SONY

Project ESSTRA

A Software Suite for
Enhancing Software Supply Chain Transparency

https://github.com/sony/esstra

ESSTRA

A Software Suite to Enhance Software Transparency and Traceability in Software Supply Chain

Aug. 25, 2025

Open Source Summit Europe 2025

Takuya NAMAE

Sony Group Corporation

©2025 Sony Group Corporation

Who am I?

- Takuya NAMAE
  - Joined Sony Corporation in 2003
  - History of software development
    - Web browser for embedded devices
    - EPUB3 viewer for e-book reader
    - etc.
  - Analyzing licenses of Chromium led to my deep involvement in open-source license compliance
  - Current responsibilities
    - Leading open-source license compliance (500+ src pkgs/year) for Linux-based system software across various Sony products
    - Leading the development of related tools such as ESSTRA
    - Teaching "Open Source Software Lecture: License Clearance Practical Course“ in Sony

©2025 Sony Group Corporation

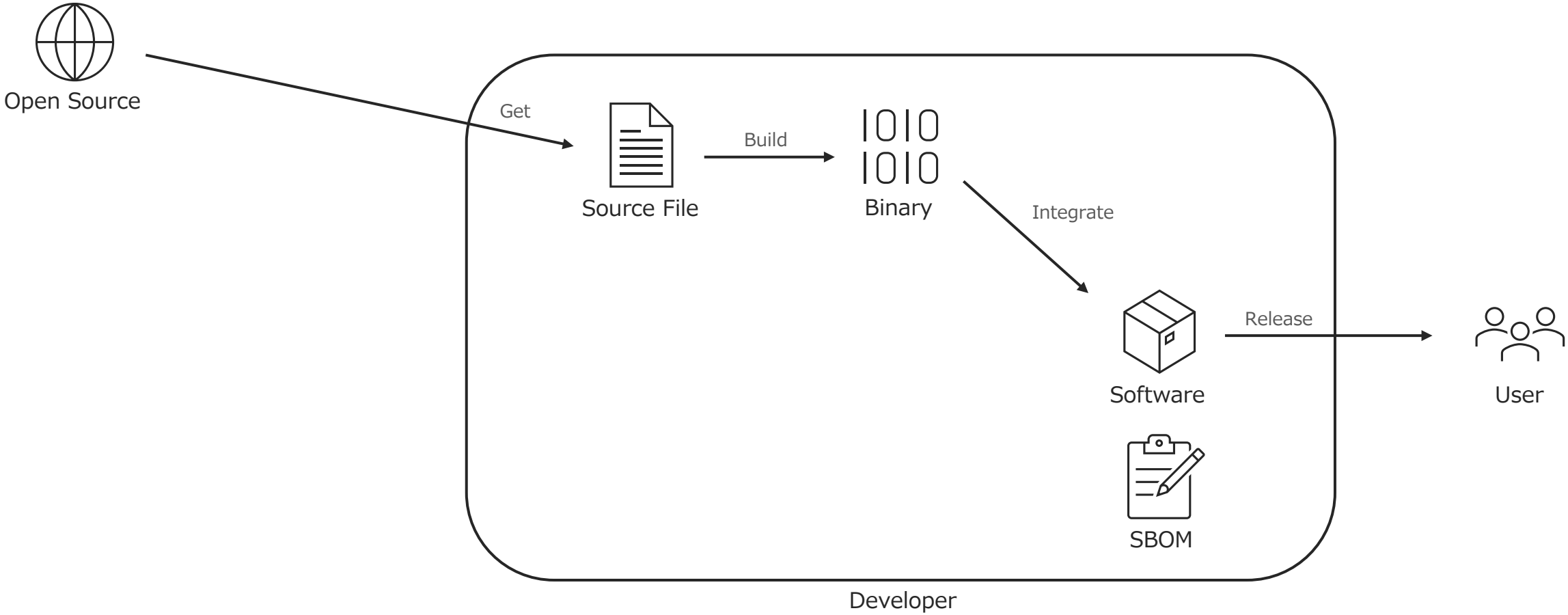
Agenda

- Introduction
- What's ESSTRA?
- Background and Goal
- Challenge and Solution
- Key Concepts
- Technical Overview
- Use Case
- Measured Impact
- Why We Open-Sourced
- Where to Get
- Future Work
- Q&A

©2025 Sony Group Corporation

Introduction

- When you use open-source software, do you ever wonder which source files were actually compiled into the binaries?

