 Radar," which is OSS, the source code of "COVID-19 Radar" is provided in the source repository of "COCOA" according to the license of "COVID-19 Radar" (Red frame in figure [https://github.com/cocoa-mhlw/cocoa).](https://github.com/cocoa-mhlw/cocoa%EF%BC%89%E3%80%82) [\*] [\*] [\*] [\*]

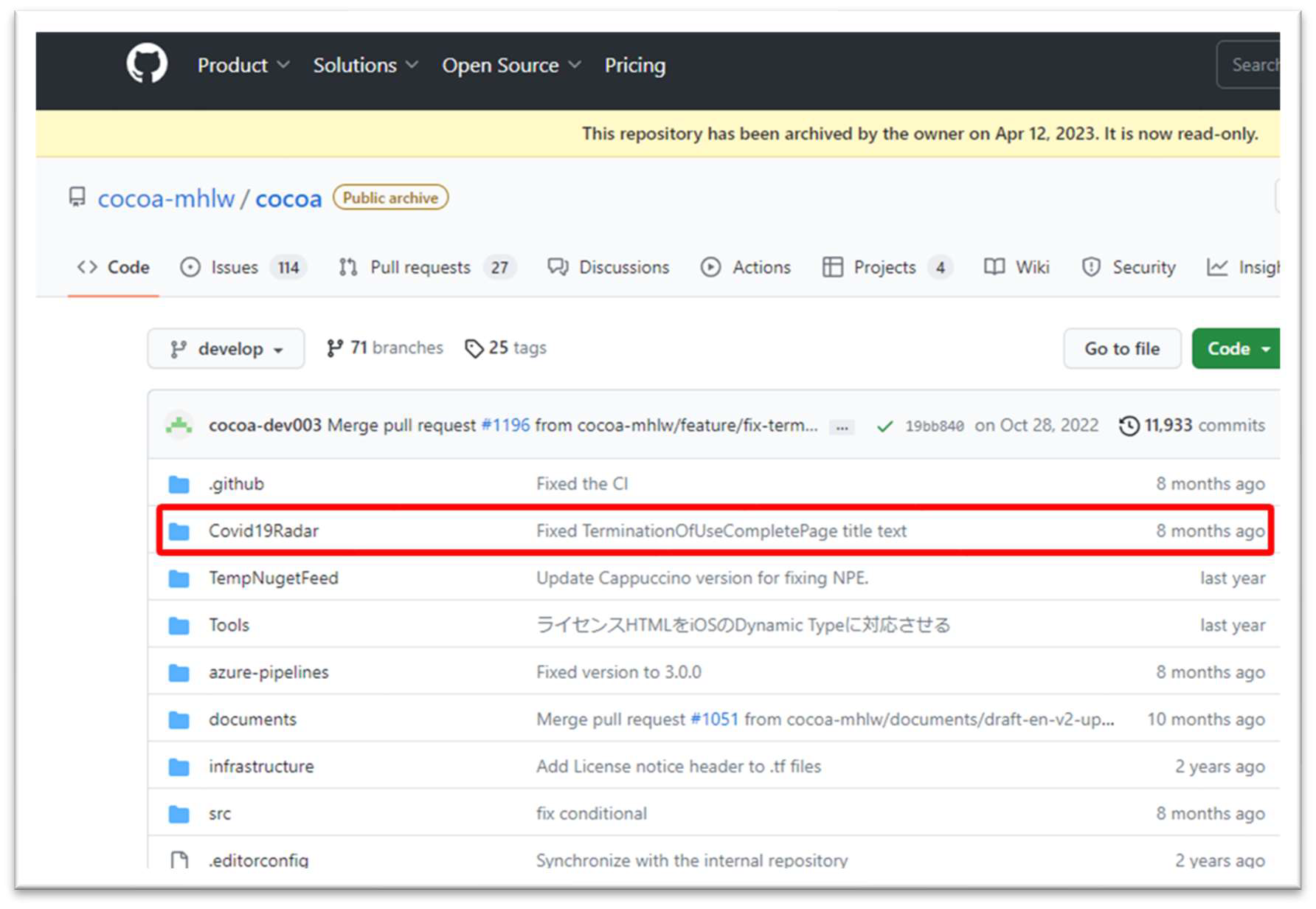


Fig X. Source repository of the novel coronavirus contact confirmation app "COCOA" (Source code of OSS "COVID-19 Radar" is published in the red box)

