

SONY

Project ESSTRA

A Software Suite for Enhancing Software Supply Chain Transparency

ソフトウェアの透明性とトレーサビリティを強化するソフトウェアスイート

2025/3/3

OpenChain Japan Community Day

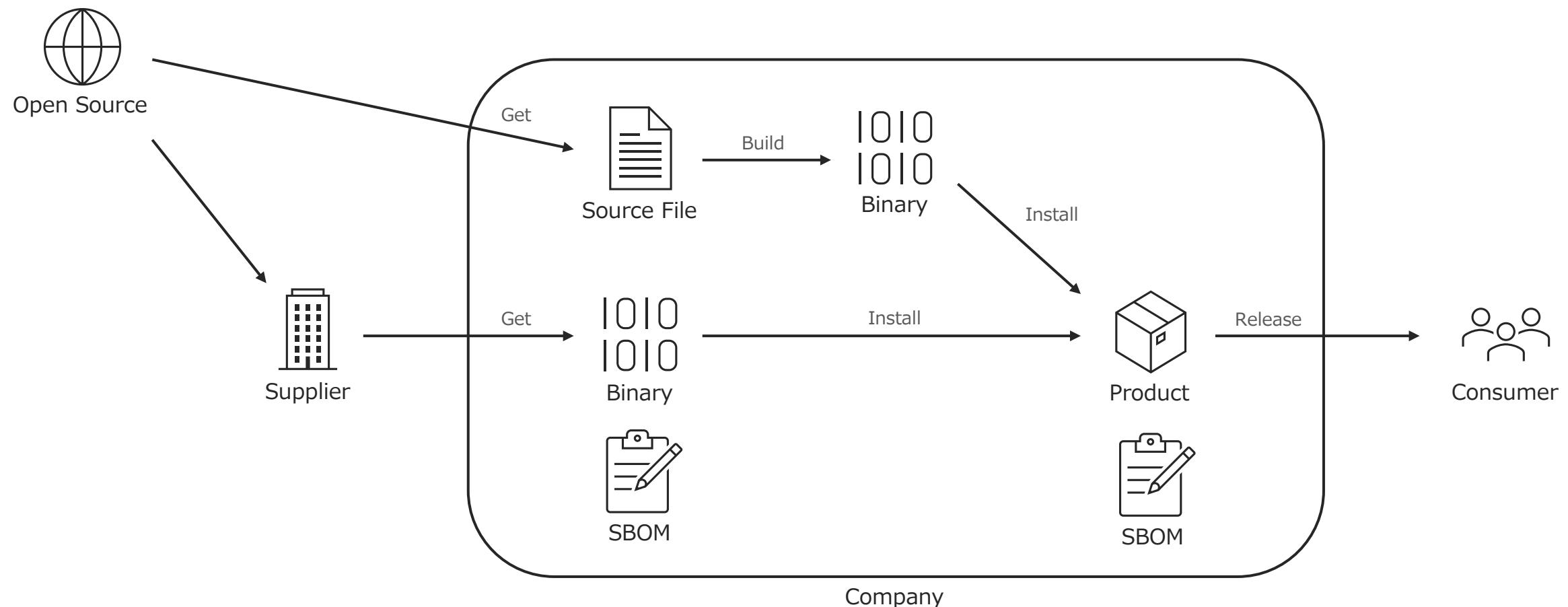
生江 拓也 / NAMAE Takuya

ソニーグループ株式会社

©2025 Sony Group Corporation

はじめに

Q. 製品にインストールされるバイナリが、どのソースファイルから作られたか、そのソースファイルのライセンスは何か、知りたいと思ったことはありますか？



自己紹介

- 生江 拓也 / NAMAE Takuya

- 2003年 ソニー株式会社へ入社
 - 組み込み機器向けWeb Browser開発
 - 電子書籍リーダー向けEPUB3 Viewer開発
 - Chromium（オープンソースのWeb Browser）のライセンス調査がきっかけでOSSライセンス遵守活動に携わるようになります
- 現在
 - ソニー製品向けLinux DistributionのOSSライセンスクリアランス（500+/year）および関連ツール開発をリーディング
 - 社内研修 オープンソースソフトウェア講座【ライセンスクリアランス実務編】講師



目次

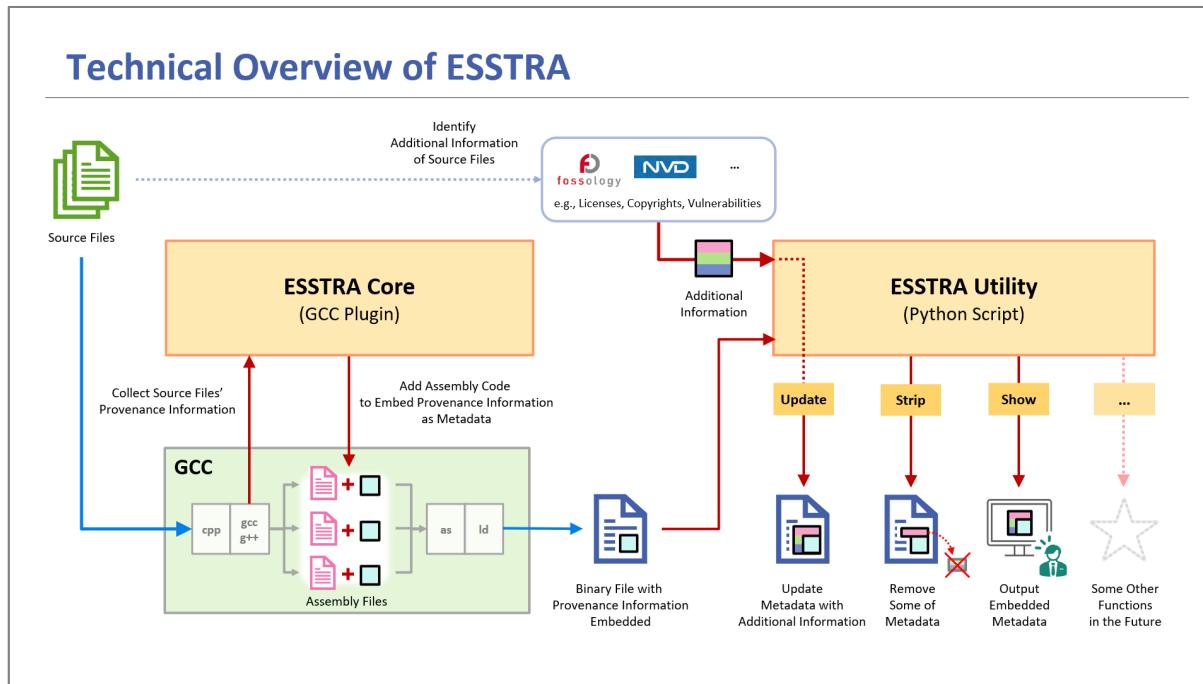
- ESSTRAとは
- 背景・課題
- 目的・効果
- 技術概要・使い方
- 今後の予定
- Q&A

ESSTRAとは

- OSSをGCC (GNU Compiler Collection) でビルドしながら実際に利用しているソースファイル情報を収集しバイナリに埋め込むツール

2024年10月に
オープンソースにしました

<https://github.com/sony/esstra>



バイナリに埋め込まれた情報の表示例

```
#  
# BinaryFileName: hello  
# BinaryPath: /home/snagao/esstra/samples/hello/hello  
#  
SourceFiles:  
/home/snagao/esstra/samples/hello:  
- File: hello.c  
SHA1: e7834d0b9cb6cb116c72f0bee7da29e3d280b27e  
LicenseInfo:  
- MIT  
/usr/include:  
- File: features-time64.h  
SHA1: 57c3c8093c3af70e5851f6d498600e2f6e24fdeb  
- File: features.h  
SHA1: d8725bb98129d6d70ddcbf010021c2841db783f7  
- File: stdc-predef.h  
SHA1: 2fef05d80514ca0be77efec90bda051cf87d771f  
:  
(snip)  
:
```

背景

- 近年OSSを含むソフトウェアサプライチェーン全体の透明性とトレーサビリティを確保することで、脆弱性やライセンス遵守のリスクを軽減させようとする動きが加速している

<https://www.gsa.gov/technology/it-contract-vehicles-and-purchasing-programs/information-technology-category/it-security/executive-order-14028>

The screenshot shows the GSA (U.S. General Services Administration) website. The main navigation bar includes links for 'Buy through us', 'Sell to government', 'Real estate', 'Policy and regulations', 'Small business', 'Travel', 'Technology', and 'About us'. The 'Technology' section is currently selected. Below the navigation, a breadcrumb trail shows the path: Home > Technology > IT contract vehicles and purchasing programs > Information technology category > IT security > Executive Order 14028. The main content area is titled 'Executive Order 14028: Improving the Nation's Cybersecurity'. It includes a summary of requirements, a list of requirements (such as requiring service providers to share cyber incident and threat information), and a section on what contractors can expect. A note at the bottom states: 'Modification of contract language to reflect new guidance from NIST and CISA. If your company cannot accept the modification, you will not be able to do business with the federal government.' The sidebar on the left lists various IT security topics, with 'Executive Order 14028' being the active link.

2021年5月に米国で発令された「国家のサイバーセキュリティ改善に関する大統領令」に取り上げられたことがきっかけでSBOM（ソフトウェア部品表）が注目されるようになり、仕様策定や利用検討などが活発に進んでいる

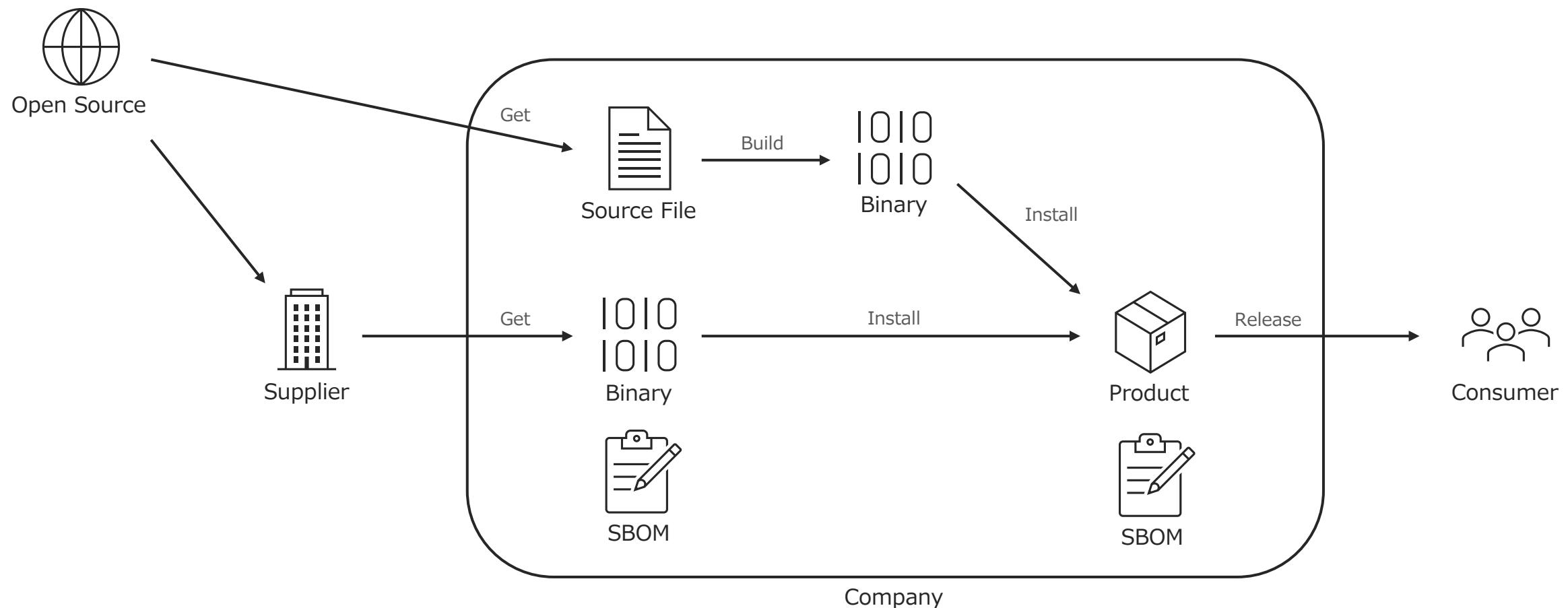
<https://www.iso.org/standard/81039.html>

The screenshot shows the ISO (International Organization for Standardization) website. The top navigation bar includes links for 'Standards', 'Sectors', 'About ISO', 'Insights & news', 'Taking part', and 'Store'. The main content area is titled 'ISO/IEC 5230:2020'. It includes a thumbnail of the standard document, a brief description ('Information technology — OpenChain Specification'), and a note that it was 'Published (Edition 1, 2020)'. On the right side, there is a sidebar for 'ISO/IEC 5230:2020' with options to change language (English), format (PDF or Paper), and an 'Add to cart' button. At the bottom, there is a 'General information' section with details like status (Published), publication date (2020-12), stage (International Standard published), edition (1), number of pages (9), technical committee (ISO/IEC JTC 1), and ICS (35.020). There is also a link for 'RSS updates'.