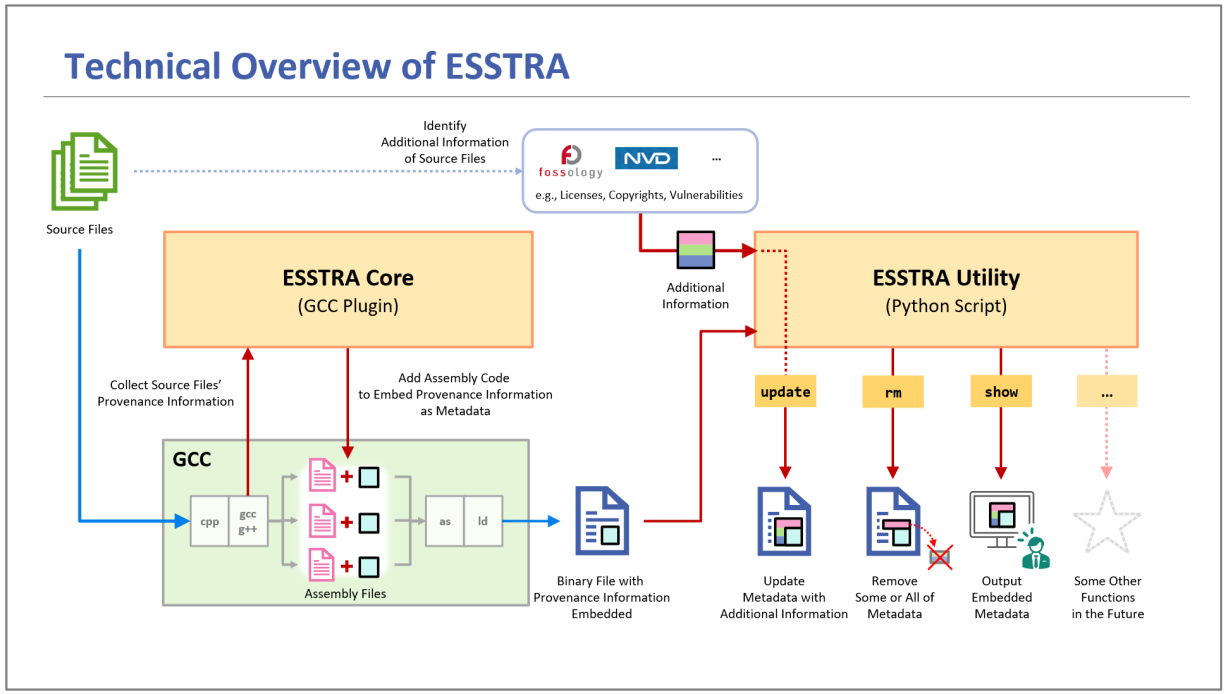


 ©2025 Sony Group Corporation

What's ESSTRA?

- A tool that collects information about the source files actually used during the build process of open-source software with GCC, and embeds that information into the resulting binary

https://github.com/sony/esstra (open-sourced in Oct. 2024)



Example: Showing information embedded in a binary file

```
#
# BinaryFileName: hello2
# BinaryPath: /home/tnamae/github.com/sony/esstra/samples/hello2/hello2
#
Headers:
  ToolName: ESSTRA Core
  ToolVersion: 0.1.1-develop
  DataFormatVersion: 0.1.0
  InputFileNames:
  - hello_main.c
  - hello_sub.c
SourceFiles:
- Directory: /home/tnamae/github.com/sony/esstra/samples/hello2
  Files:
  - File: hello_main.c
    SHA1: f7f5c447d68fd9685594a31cb10c8d8b1dd5ebd6
    LicenseInfo:
    - MIT
  - File: hello_sub.c
    SHA1: cfb72998ae0242237fa42c8bcf61ee5887137392
    LicenseInfo:
    - BSD-3-Clause
  - File: hello_sub.h
    SHA1: 3e5b3ed1aed966c0e0c183eac8fe6ea02dfa62a0
    LicenseInfo:
    - LGPL-2.1-or-later
- Directory: /usr/include
  Files:
  - File: features-time64.h
```

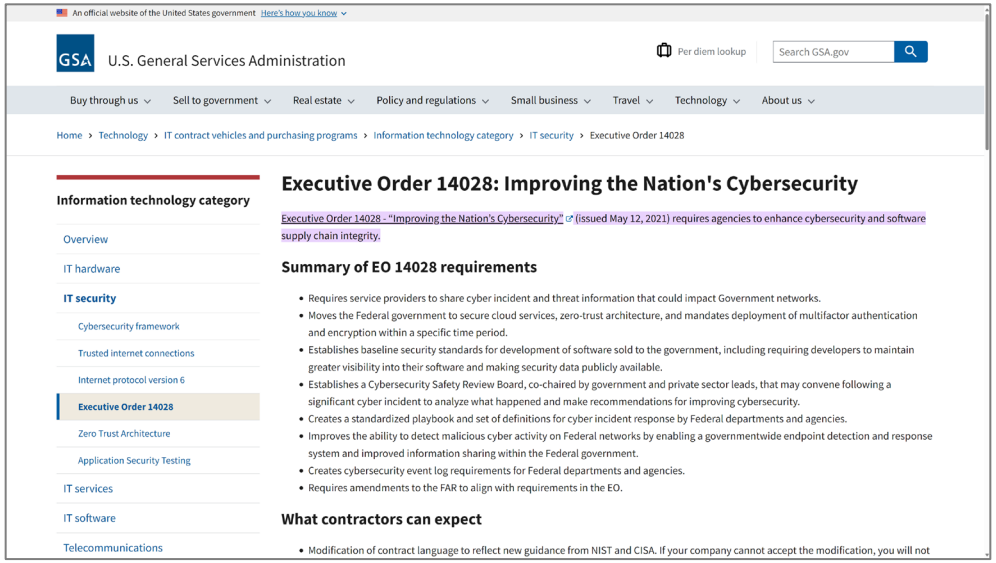
You can check which source files and licenses come with each binary

©2025 Sony Group Corporation

Background: Software Bills of Materials (SBOMs)