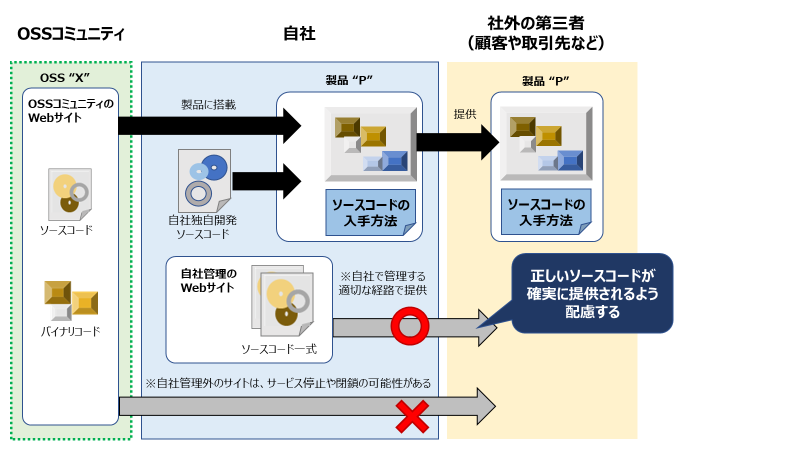
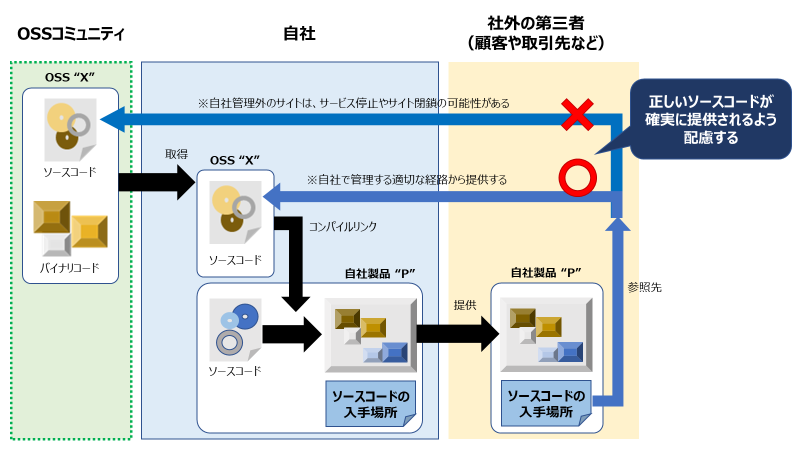


Fig X. Source code provider is a binary code provider [\*] [\*] [\*] [\*] [\*]



\*Editing comments (leave unmodified image for reference comparison)

* 1. Addendum to the contract accompanying the product (end-user license, etc.) When a product is sold, the end-user license (End-User License Agreement, EULA) may be presented as a contract specifying the terms and conditions of use of the product, and the buyer may be asked to agree to it.  
     This is a general measure to prevent unintentional infringement, accidents, or damage by both parties by specifying the attribution of rights and the terms and conditions of use of the product in advance.

However, when OSS is incorporated into a product, there may be a conflict between the terms of the traditional end-user license and the terms of the OSS license. For example, an OSS license may include conditions such as "No additional restrictions may be imposed outside the terms of the OSS license" or "Allow reverse engineering of OSS-linked programs."

In such cases, the end-user license must state "For OSS, OSS license terms take precedence over end-user license terms." In this case, the terms of the OSS license apply to the conflicting terms, and both the end-user license and the OSS license apply to the rest.

Therefore, when incorporating OSS into a product, it is important to work with the department responsible for the end-user license and the legal department to ensure that the terms of the end-user license do not conflict with the terms of the OSS license.

# SBOM

* 1. Know the software configuration

So far we have seen that license compliance is important in Open Source Software (OSS).