組織がOSSライセンスコンプライアンスを適切に実行するための要件をThe Linux Foundation傘下のOpenChainプロジェクトが定義し、オープンソースコンプライアンスの国際規格「ISO/IEC 5230:2020」として2020年12月に承認された

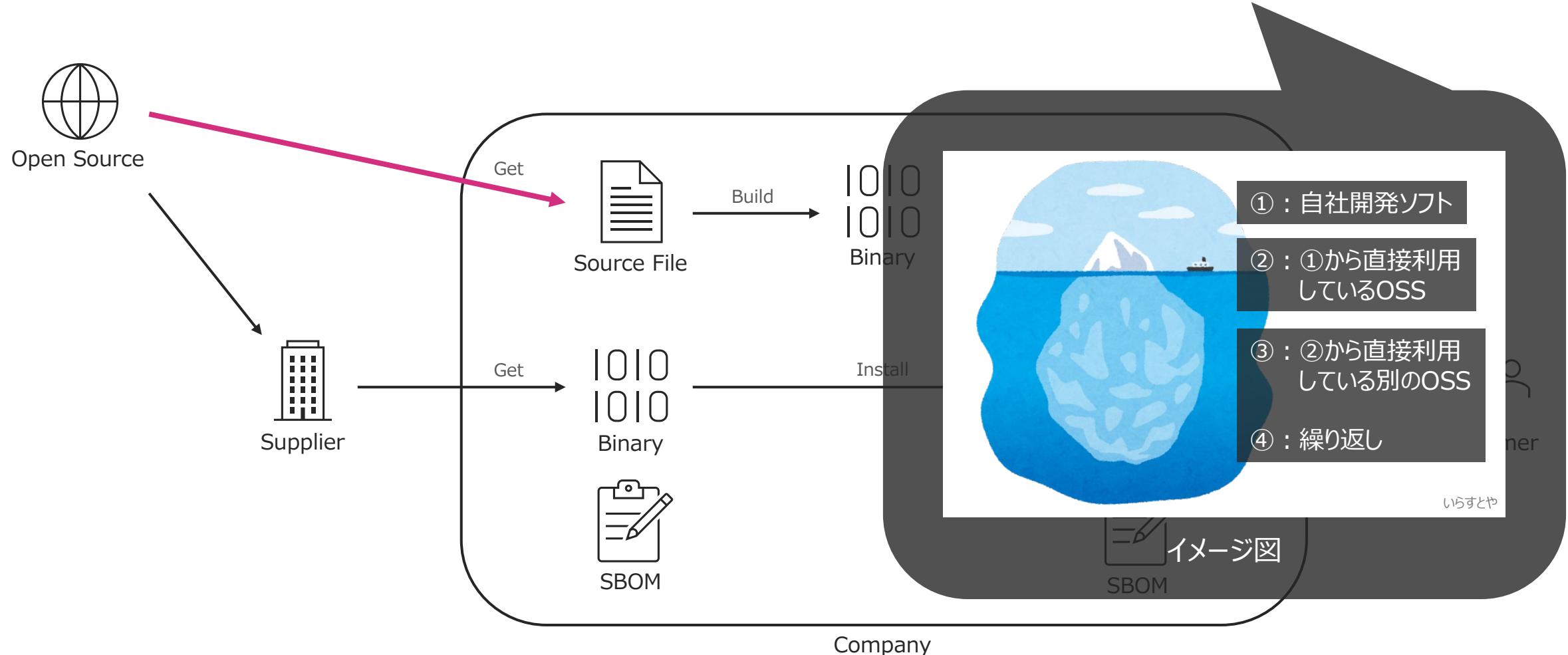
オープンソースを活用した製品開発

- おさらい



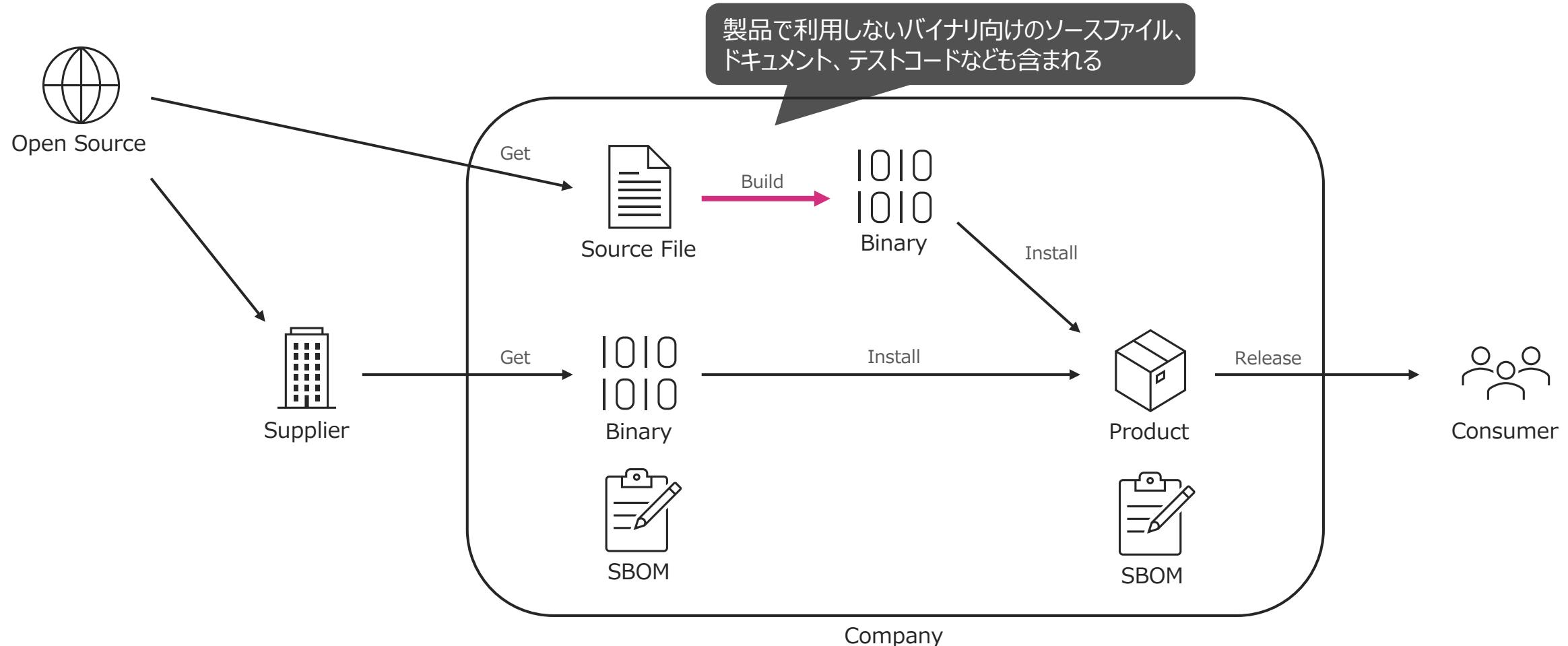
オープンソースを活用した製品開発

- たくさんのオープンソースを直接・間接的に利用している（依存で芋づる式に）



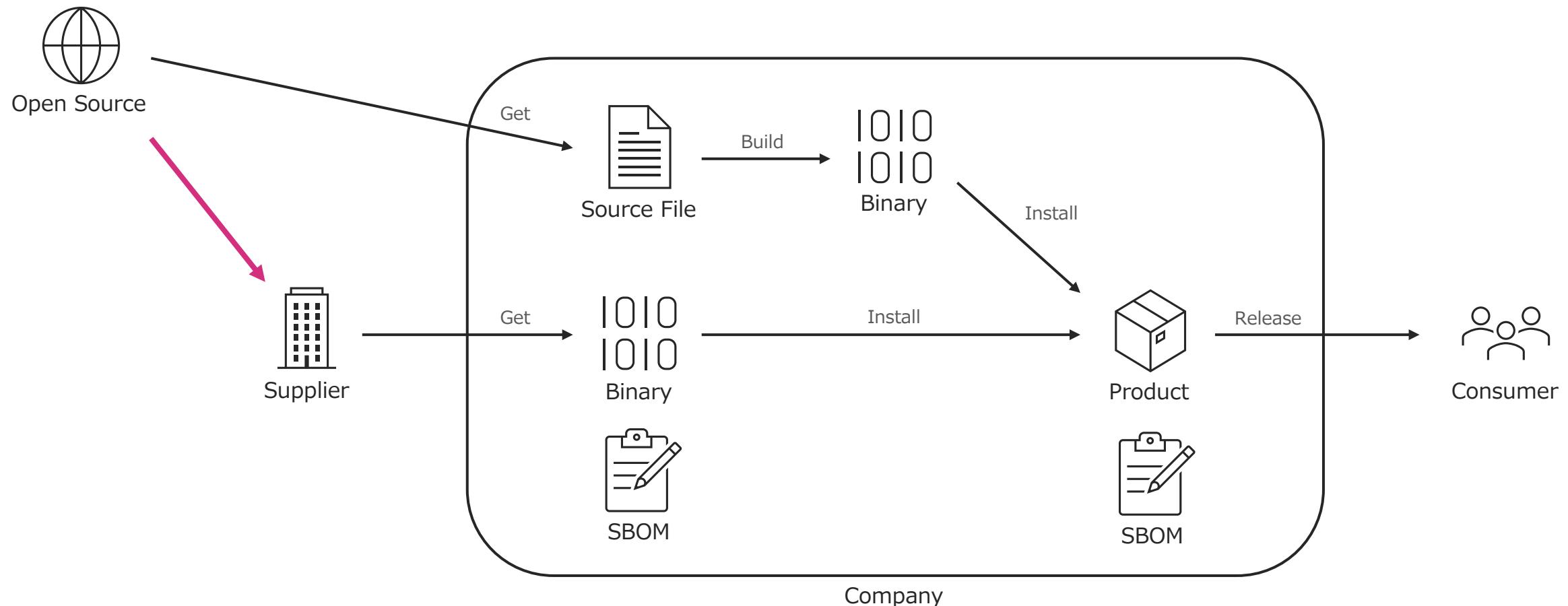
オープンソースを活用した製品開発

- すべてのファイルが製品で利用するバイナリに含まれる、わけではない



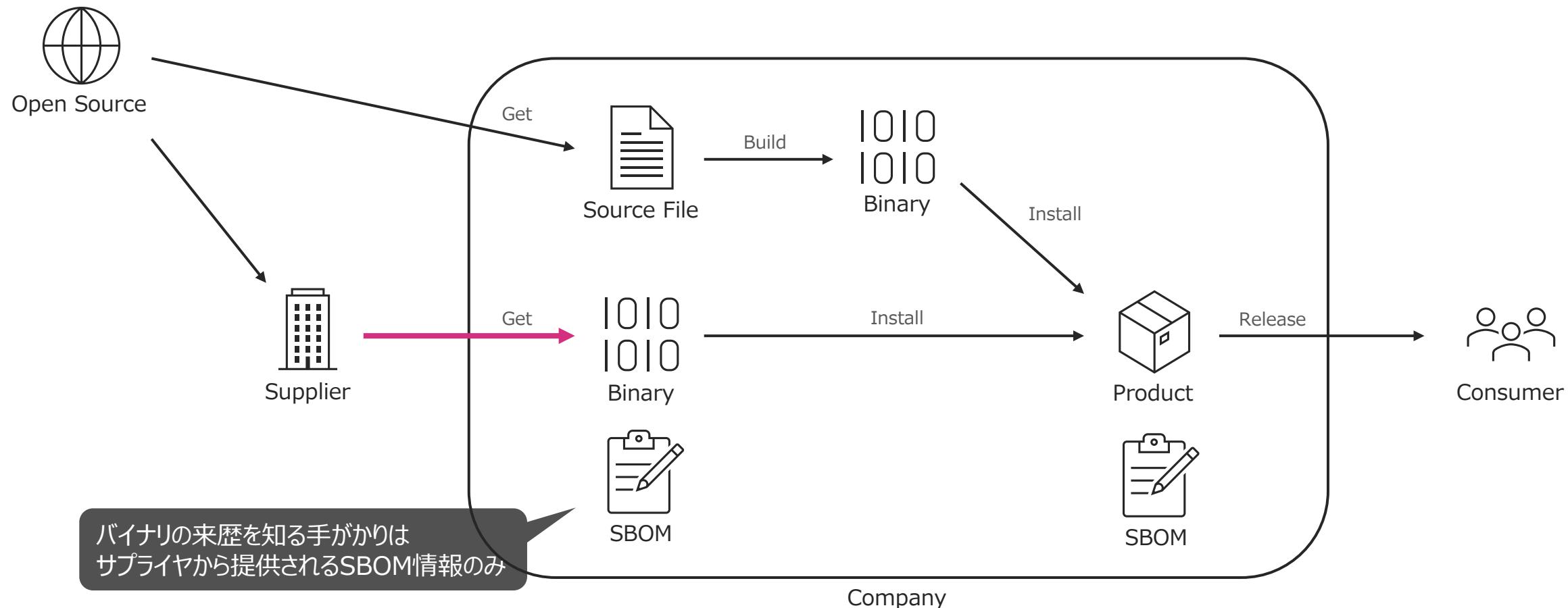
オープンソースを活用した製品開発

- サプライヤもオープンソースを活用している場合がある



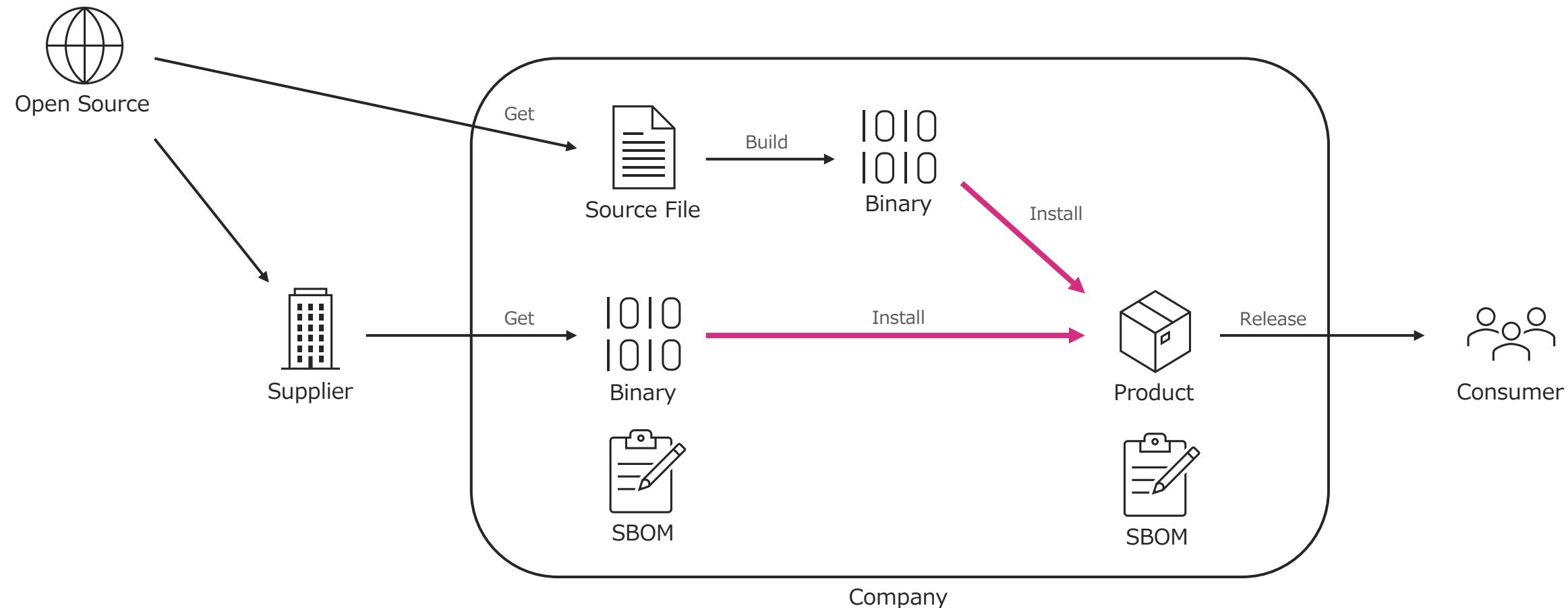
オープンソースを活用した製品開発

- サプライヤからバイナリのみ提供される場合がある



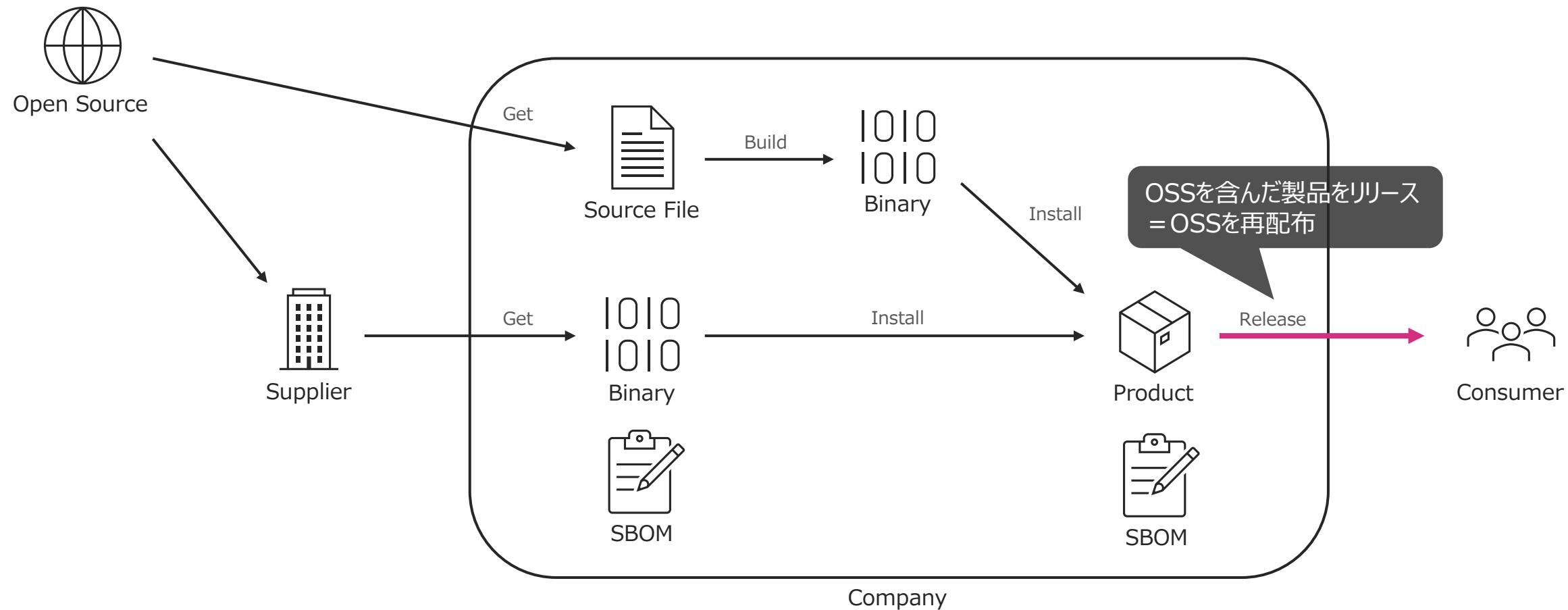