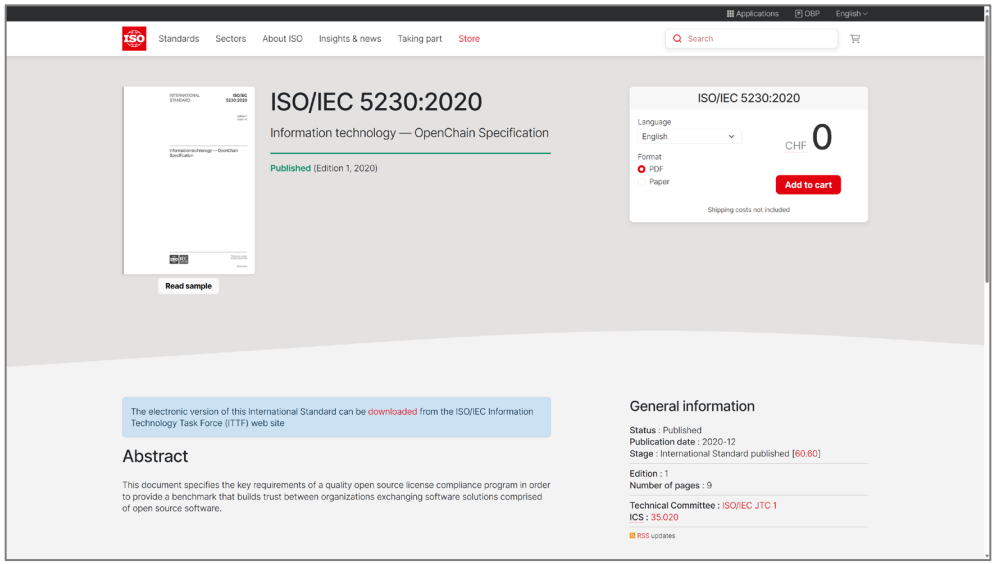
- The importance of SBOMs is growing, driven by the need for greater transparency and traceability in software supply chains, especially for vulnerability management and license compliance

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



The Executive Order on Improving the Nation's Cybersecurity, issued in the United States in May 2021, brought attention to SBOMs, leading to active efforts in developing specifications and exploring their use

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

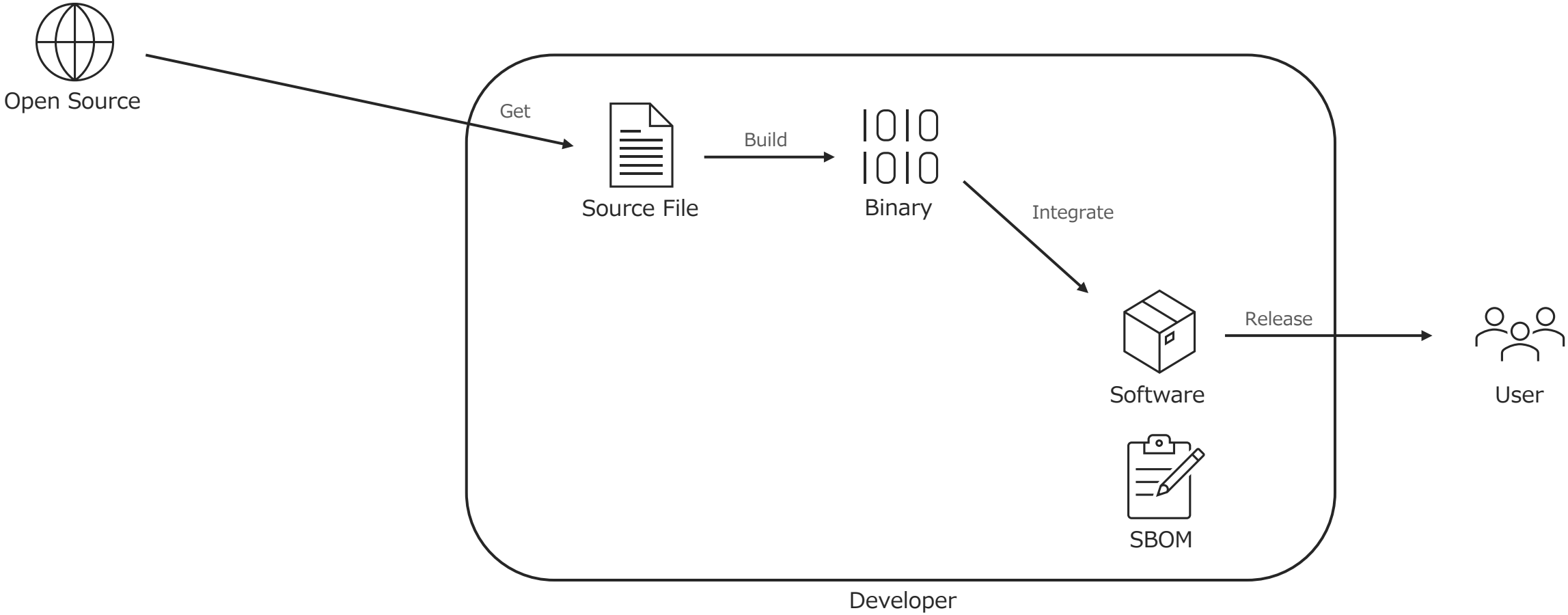


The OpenChain Project, under The Linux Foundation, defined the requirements for organizations to properly implement open-source license compliance, which was approved as the international standard for the compliance, ISO/IEC 5230:2020, in December 2020