1. The OSS you intend to use:
2. If the OSS contains non-developer OSS: OS
3. If the OSS uses other OSS: O
   1. How to get the SBOM

With an SBOM, you can keep track of your software's licenses, copyrights, and other information, and link it to relevant vulnerability information. Possible ways to obtain an SBOM include: It is important to select the appropriate means for your business partners and business environment.

1. Obtain software from:
2. Analyze manually:
3. Use Software Composition Analysis (SCA) tools and solutions:
   1. SBOM management items \*Do you outline minimum elements, etc. as software components can be identified? In addition:

* 1. SBOM Format

The Minimum Elements mentioned above requires that you use SPDX, CycloneDX, or SWID, and that you can convert to either format. This section introduces the open standard SBOM format.

* + 1. SPDX:
    2. SPDX Lite: Specified as a subset of SPDX (ISO/IEC 5962 and SPDX v2.3, Annex. G). It is intended to be managed at the package level and can be used as a read-write format for spreadsheets such as Excel, so it handles less management information than the full set of SPDX. Depending on the transaction, you may use SPDX Lite or a customized version of it.
    3. CycloneDX: Open Web Application Security Project (OWASP)
  1. Best Practices for Managing SBOMs
     1. Define policies and processes for managing SBOMs: In light of ISO/IEC 5230 (OpenChain Specification), particularly §3.3 “Open source content review and approval, ” define, document, and evidence-manage policies and processes for creating and managing SBOMs.
     2. Update and manage software lifecycles: Update SBOMs whenever there is a change in software configuration. For example, the state of your source code, the state of your build readiness, the state of your built binaries, and software updates after you release and run them can change your configuration. It's important that your SBOM is up-to-date to reflect the state you're using.
     3. SCA tools, configuration management systems, etc. OSS itself may be larger, or there may be a lot of OSS that you rely on indirectly. Use tools to create SBOMs efficiently. Depending on your software development environment, you may want to consider working with CI/CD. You may also want to consolidate the SBOM, such as providing a configuration management system so that you can check it at any time.
     4. E-learning, training: Train developers and SBOM management operators on how to use rules, processes, tools, and evidence management. By taking SBOM management for granted, you can prevent compliance and vulnerability issues from occurring and quickly address them when they do occur.
     5. Respond to legal requests, respond to industry and business partners, use of SPDX Lite: Legal requests may determine the SBOM and associated management information. In addition, some industry and business partners may set their own management information requirements in addition to responding to such legal requests, so gather information as soon as possible, consult with business partners as necessary, and proceed. In some cases, the information handled by SPDX Lite is sufficient, and in other cases, it can be modified.
     6. Open Source Community Participation: SBOM management is discussed in various open source communities. Because the SBOM formats SPDX and CycloneDX are open standards, their specification and management are discussed in each open source community. We also see this activity in communities that develop tools for open source compliance. In addition, the SBOM and tool communities may collaborate and hold Plugfests to validate SBOM interoperability. By participating in these communities, you can discover common challenges and work together to solve them.
  2. Case studies, examples

the Ministry of Economy, Trade and Industry provides examples of SBOM operations,

* + 1. utilization of open source software and management methods for ensuring its security (expanded May 2022) <https://www.meti.go.jp/press/2022/05/20220510001/20220510001.html>
    2. Reference Materials: Introduction of Software Bill of Materials (SBOM) (draft) <https://www.meti.go.jp/shingikai/mono_info_service/sangyo_cyber/wg_seido/wg_bunyaodan/software/009.html>
  1. evolving topic

\*NTIA, CISA, EU CRA, EO14028, Economic Security Promotion Bill, etc.?

\*SBOM Attestation. SBOM related efforts in OpenSSF.

\*Introduction of each tool?

# Addressing OSS security risks

* 1. Introduction

Open source software (OSS) plays an important role in modern software development, but in addition to understanding licensing, you also need to understand security correctly. In 2021, Apache Log4j, the log library where the vulnerability was discovered, became a public issue. There have also been [reports of attacks targeting](https://jpn.nec.com/cybersecurity/blog/211217/index.html) Node.js (NPM) and Python (PyPI) package repositories. Attacks exploiting OSS vulnerabilities are on the rise, and against this backdrop, companies and organizations are being asked to take more security measures against OSS than ever before.