オープンソースを活用した製品開発

- ・ 製品にインストールされるもの = SBOMに掲載するもの



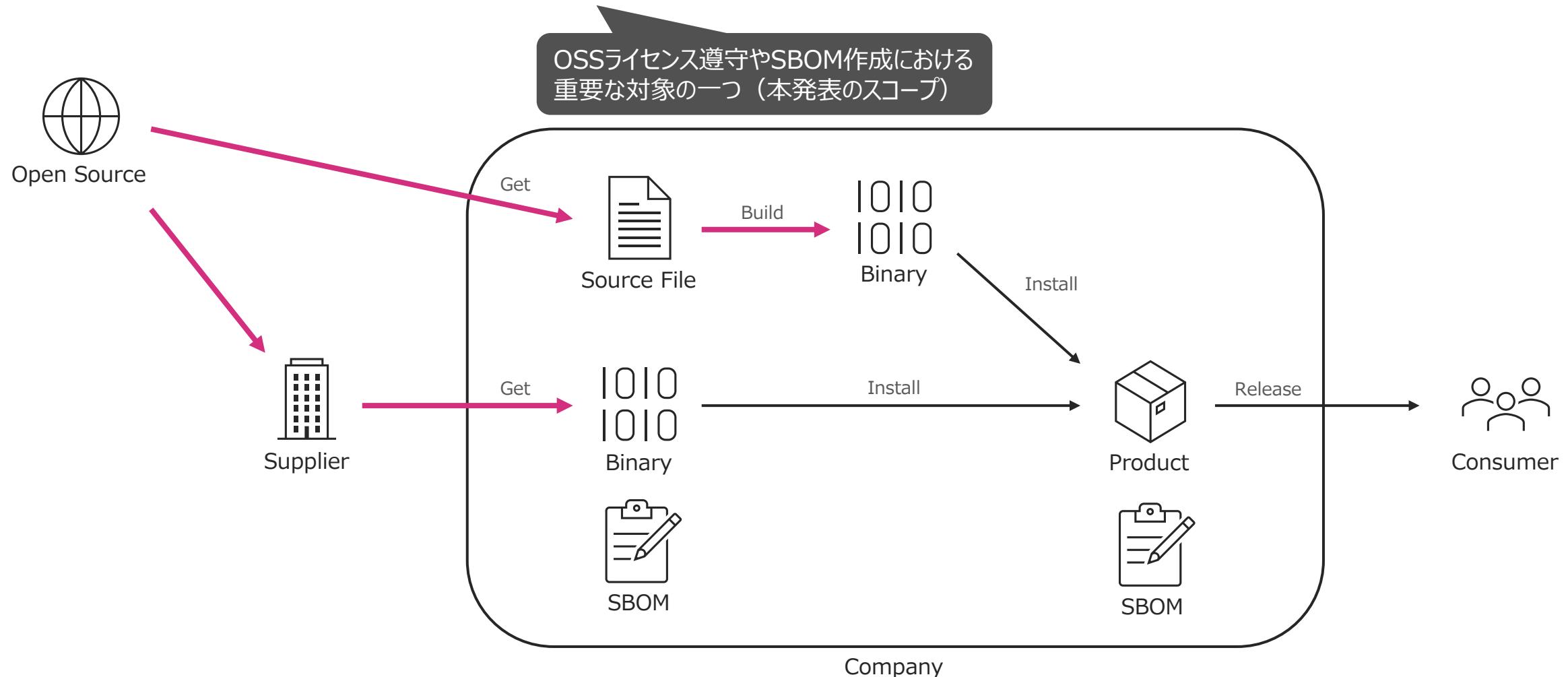
オープンソースを活用した製品開発

- ・ 製品にインストールされるもの = SBOMに掲載するもの = OSSライセンス遵守の対象



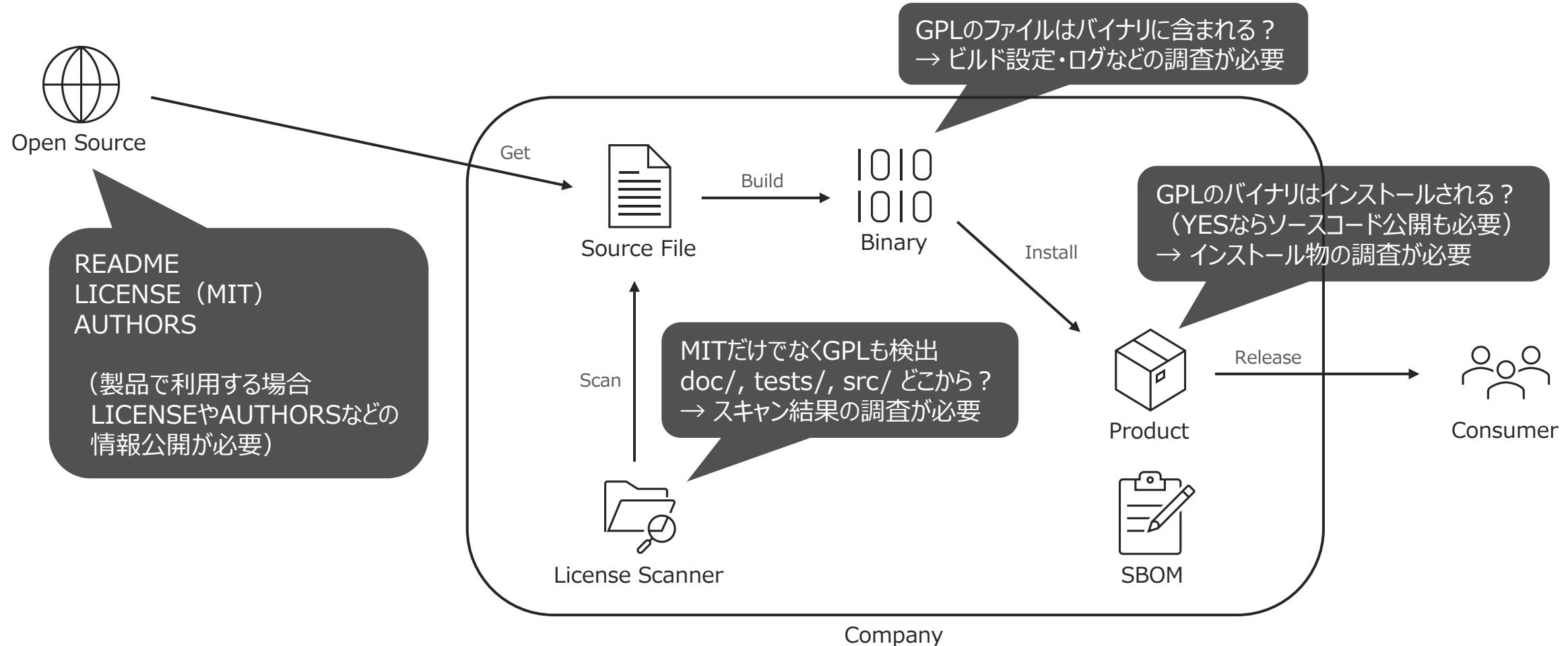
課題

- 製品にインストールされるバイナリの来歴を正確に把握することは困難



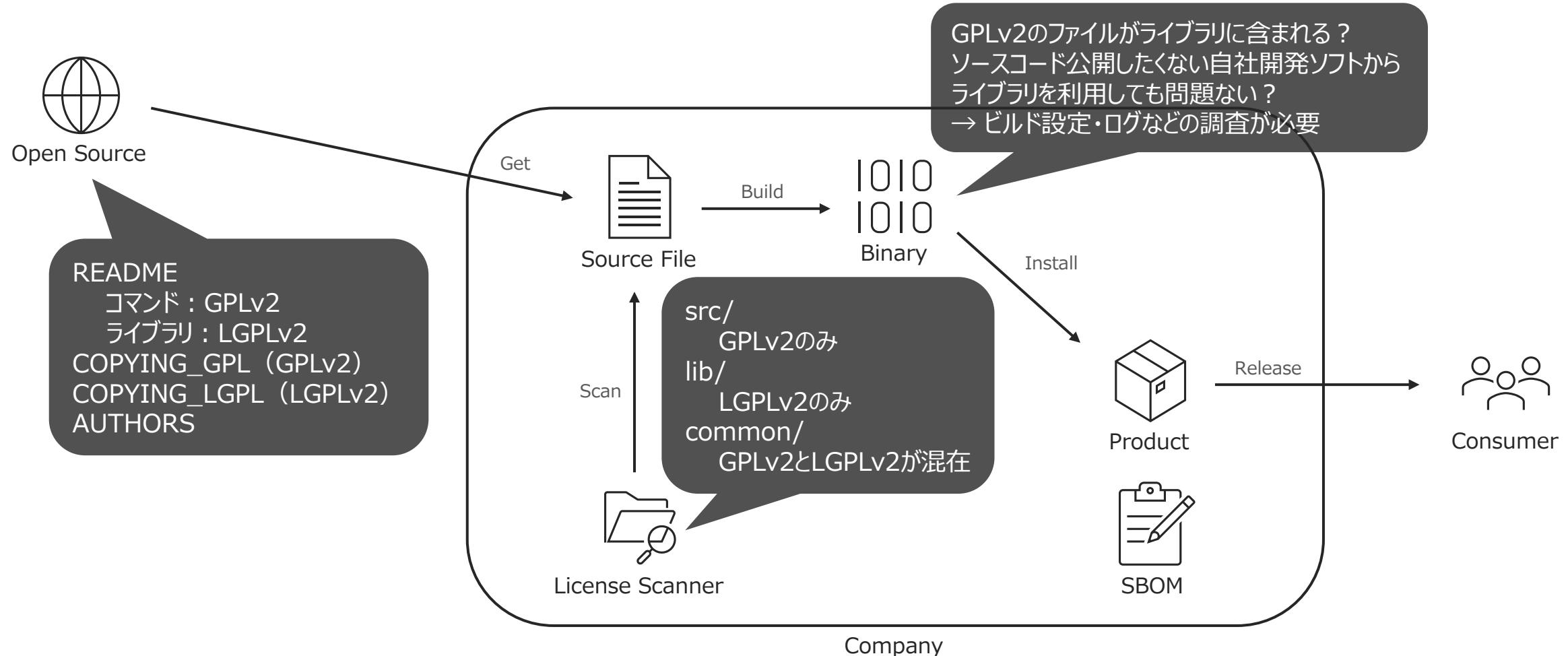
事例 1

- 著作者が宣言していないライセンスがバイナリに含まれている



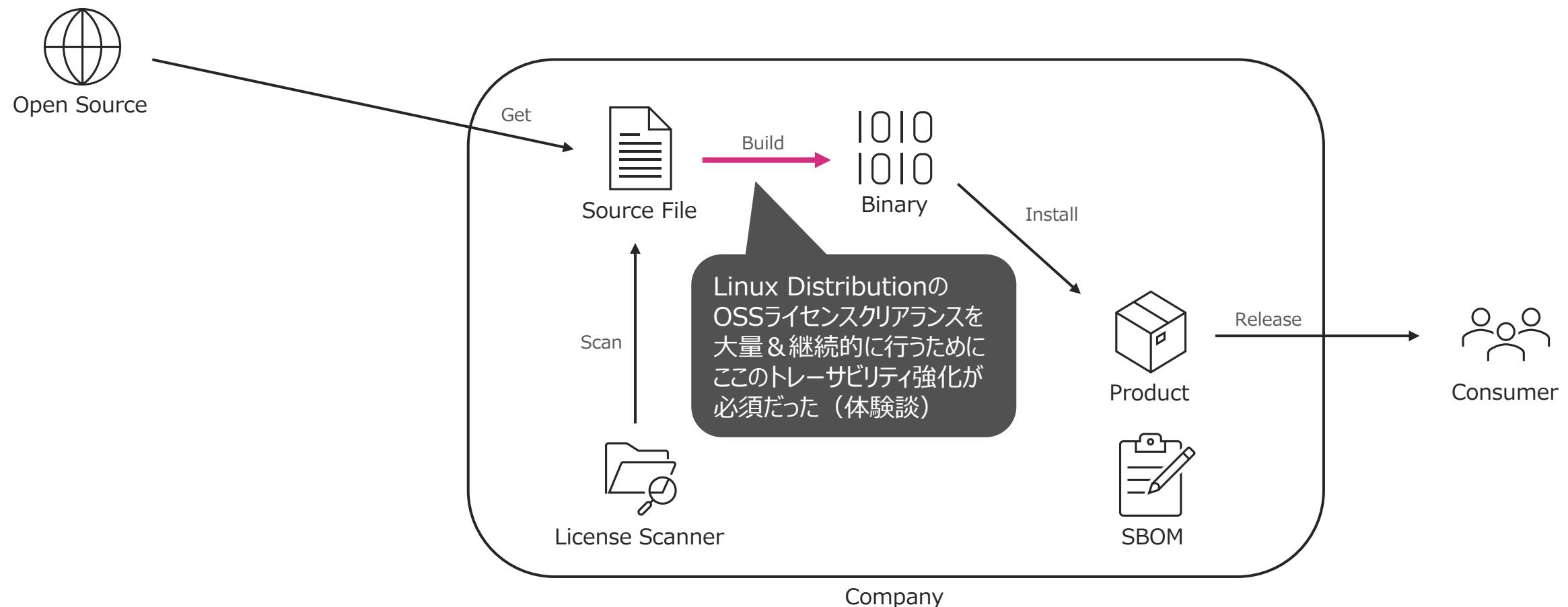
事例 2

- 著作者が宣言したとおりのバイナリ構成になっていない



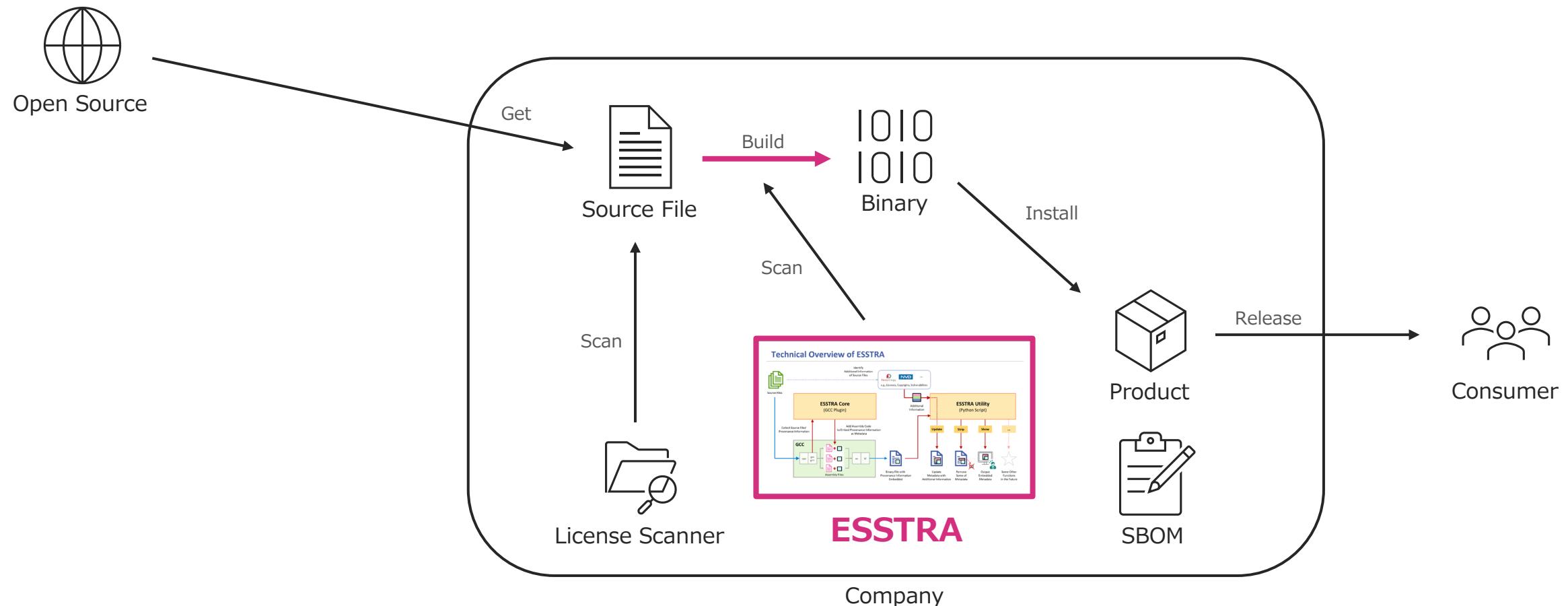
課題解決に向けたポイント

(再掲) Q. 製品にインストールされるバイナリが、どのソースファイルから作られたか、そのソースファイルのライセンスは何か、知りたいと思ったことはありませんか？



目的：なぜESSTRAを開発したのか？

- ・ ソフトウェアの透明性とトレーサビリティを強化するため