 ©2025 Sony Group Corporation

Background: Software Development Using Open Source

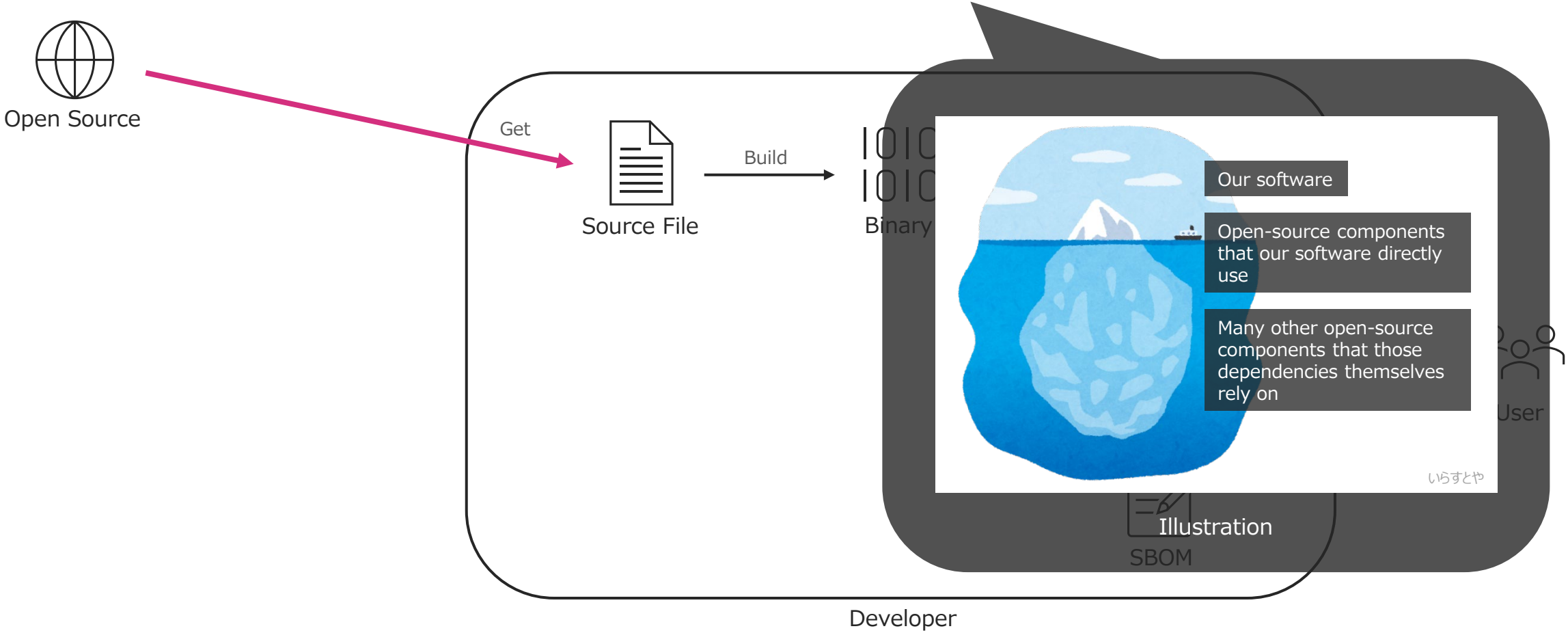
- Let’s look back from several perspectives



 ©2025 Sony Group Corporation

Background: Software Development Using Open Source

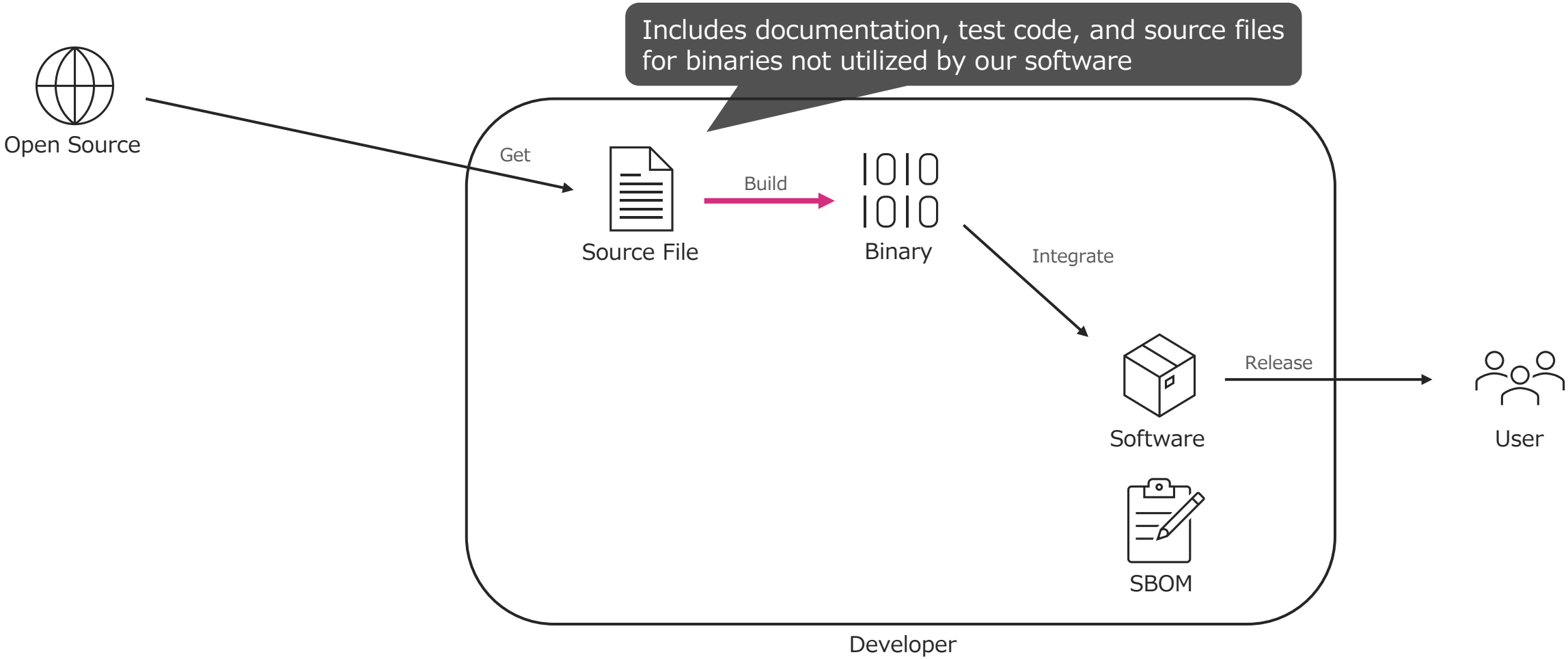
- We rely on a wide range of open-source software,
  both directly and through a cascading chain of dependencies