OSS can have different security levels, such as the number and severity of vulnerabilities, and the speed at which vulnerabilities can be fixed, depending on whether they have similar functionality or different versions of the same component. For this reason, enterprises must choose the appropriate OSS components and versions for their intended use. In addition, although there may be multiple contributors to OSS and security fixes have been made, new vulnerabilities may be discovered and reported at all times, so it is necessary to collect the latest information and take appropriate action. [\*]

This chapter analyzes the security risks of OSS and considers effective countermeasures against those risks. It also includes reports and useful guidelines to help you better understand the security risks of OSS. Readers include developers, system administrators, security professionals, and OSS proponents. We hope you will reaffirm the importance of OSS security and offer some tips on how to address risks.

* 1. Types of OSS Security Risks

Open Source Software (OSS) Many people work on it, but the quality varies. Consider the security risks involved in handling it.

1. Adoption of vulnerable versions: Older versions of OSS released in the past may have security issues. When new vulnerabilities are discovered, OSS projects typically release the latest version that fixes the vulnerability. If you use an older version of OSS, an attacker could exploit an unfixed vulnerability to attack you. [\*]In addition, older versions may use vulnerable protocols, and even then, they may not meet the latest security standards.
2. Vulnerability in Package Libraries: The OSS you are trying to use may have packages or libraries that they depend on. If the dependent software version is old and vulnerable, a. [\*]can affect applications and systems in the same way as the risks listed in a.
3. Unpatched: When an OSS project releases a patch for a vulnerability, a repository administrator, such as a Linux distribution, distributes the patch released by the OSS project to the repository. If the system administrator does not apply the patch, the vulnerability continues to be threatened.
4. Risk due to source code exposure: Because the OSS source code is publicly available, it is important to be aware that a malicious party could seek out a vulnerability and exploit it. However, because OSS is often monitored and improved by large numbers of people, the presence of a collaborative community can mitigate that risk. [\*]The benefits and risks are inextricably linked. In published code, the more users there are, the more likely it is that a vulnerability will be discovered early, and the more likely it is that it will be fixed.
5. Support for development projects: Because OSS is developed by development projects, the development project is responsible for fixing bugs and vulnerabilities. Some activities, such as user communities and companies, help address vulnerabilities, and third parties may provide fixes. However, for communities that are not active or do not prioritize security risk resolution, there may be delays in resolving issues, and some reports (\*) report higher average time to fix than proprietary software.

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

Survey finds that the time it takes to fix vulnerabilities in OSS projects has increased. <https://www.conversion.co.jp/tecblog/20220805>

OSS may be compared to non-open source software such as proprietary software due to the risks and trends mentioned above, but it is not possible to decide which is safer. Any software poses a risk to security, but it is important to correctly assess the risk and take appropriate measures.

* 1. How to discover OSS security risks

Ways to discover known vulnerabilities include the following: Combine these methods to determine the best vulnerability collection method for your organization and project context.

1. Security risk information databases: CVEs, JVNs, and other databases that provide information security information contain many types of security risk information, including OSS vulnerabilities.

Sources of Related Information You Can See

* 1. CVE(Common Vulnerabilities and Exposures):<https://www.cve.org/>
  2. IPA (Information Processing Promotion Organization) Vulnerability Countermeasures HP: <http://www.ipa.go.jp/security/vuln/>
  3. JVN (Japan Vulnerability Notes) Vulnerability Countermeasure Information Database: <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. Using the Security Information Distribution Service: You can learn about known vulnerabilities by checking the services that disclose vulnerability information discovered by security vendors. Some services may provide additional vulnerability information as an added value.
2. Joining the Security Group Community: By joining a security group of companies and corporations, you can obtain the latest information on OSS security trends, attack vectors, and solutions with other members of related industries. [\*]
3. Tool-based Vulnerability Scanning: You can discover known vulnerabilities by scanning with risk analysis tools provided by security vendors or tools published as OSS. [\*]
   1. When an OSS security risk response vulnerability is discovered, it is important to quickly apply the released patches.