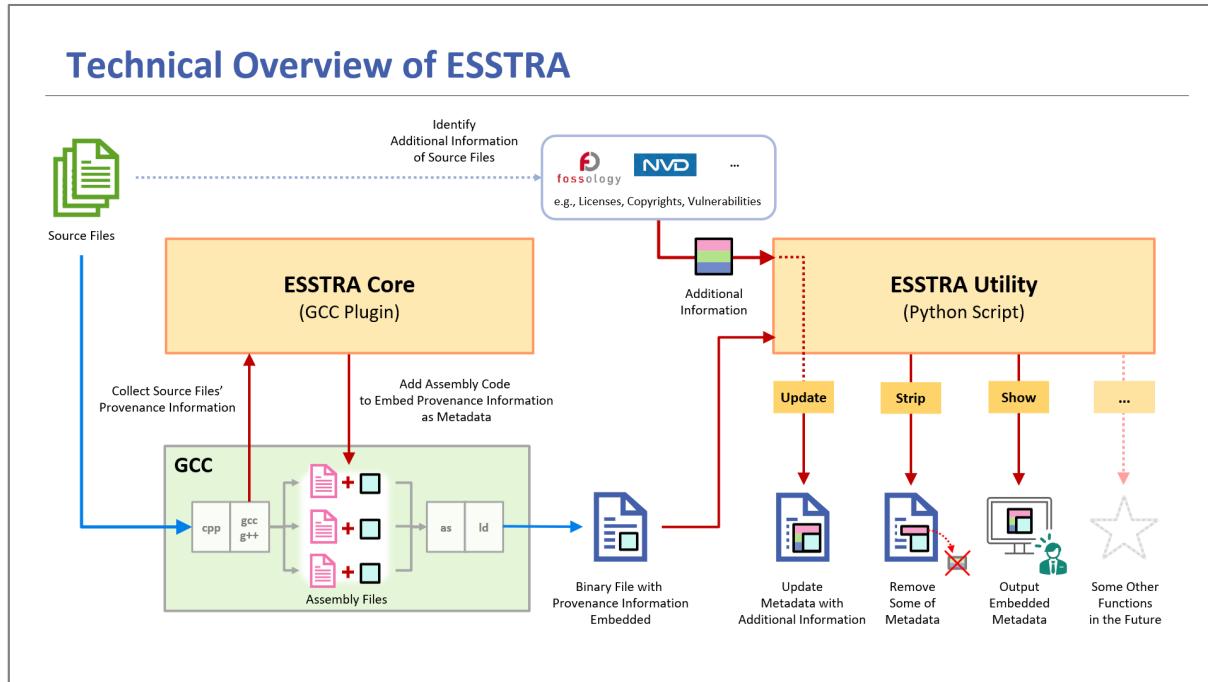


(再掲) ESSTRAとは

- OSSをGCC (GNU Compiler Collection) でビルドしながら実際に利用しているソースファイル情報を収集しバイナリに埋め込むツール

どんなソースファイルやライセンスが含まれるかバイナリごとに分かる

<https://github.com/sony/esstra>



バイナリに埋め込まれた情報の表示例

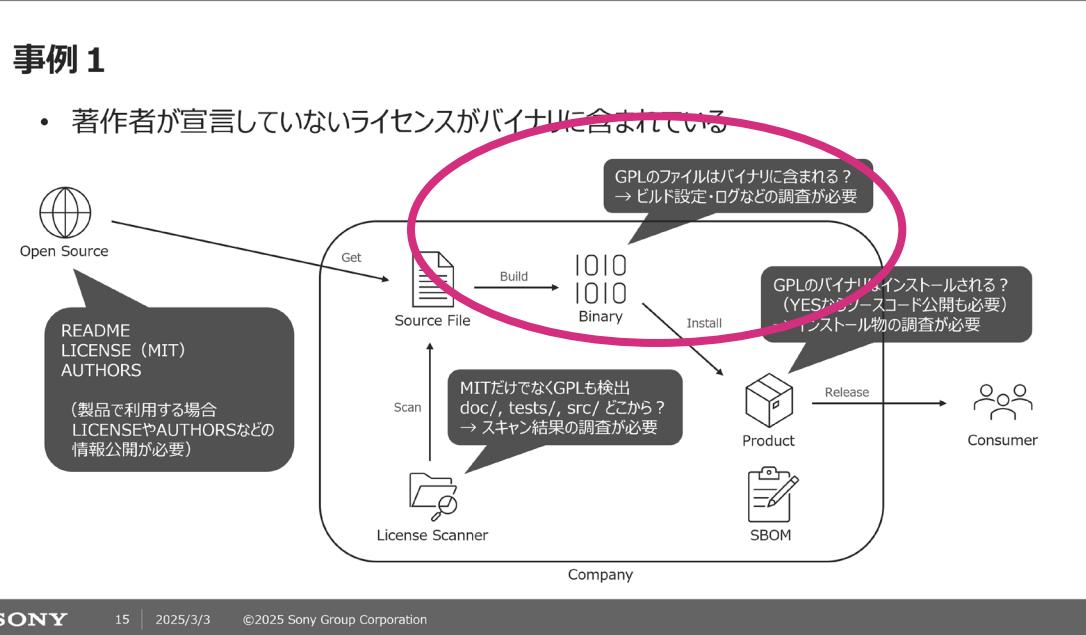
```
#  
# BinaryFileName: hello  
# BinaryPath: /home/snagao/esstra/samples/hello/hello  
#  
SourceFiles:  
/home/snagao/esstra/samples/hello:  
- File: hello.c  
SHA1: e7834d0b9cb6cb116c72f0bee7da29e3d280b27e  
LicenseInfo:  
- MIT  
/usr/include:  
- File: features-time64.h  
SHA1: 57c3c8093c3af70e5851f6d498600e2f6e24fdeb  
- File: features.h  