 ©2025 Sony Group Corporation

Background: Software Development Using Open Source

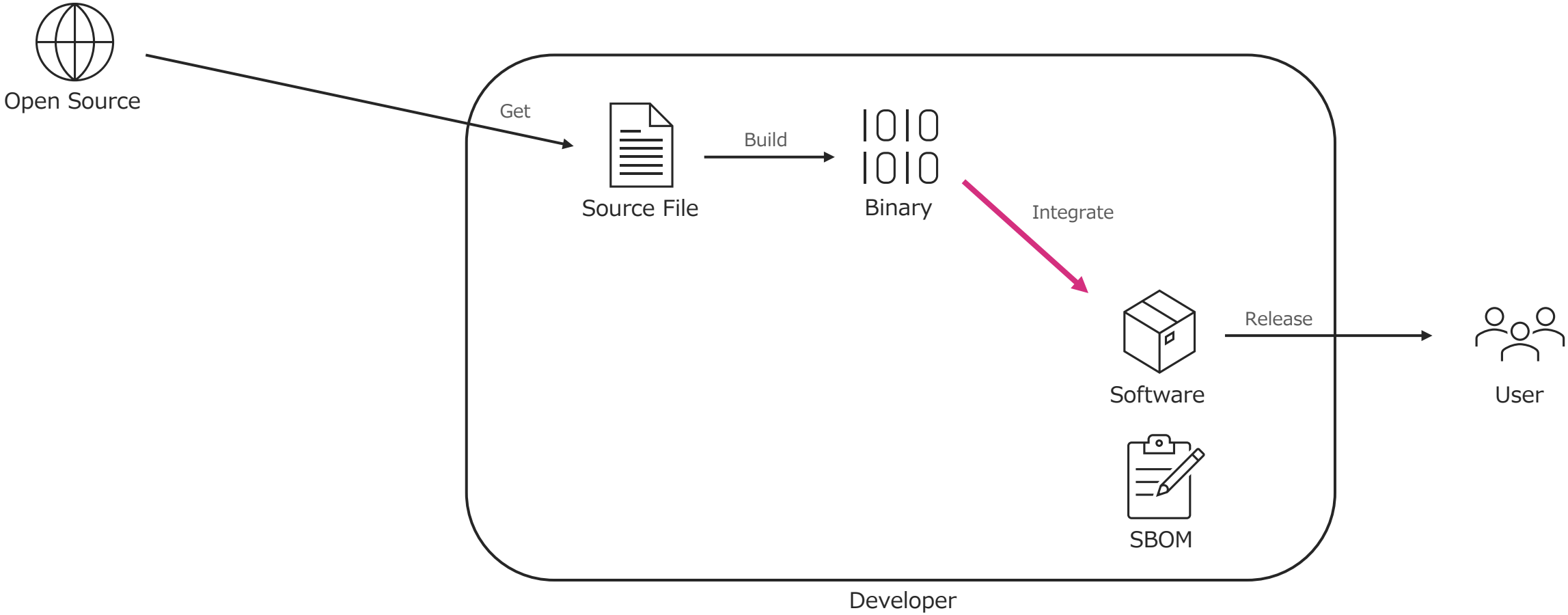
- Not all files are included in the binary our software use