If it is difficult to respond to all vulnerabilities, use the published CVSS (\*) evaluation results to determine the degree of impact of the vulnerability on the system, and triage based on the impact on the environment is effective. Triage involves communicating with stakeholders such as customers, management, and developers of an organization, and prioritizing the corresponding vulnerabilities. [\*]In an emergency that requires immediate environmental attention, it can be practically difficult to address all identified risks, and it is necessary to clarify unacceptable risks.

If it is difficult to apply a patch, or if the patch itself is not released, it is necessary to consider an alternative library that does not compromise security. [\*]

\*The CVSS score is a rating of vulnerability severity in the range of 0~10.0. IPA Common Vulnerability Scoring System CVSS Overview. <https://www.ipa.go.jp/security/vuln/CVSS.html>IPA Common Vulnerability Scoring System CVSS Overview shows that in addition to the basic assessment criteria, the "Status Assessment Criteria" used to assess whether a vulnerability is currently available, and the "Environmental Assessment Criteria" used to assess the degree to which a vulnerability affects an organization's environment, can be used to determine the priority of an efficient vulnerability response for an organization. As an example of a metric, Access Vector is a metric that indicates whether an attack is carried out over a network or can be attacked locally. This allows you to see various threats, such as the scope of attack and whether privileges are required. To properly understand the threat of a vulnerability, it is important to consider each factor, not just the overall CVSS score.

However, even if the CVSS score is not high, if an attack has already been made against the vulnerability, you should consider taking action immediately. In this case, you can refer to the "[Known Exploited Vulnerabilities Catalog (KEV)](https://www.cisa.gov/known-exploited-vulnerabilities-catalog)" published by CISA. It is also useful to refer to the dangerous vulnerabilities that have already been observed to be exploited. (US Government systems must be patched by the Due Date)

* 1. OSS security risk management and prevention There are several ways to proactively address and prevent OSS security risks.

1. Update the version of your environment: Periodically replace your OSS components with the latest version and apply security updates.
2. Periodic vulnerability scanning: Use vulnerability scanning tools to detect known vulnerabilities that can be addressed.

The PSIRT Services Framework and the PSIRT Maturity Document, available in Japanese from JPCERT/CC, are very helpful for designing organizations and processes that respond to software security incidents. This framework and model can also be applied to products that use OSS. [\*]

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

* 1. Best Practices for Managing OSS Security Risks Here are some best practices for managing OSS security risks.
     1. Leverage Genuine Packages

We recommend that you install OSS from an official repository or package manager. If you have a malicious package registrant, the downloaded package may contain a malicious algorithm. You can ensure safety by installing from a package distributor that has official authentication.

* + 1. Understanding the support lifecycle Understanding the OSS support lifecycle that you adopt can help you select the correct version of software, adopt related packages, and plan for security updates, thereby reducing the risk associated with choosing OSS.  
       Support may be from the latest major version to several previous versions, but it is generally recommended that you adopt the latest version. For some OSS projects, support policies and software lifecycles can be found on official Web pages, so you should review this information.

* + 1. Software Vulnerability Assessment Using CVSS scores from vulnerability databases, vulnerability information sites, and other sources, OSS vulnerability assessments can be used to address vulnerabilities in software.
    2. Vulnerability Scanning Tools

You can use risk analysis tools provided by security vendors or tools published as OSS to scan for vulnerabilities. By identifying and managing vulnerabilities, you can reduce OSS security risks.

* + 1. Develop vulnerability management guides and processes Companies and organizations that use OSS can manage risks by developing and operating security policies, guidelines, and response processes for the software they use.  
       The PSIRT Services Framework is helpful in understanding the process. (→4.5 (b), 4.8 (c))

[\*]