SHA1: d8725bb98129d6d70ddcbf010021c2841db783f7  
- File: stdc-predef.h  
SHA1: 2fef05d80514ca0be77efec90bda051cf87d771f  
:  
(snip)  
:
```

導入効果

- License Scanner + ESSTRA解析により、ロジカル & 効率的な調査が可能に

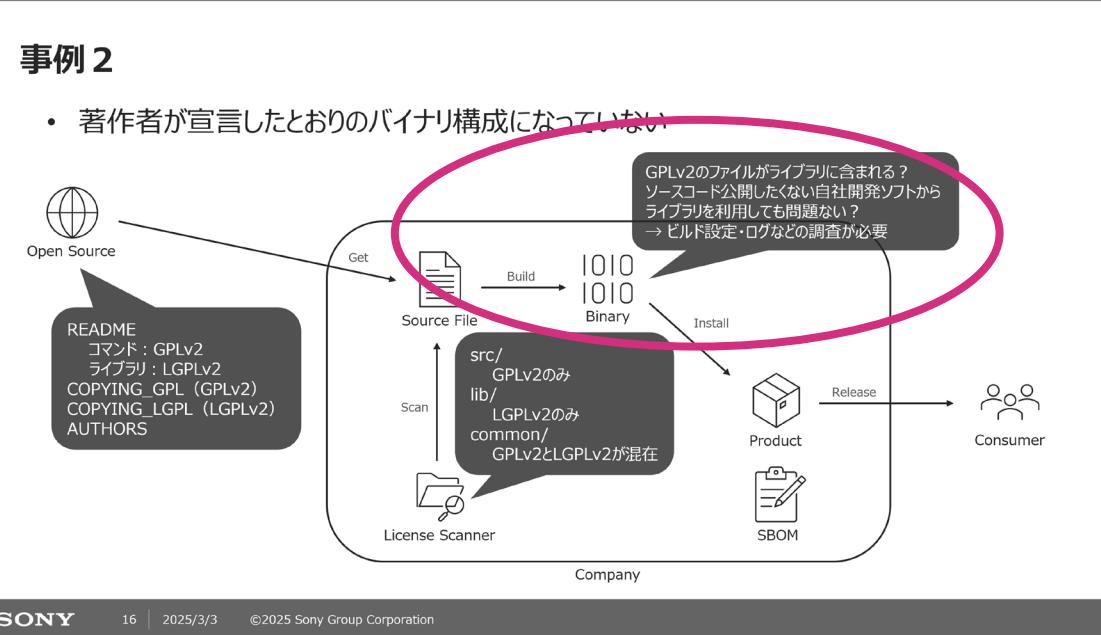
事例 1

- 著者が宣言していないライセンスがバイナリに含まれている



事例 2

- 著者が宣言したとおりのバイナリ構成になっていない



解析結果 :

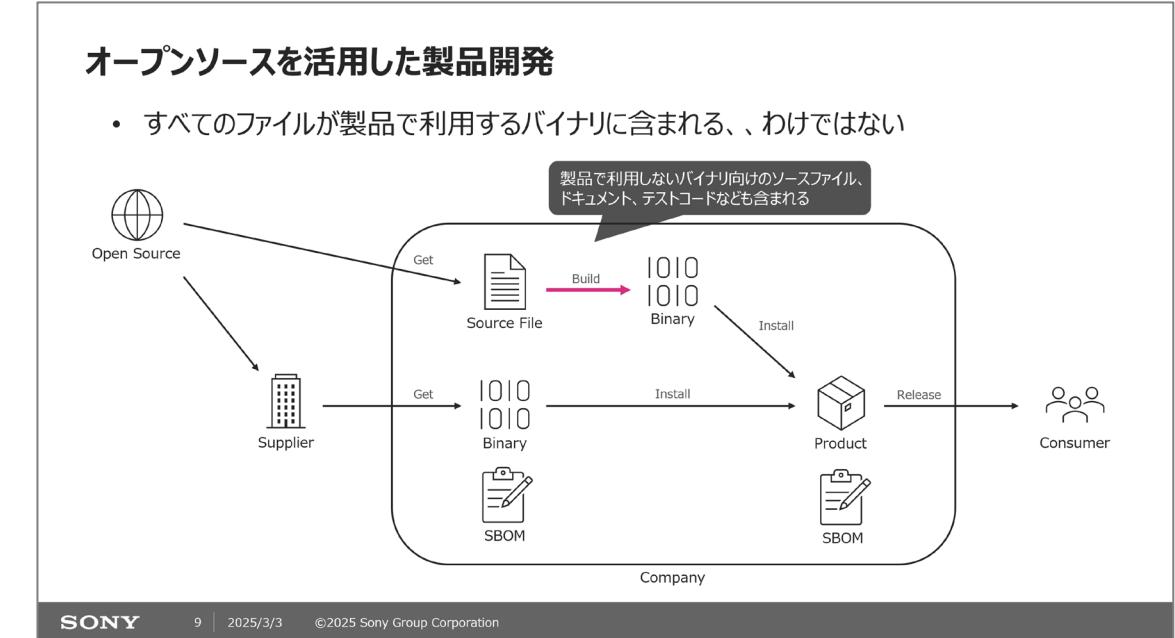
- MITのソースファイルだけでバイナリが生成されたと判明
→ LICENSE (MIT) やAUTHORSなど情報公開
or
- GPLのソースファイルもバイナリに含まれていたと判明
→ 上記に加え、GPL関連の情報 + ソースコード公開

解析結果 :

- GPLv2のソースファイルもライブラリに含まれていたと判明
→ ソースコード公開したくない自社開発ソフトから
直接利用しないように注意喚起・設計変更
or
著者の宣言どおりライブラリがLGPLv2になるように
Open Source Upstreamへcontribution