 ©2025 Sony Group Corporation

Background: Software Development Using Open Source

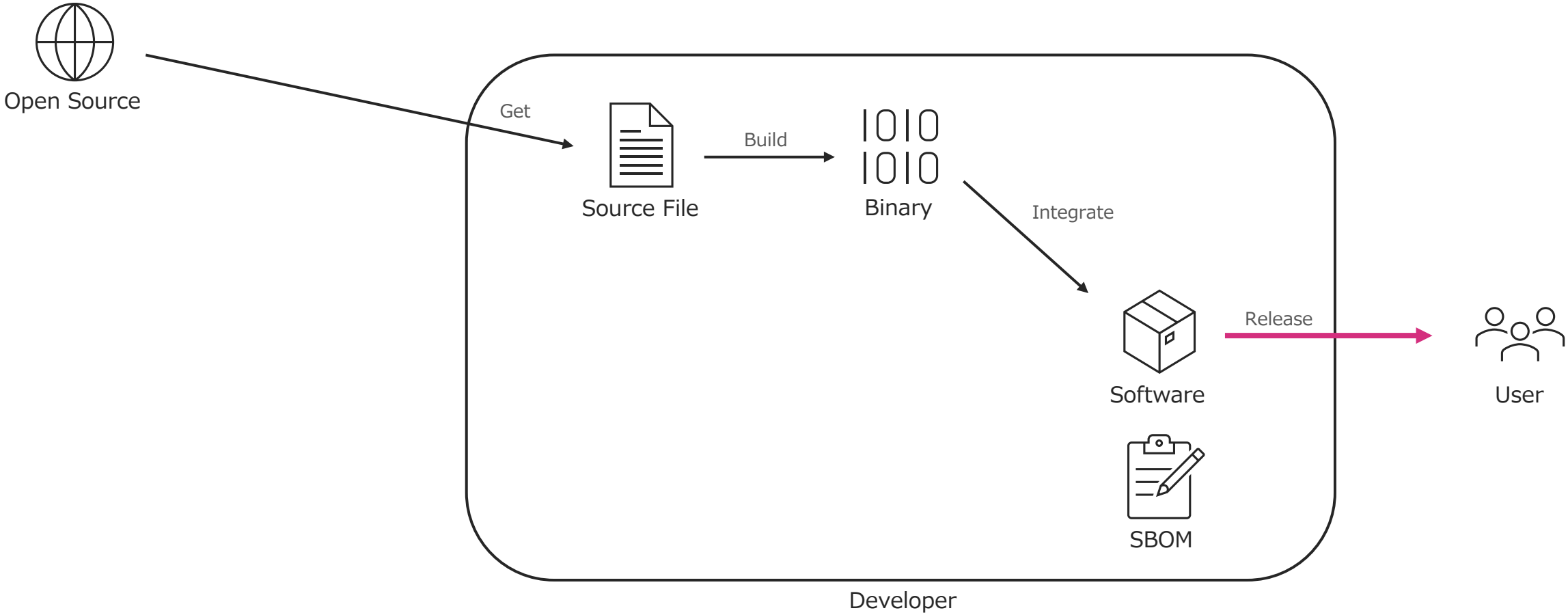
- What is included in our software = What is listed in the SBOM



 ©2025 Sony Group Corporation

Background: Software Development Using Open Source

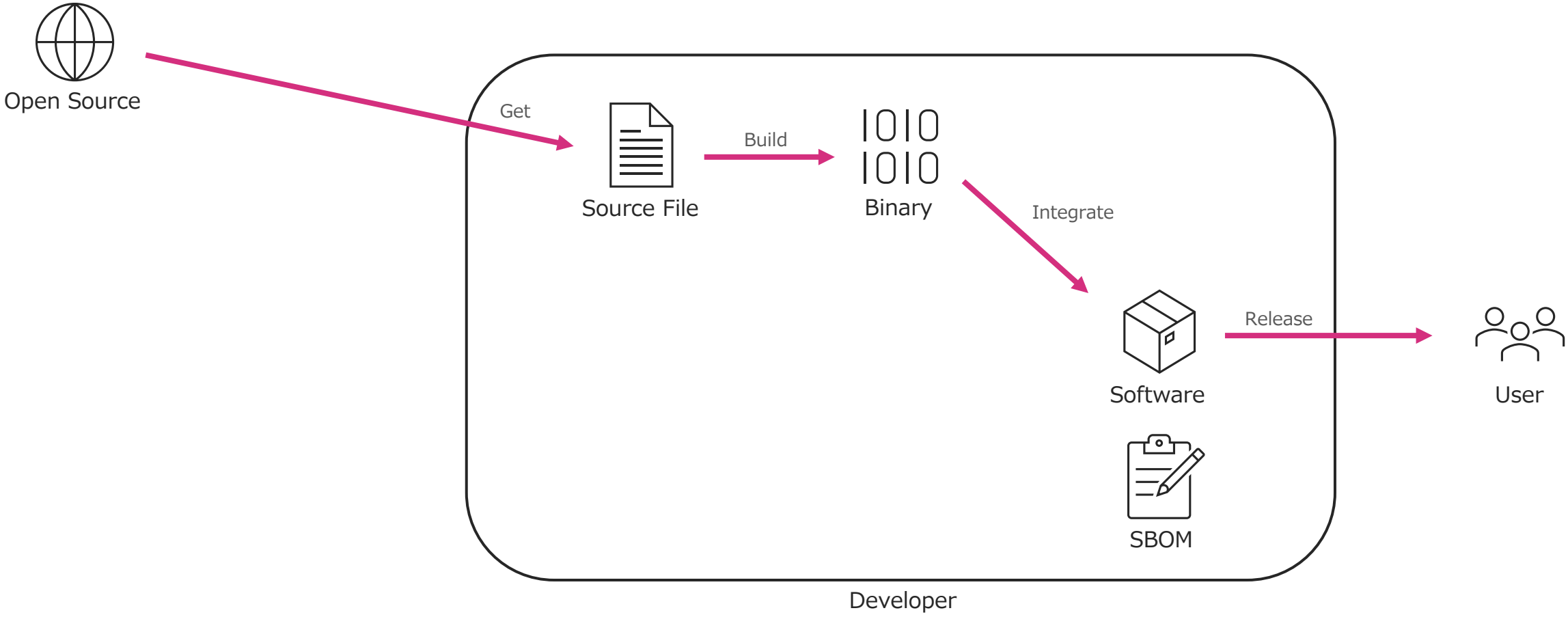
- What is included in our software = What is listed in the SBOM
  = What must comply with open-source licenses