* + 1. e-learning, training By providing security training to developers and operators who use OSS, you can develop an understanding of the correct risks in adopting OSS and the ability to address them early after vulnerability discovery.
    2. Using SBOMs Software bills of materials (SBOMs) are documents that contain information about software components, versions, OSS licenses, vulnerabilities, and more.  
       The SBOM is expected to reduce specific man-hours for the components mentioned above and reduce OSS security risks such as vulnerabilities. \*The SBOM is also useful for OSS licensing issues. The SBOM is a machine-readable file format that requires tools to read and analyze it for general use. [\*]It is explained and explained in detail in Chapter 3.
  1. Trends, Reports, and Case Studies
     1. [OpenSSF 2022 Annual Report](https://www.linuxfoundation.jp/publications/2023/02/openssf-2022-annual-report-jp/)This report was published by the Open Source Security Foundation (OpenSSF) in 2022. The report covers OSS security, community trends, and OpenSSF initiatives.
     2. [Synopsys Open Source Security & Risk Analysis Report (OSSRA)](https://www.synopsys.com/ja-jp/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html)

A report published by Synopsys that analyzes the OSS risk situation. It is based on information from audits conducted during mergers and acquisitions and focuses on OSS security as well as license compliance issues.

* + 1. The [State of Open Source Security](https://prtimes.jp/main/html/rd/p/000000007.000092857.html) Snyk provides an analysis of the state of open source security. [[LINK1][LINK2]](https://prtimes.jp/main/html/rd/p/000000007.000092857.html)It provides information on OSS security issues, including open source vulnerabilities and security trends.
    2. [Management Practices for Using and Securing Open Source Software (May 2022)](https://www.meti.go.jp/press/2022/05/20220510001/20220510001.html) A document developed by the Ministry of Economy, Trade and Industry that provides examples of how companies and organizations using open source software (OSS) can use OSS while reducing security risks. It explains the key points for each company in selecting and evaluating OSS, the appropriate procedures for implementing, operating, and maintaining OSS, and OSS licensing considerations.
  1. Resources for OSS Security (Online communities, training, etc.)
     1. [CIS Software Supply Chain Security Guide](https://www.cisecurity.org/insights/white-papers/cis-software-supply-chain-security-guide) Created by the Center for Internet Security (CIS), a US nonprofit organization. It is intended to provide best practices for software supply chain security.

This guide focuses on the importance of security in the software supply chain and includes practical advice for software developers, suppliers, customers, and other stakeholders, as well as strategies, security frameworks, and technical guidance to mitigate the risk of supply chain attacks.

* + 1. [SLSA (Supply chain Levels for Software Artifacts)](https://slsa.dev/) SLSA is a framework for security in the software supply chain. It was developed by Google and published in 2021. SLSA provides best practices for increasing reliability within the software supply chain and protecting against supply chain attacks, and can be used by OSS users and developers as a guide to increasing the reliability of the software supply chain. SLSA Whitepaper: Provides an overview of SLSA, its purpose, a description of each level, and examples of implementation. SLSA Implementation Guide: Provides a guide to implementing SLSA. You can use this guide to help you implement SLSA. SLSA Scorecard — An evaluation tool for each level of SLSA. You can use it to evaluate the SLSA level of your OSS.
    2. [PSIRT Services Framework, PSIRT Maturity Document](https://www.jpcert.or.jp/research/psirtSF.html) (4.4) The PSIRT Services Framework, also introduced in Security Risk Management and Prevention for OSS, is a framework that provides guidance for PSIRT teams to handle product security incidents. Products that leverage OSS can also benefit from this framework. On the other hand, the PSIRT Maturity Document provides information to assess the organization's PSIRT configuration and process maturity. These documents are available in Japanese from JPCERT/CC.
  1. Practical tips, tips, and more related to source software security
     1. How do I deal with OSS vulnerabilities that I discover? When OSS vulnerabilities are discovered, it is important to report them to OSS developers and maintainers as soon as possible. [\*]Reporting policies and procedures may be posted on the OSS project website or repository. If vulnerabilities are discovered, review the policies for each OSS project.

If you don't know where to report them, you can report them to the [Zeroday Initiative](https://www.zerodayinitiative.com/advisories/disclosure_policy/). The Zeroday Initiative is a program run by Trend Micro that collects, reports, and researches vulnerability information. You can report it. Based on the reported information, provide the vendor with information to fix the vulnerability before the patch is published. [\*]

* 1. Other

revision history

|  |  |  |
| --- | --- | --- |
| No. | revision date (Version) | Revision details |
| 1 | (V1.0) |  |
|  |  |  |