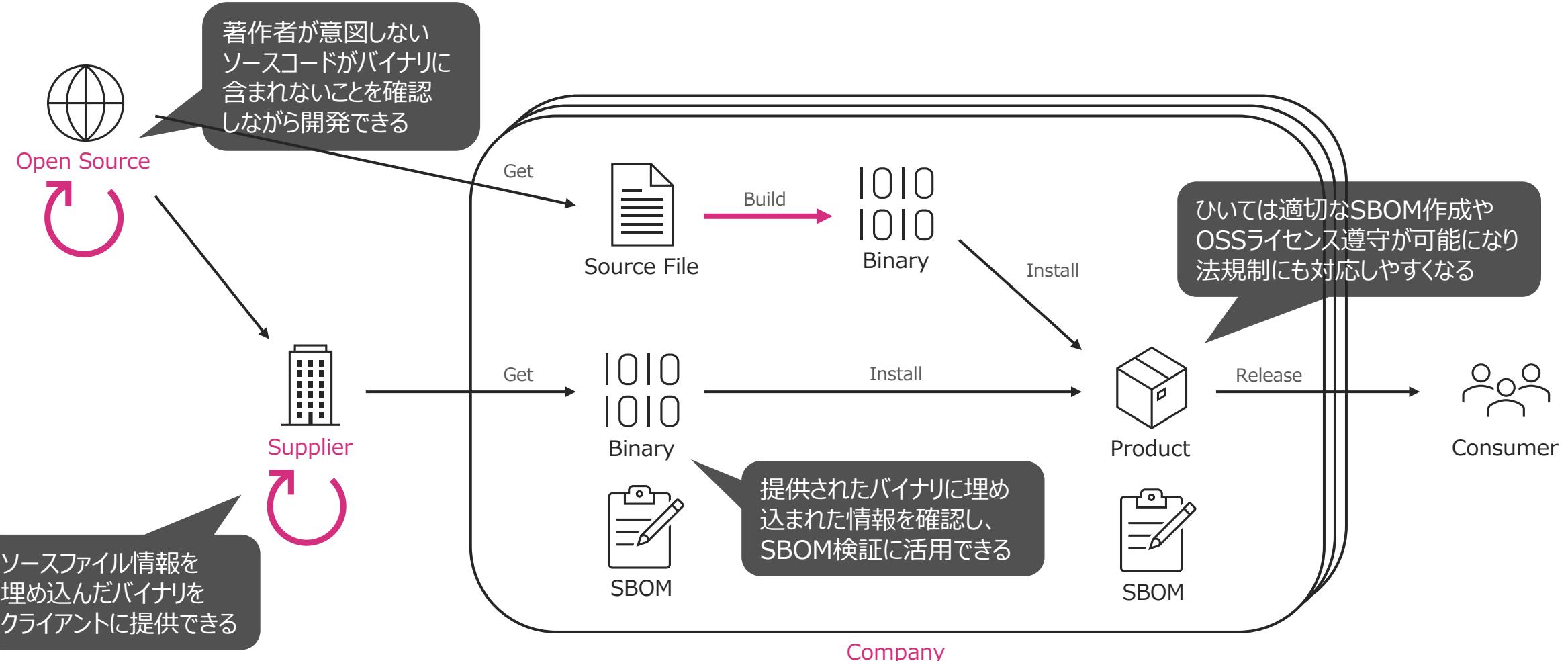
参考値

- Ubuntu24.04LTSベースのシステムソフトウェア系オープンソースのDebian Package（187個）に含まれるバイナリをGCCでビルドして生成する際にESSTRAを活用して元となるソースファイル情報を特定
 - 対象OSSに含まれるファイル数の合計（A）
 - 352,986 個
 - 対象バイナリの元となるソースファイル数の合計（B）
 - 31,159 個
- 商品にインストールされるバイナリの元となるソースファイル数は全体の8.8%
 - $(B) / (A) *100 = 8.8\%$