 ©2025 Sony Group Corporation

Goal

- Technically ensure efficient and effective alignment between software content, SBOM, and open-source license compliance

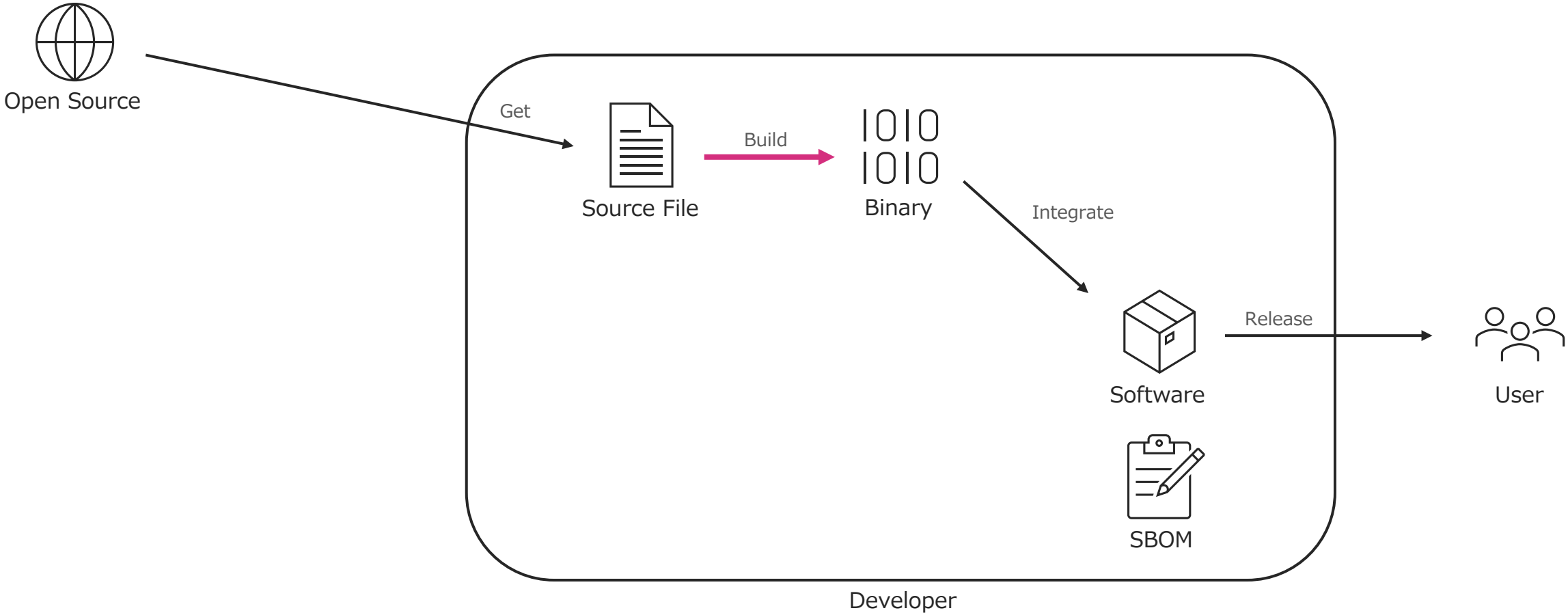


 ©2025 Sony Group Corporation

Challenge

Today's focus

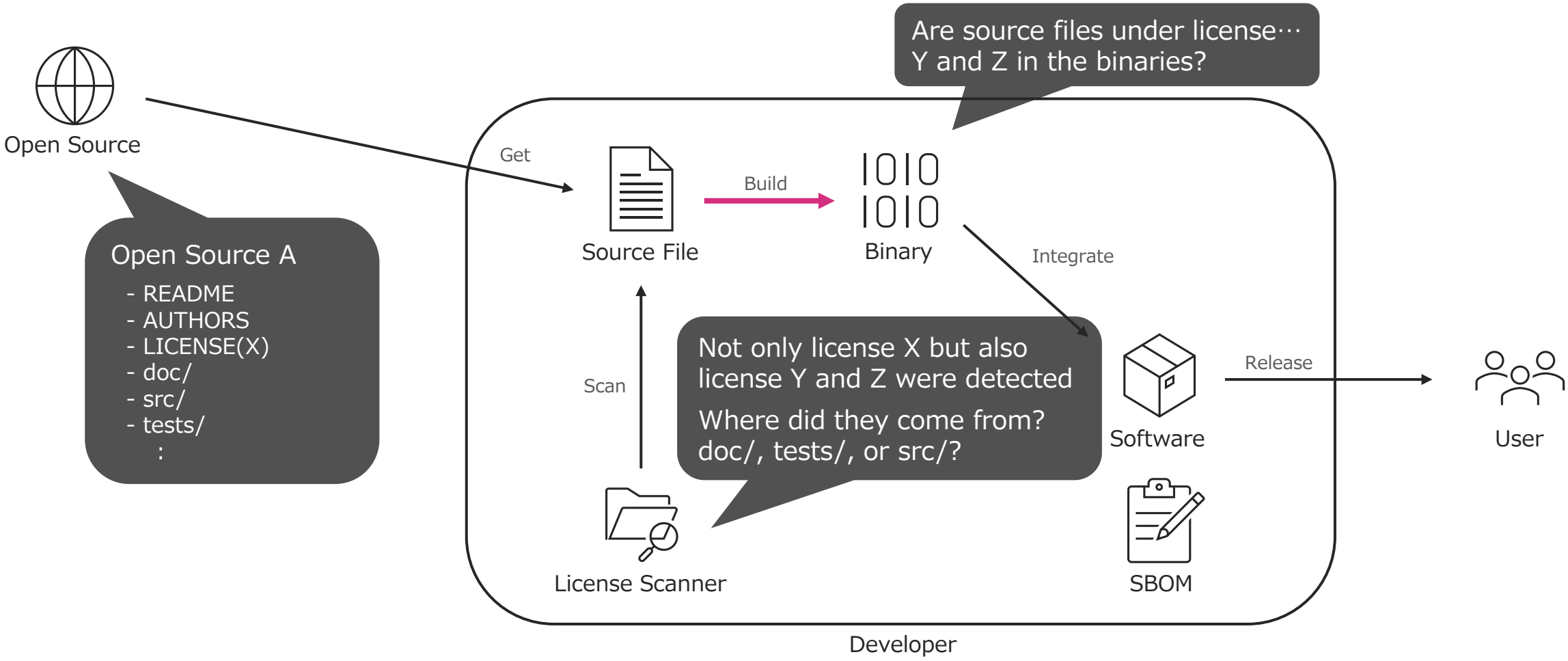