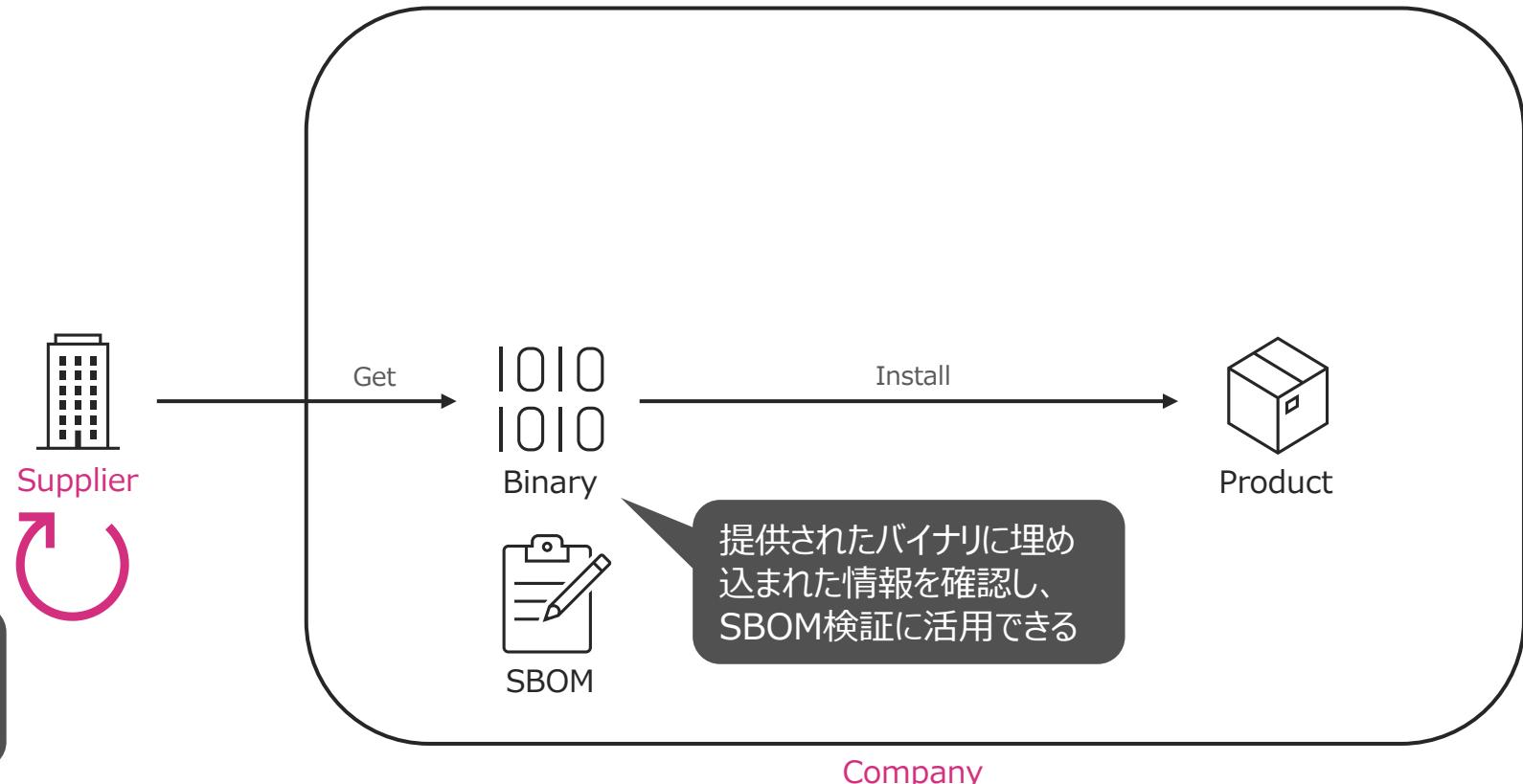
なぜESSTRAをオープンソースにしたのか？

- ・ ソフトウェアサプライチェーンのどこでも、どのフェーズでも、活用できる、活用してほしい

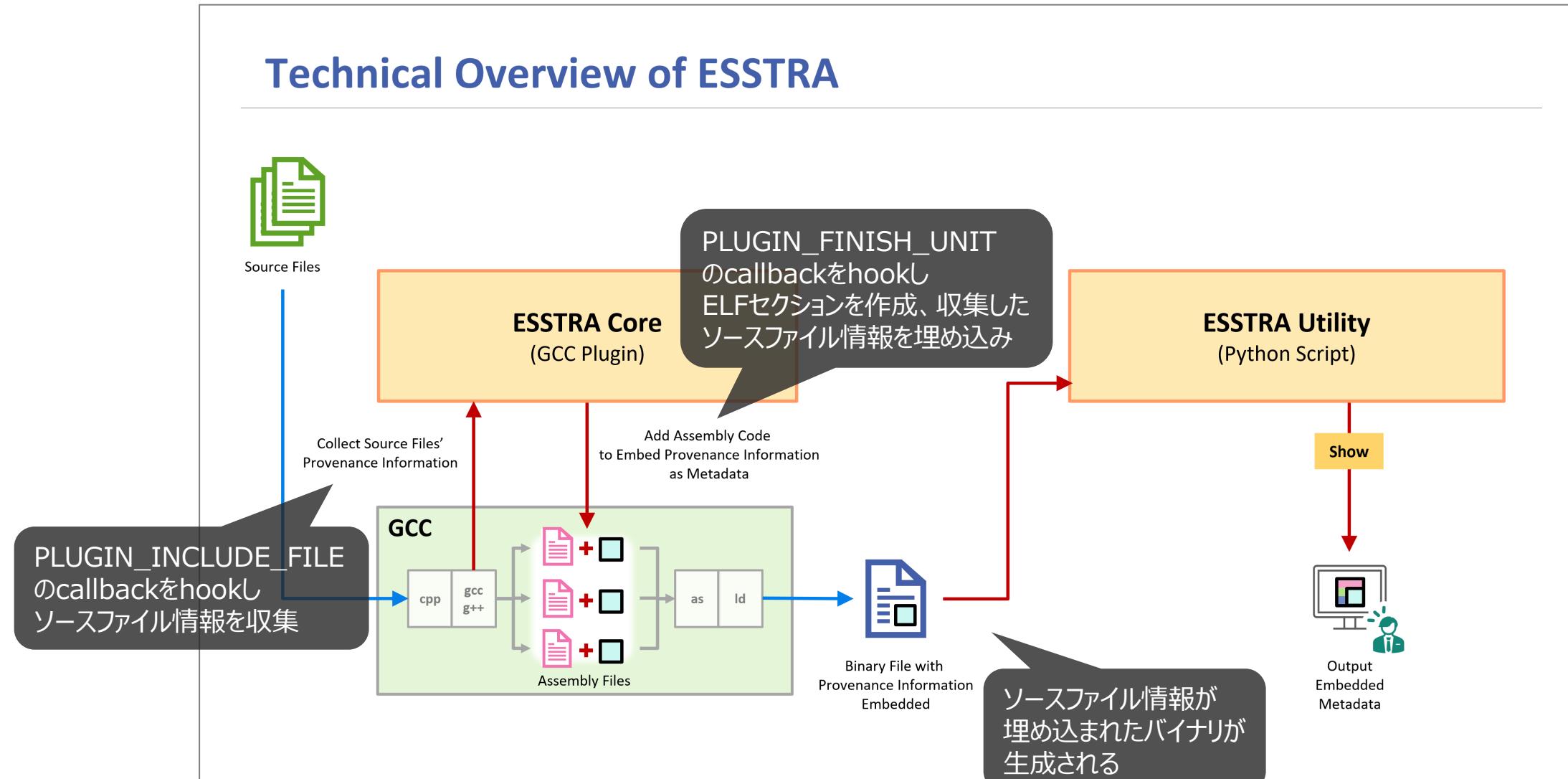


使い方

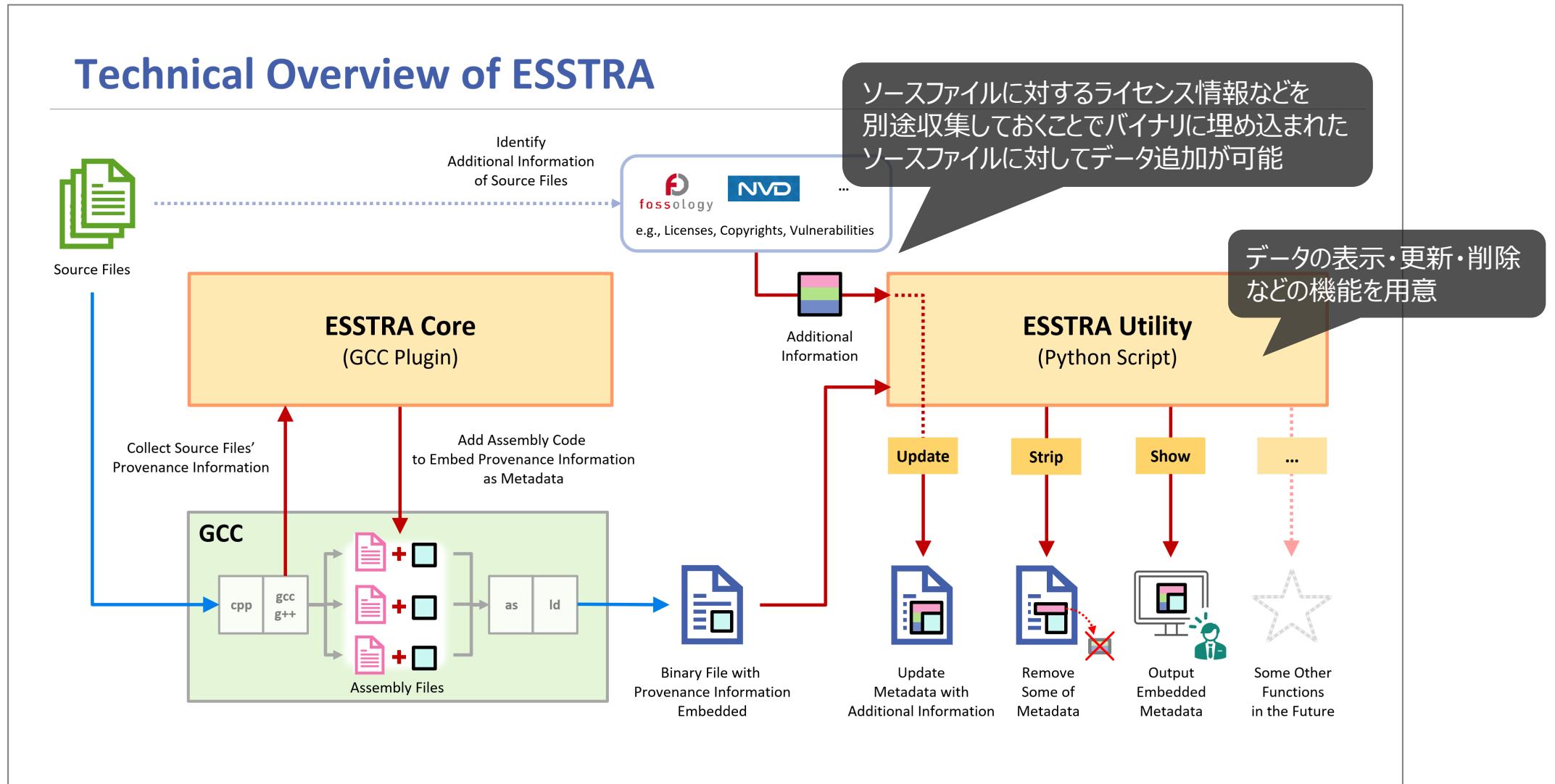
- 以下のシナリオに沿って解説
 - (その前に技術概要を説明)



技術概要 - 基本：ソースファイル情報の収集

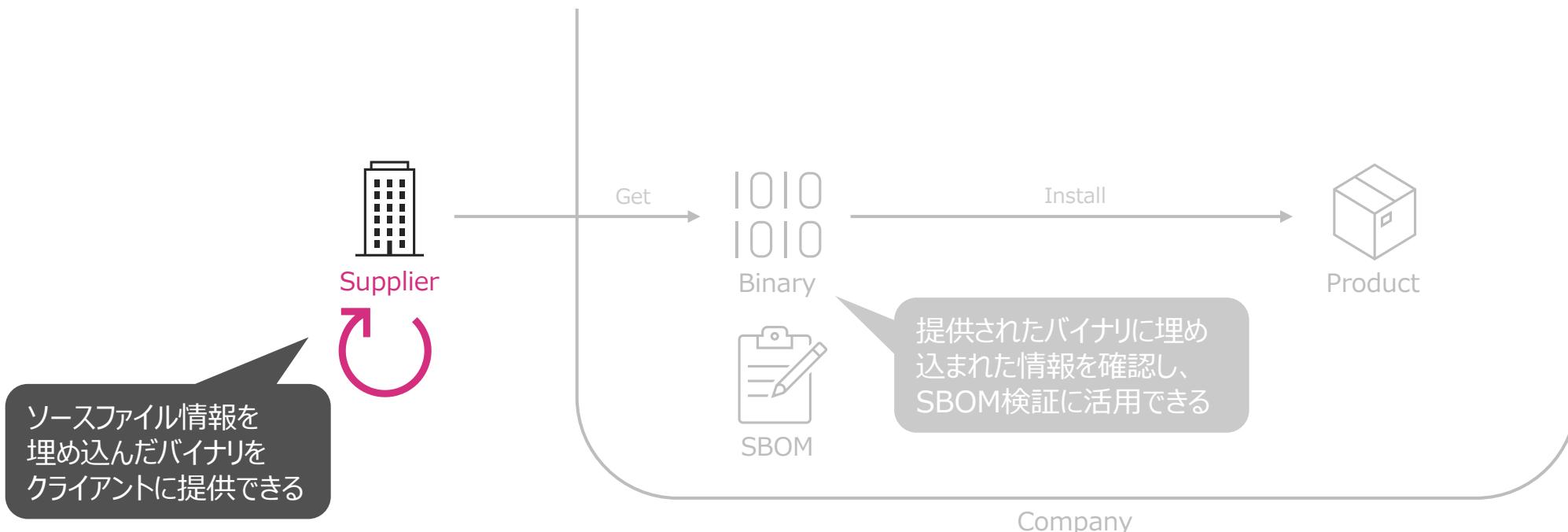


技術概要 - 應用：ソースファイル関連情報の追加



使い方

- シナリオ：ソースファイル＋ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する



使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

Before you build the GCC plugin, you have to install a package on your system. For Debian/Ubuntu, check the version of GCC first:

```
$ gcc --version  
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0  
...
```

In this case, the major version is 11, so install the package named `gcc-11-plugin-dev`:

```
$ sudo apt install gcc-11-plugin-dev
```

After that, run `make` in the top directory:

```
$ make
```

If no errors, a file named `esstracore.so` is generated. This is the GCC plugin "ESSTRA Core."

使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする (Core)
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

To install ESSTRA Core on your system, run the following command:

```
$ sudo make install
```



Then `esstracore.so` is installed in `/usr/local/share/esstra/`.

使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ESSTRA Core (GCC Plugin) をビルドする
 - 対象ソフトウェアをビルドする環境にESSTRAをインストールする (Utility)
 - ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

Since ESSTRA Utility depends on the [PyYAML](#) module to handle YAML data, you may need to install it by, for example, typing:

```
$ pip install pyyaml  
or  
$ sudo apt install python3-yaml
```

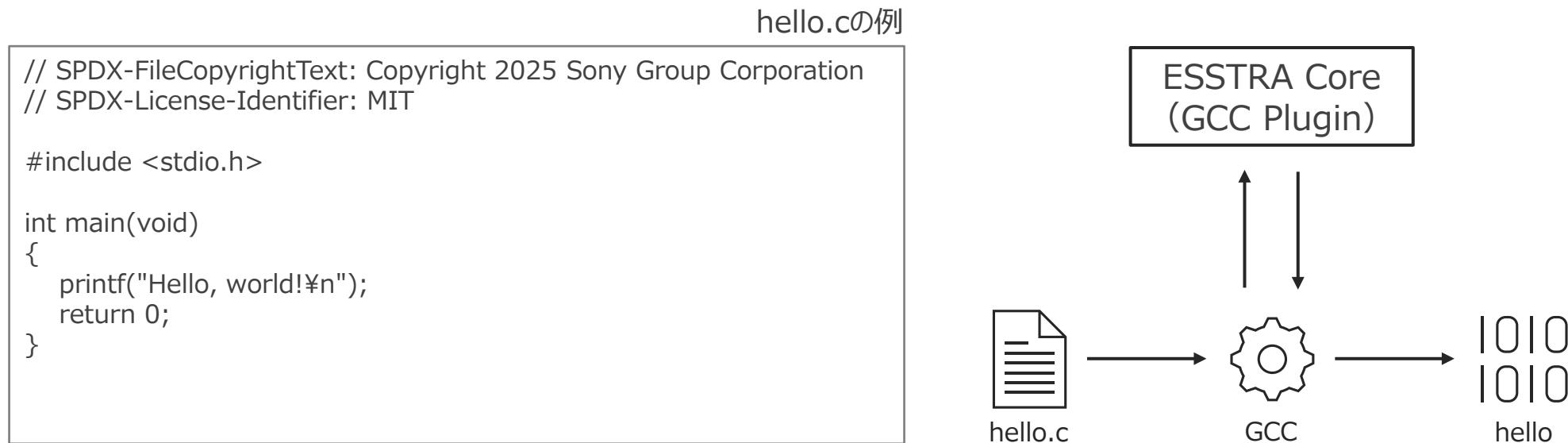
To install `esstra.py` in `/usr/local/bin/`, type:

```
$ sudo make install
```

Make sure that the environment variable `PATH` contains `/usr/local/bin/`.

使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する



使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

To use ESSTRA Core, specify the path of `esstracore.so` using the option `-fplugin=` of `gcc` or `g++`.

For example, if you compile a source file `hello.c` or `hello.cpp` with `gcc` or `g++` to generate a binary file `hello`, type:

```
$ gcc -fplugin=/usr/local/share/esstra/esstracore.so hello.c -o hello  
or  
$ g++ -fplugin=/usr/local/share/esstra/esstracore.so hello.cpp -o hello
```

①でビルドしたesstracore.soを
"-fplugin=" オプションで指定して
hello.cをコンパイルする

The generated binary file `hello` has metadata embedded by ESSTRA Core.

Use [ESSTRA Utility](#) to access metadata embedded in binary files.

Note that this does not affect the behavior of the binary file itself.

使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

Installing a Spec File

It will surely be annoying that you have to specify `-fplugin=....` for every gcc/g++ invocation. If you want to apply ESSTRA Core to any gcc/g++ occurrence, just type:

```
$ sudo make install-specs
```

This installs a [GCC spec file](#) on your system which enables the option `-fplugin=....` as default. So, compiling anything hereafter with GCC as usual:

```
$ gcc hello.c -o hello
```

generates a binary file with metadata embedded by ESSTRA Core.

This is a very useful feature if you compile some open source (or closed or whatever) projects and also want information ESSTRA generates for them. You do not ever need to modify Makefiles, CMakeList.txts, build.ninjas, meson.builds, etc, etc...

However, please note that installing the spec file causes every gcc/g++ call hereafter will be intervened by ESSTRA Core on systemwide.

To disable this, type:

```
$ sudo make uninstall-specs
```

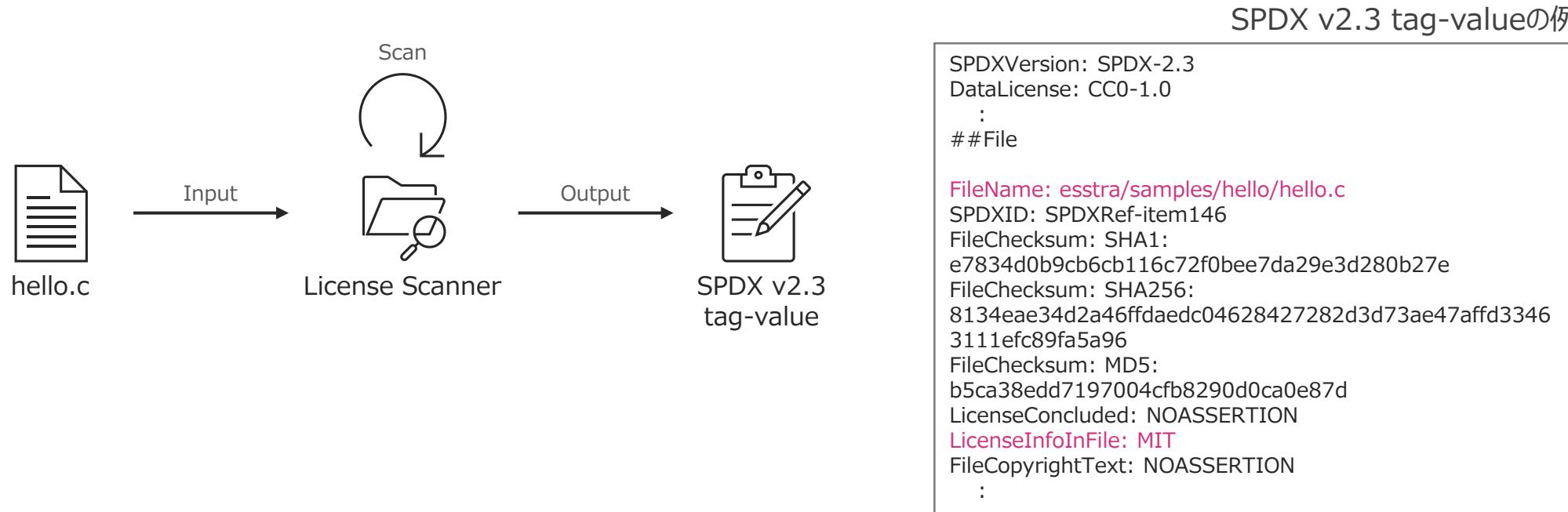
to remove the spec file.

GCCのspecファイル（※）を利用すると
対象ソフトウェアのMakefileなどを編集することなく
ESSTRA Coreのオプション指定を有効化できる

※specファイルとは：
GCCの動作を制御するためのプレーンテキストファイル

使い方

- シナリオ：ソースファイル＋ライセンス情報を埋め込んだバイナリを作つて提供する
 - ① ESSTRA Core (GCC Plugin) をビルドする
 - ② 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ③ ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - ④ License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ⑤ ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する



使い方

- シナリオ：ソースファイル+ライセンス情報を埋め込んだバイナリを作つて提供する
 - ESSTRA Core (GCC Plugin) をビルドする
 - 対象ソフトウェアをビルドする環境にESSTRAをインストールする
 - ESSTRA Coreをオプションに指定して対象ソフトウェアをコンパイルしバイナリを作る
 - License Scannerで対象ソフトウェアを解析し、ソースファイルに対するライセンス情報を取得する
 - ESSTRA Utility (Python Script) を用いてバイナリに埋め込んだデータを更新する

The command `update` of the current version attaches "license information" to each file's information in binary files' metadata.

We think some other kinds of information would also be helpful. For example, copyright information, CVE numbers and so on. Since ESSTRA is at an early stage in development, we have developed a feature that attaches license information as a sort of feasibility study.

To attach license information, you need to prepare an [SPDX 2.3 tag-value](#) file including `LicenseInfoInFile:` tags. Some license scanners like [FOSSology](#) can generate such kind of files.

A typical usage is:

```
$ esstra.py update <binary> -i <spdx-tv-file>
```

例)

```
$ esstra.py update hello -i SPDX2TV_hello.spdx
```

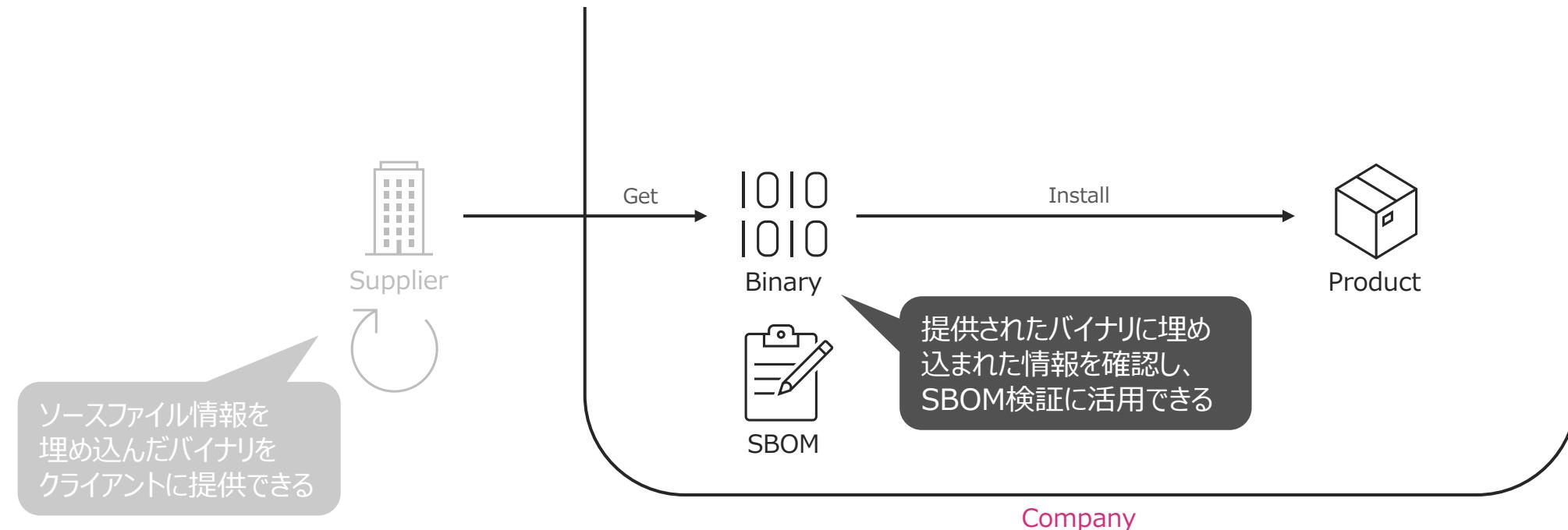
If you want to update two or more binary files with two or more license information files at once, you can specify them all on the command line:

```
$ esstra.py update <binary> [<binary> ...] -i <spdx-tv-file> [<spdx-tv-file> ..]
```

For more details of the command `update`, please refer to documents of the samples stored in the directory [./samples](#).

使い方

- シナリオ：提供されたバイナリの受け入れ確認をする
ESSTRA Utility (Python Script) を用いてバイナリに埋め込まれた情報を
① 表示する
② 削除する



使い方

- シナリオ：提供されたバイナリの受け入れ確認をする
ESSTRA Utility (Python Script) を用いてバイナリに埋め込まれた情報を
① 表示する
② 削除する

Command "show"

A command line:

```
$ esstra.py show <binary> [<binary>...]
```

outputs metadata embedded in specified binary files in YAML format. For example, passing a binary file `hello` built from `hello.c` as in [./samples/sample-hello](#) to the command:

```
$ esstra.py show hello
```

would give you an output as follows:

バイナリhelloに対するshowコマンド実行例

```
#  
# BinaryFileName: hello  
# BinaryPath: /home/snagao/esstra/samples/hello/hello  
#  
SourceFiles:  
/home/snagao/esstra/samples/hello:  
- File: hello.c  
SHA1: e7834d0b9cb6cb116c72f0bee7da29e3d280b27e  
LicenseInfo:  
- MIT  
/usr/include:  
- File: features-time64.h  
SHA1: 57c3c8093c3af70e5851f6d498600e2f6e24fdeb  
- File: features.h  
SHA1: d8725bb98129d6d70ddcbf010021c2841db783f7  
- File: stdc-predef.h  
SHA1: 2fef05d80514ca0be77efec90bda051cf87d771f  
:  
(snip)  
:
```

使い方

- シナリオ：提供されたバイナリの受け入れ確認をする

ESSTRA Utility (Python Script) を用いてバイナリに埋め込まれた情報を

- ① 表示する
- ② 削除する

Command "strip"

The command `strip` completely removes metadata embedded in binary files by the ESSTRA Core during compilation.

```
$ esstra.py strip <binary> [<binary> ...]
```



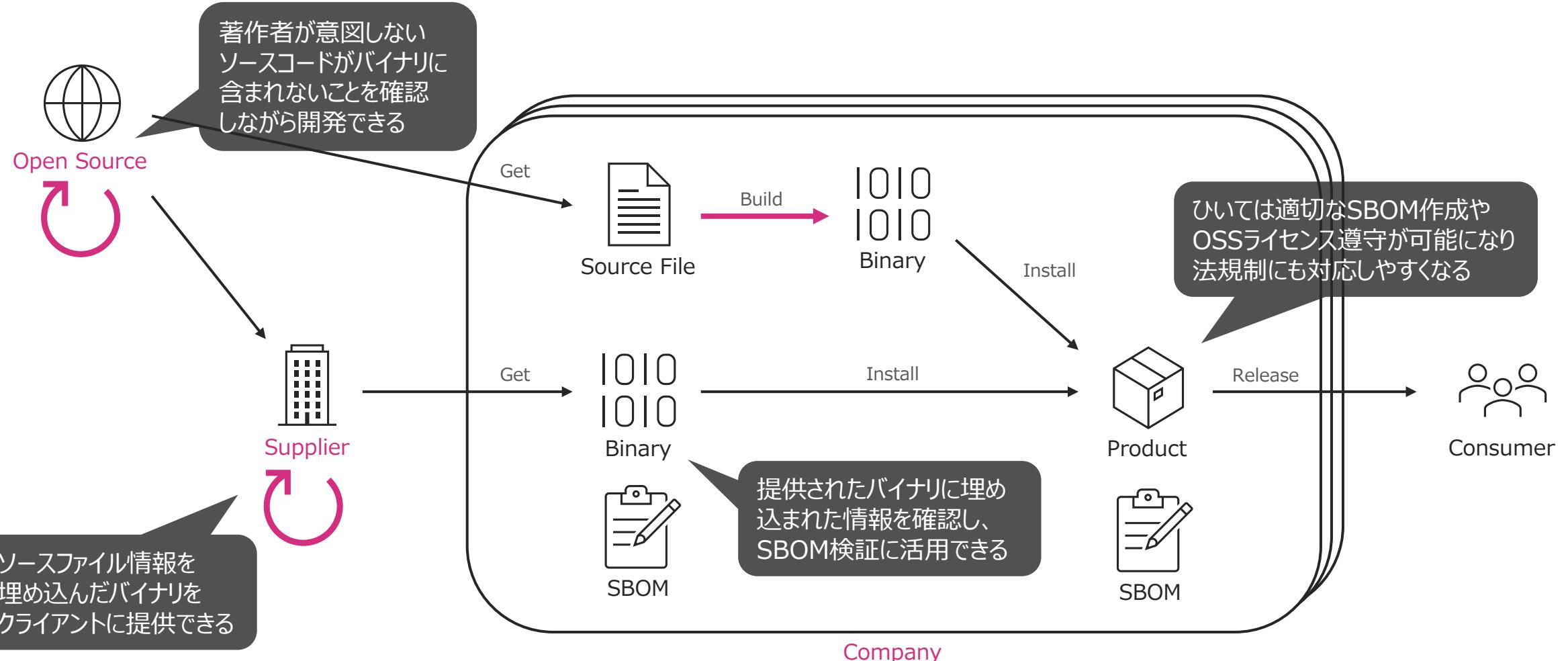
When a binary file with metadata removed is specified for the ESSTRA Utility, an error occurs as shown below:

```
$ esstra.py strip hello
* processing 'hello'...
* done.
$ esstra.py show hello
#
# BinaryFileName: hello
#
Exception: RuntimeError("section not found: 'esstra_info'")
```



(再掲) なぜESSTRAをオープンソースにしたのか?

- ・ ソフトウェアサプライチェーンのどこでも、どのフェーズでも、活用できる、活用してほしい



今後の予定

- LLVM対応
- 他ツールとのコラボレーション

2024年11月のJapan SBOM Summitで
ソニー小保田氏がArmijn Hemel氏にESSTRAを紹介
Binary Analysis Next Generation (BANG) で
ESSTRAのELFセクションを読み取り可能に

<https://github.com/armijnhemel/binaryanalysis-ng>

The screenshot shows a GitHub repository page for 'armijnhemel / binaryanalysis-ng'. The 'Code' tab is selected. A commit is shown with the message: 'elf: flag 'esstra_info' sections as interesting. See https://github.com/sony/esstra/'. The commit was made by 'armijnhemel' on Nov 13, 2024, with 1 parent and a commit hash of 5d0e703. Below the commit message is a diff view for the file 'src/bang/parsers/executable/elf/UnpackParser.py'. The diff highlights changes in line 380, where the code has been modified from a single-line if statement to a multi-line if statement that includes the 'interesting' variable assignment.

```
elf: flag 'esstra_info' sections as interesting. See https://github.com/sony/esstra/
...
diff --git a/src/bang/parsers/executable/elf/UnpackParser.py b/src/bang/parsers/executable/elf/UnpackParser.py
--- a/src/bang/parsers/executable/elf/UnpackParser.py
+++ b/src/bang/parsers/executable/elf/UnpackParser.py
@@ -377,7 +377,8 @@ def unpack(self, meta_directory):
    # * .BTF and .BTF.ext: eBPF related files
    # * .rom_info: Mediatek preloader(?)
    # * .init.data: Linux kernel init data, sometimes contains initial ramdisk
-   if header.name in ['.gnu_debugdata', '.qtmimedatabase', '.BTF', '.BTF.ext', '.rom_info', '.init.data']:
+   if header.name in ['.gnu_debugdata', '.qtmimedatabase', '.BTF', '.BTF.ext',
+                     '.rom_info', '.init.data', 'esstra_info']:
        interesting = True

```

Q&A

ありがとうございました

SONY

SONY is a registered trademark of Sony Group Corporation.

Names of Sony products and services are the registered trademarks and/or trademarks of Sony Group Corporation or its Group companies.

Other company names and product names are registered trademarks and/or trademarks of the respective companies.