- It is difficult to accurately determine the provenance of binaries included in our software



 ©2025 Sony Group Corporation

Challenge: Case Study 1

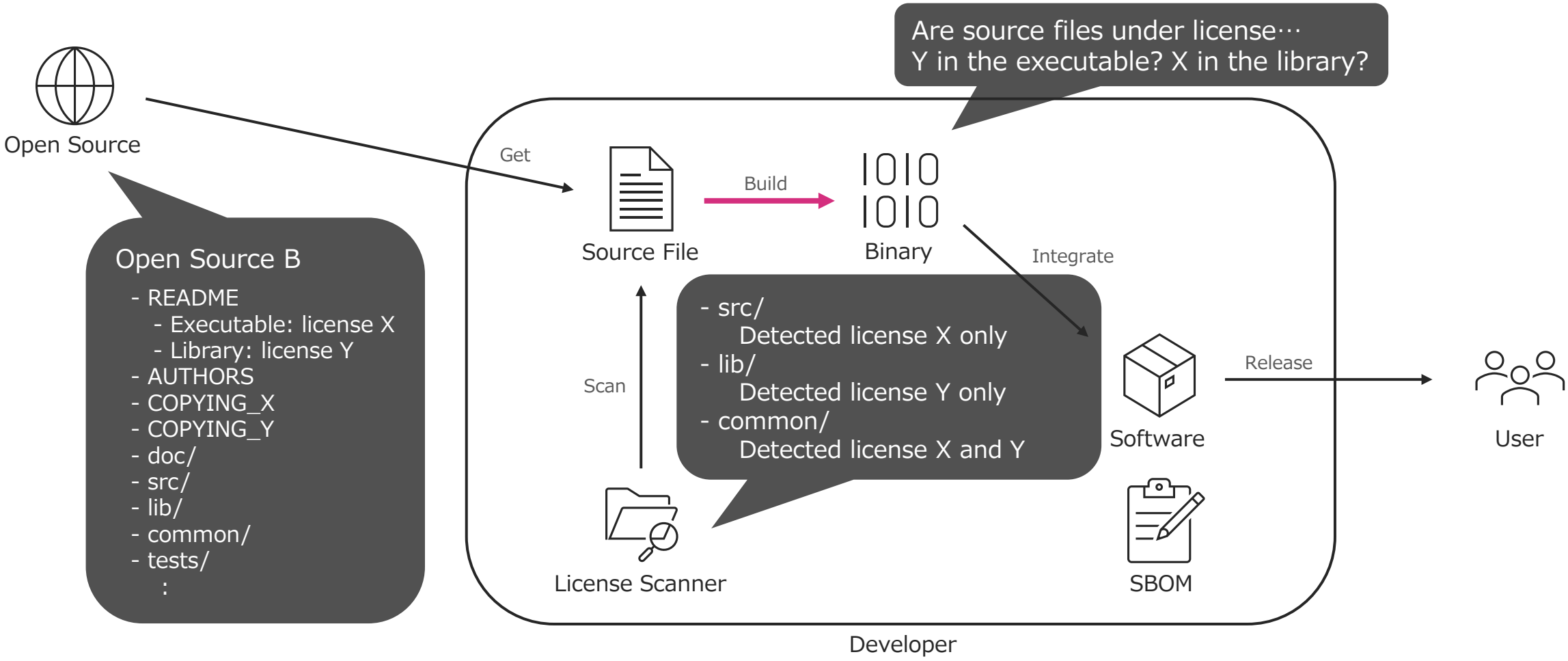
- Binaries contain licenses not declared by the original authors



 ©2025 Sony Group Corporation

Challenge: Case Study 2

- Binaries contain licenses not declared by the original authors

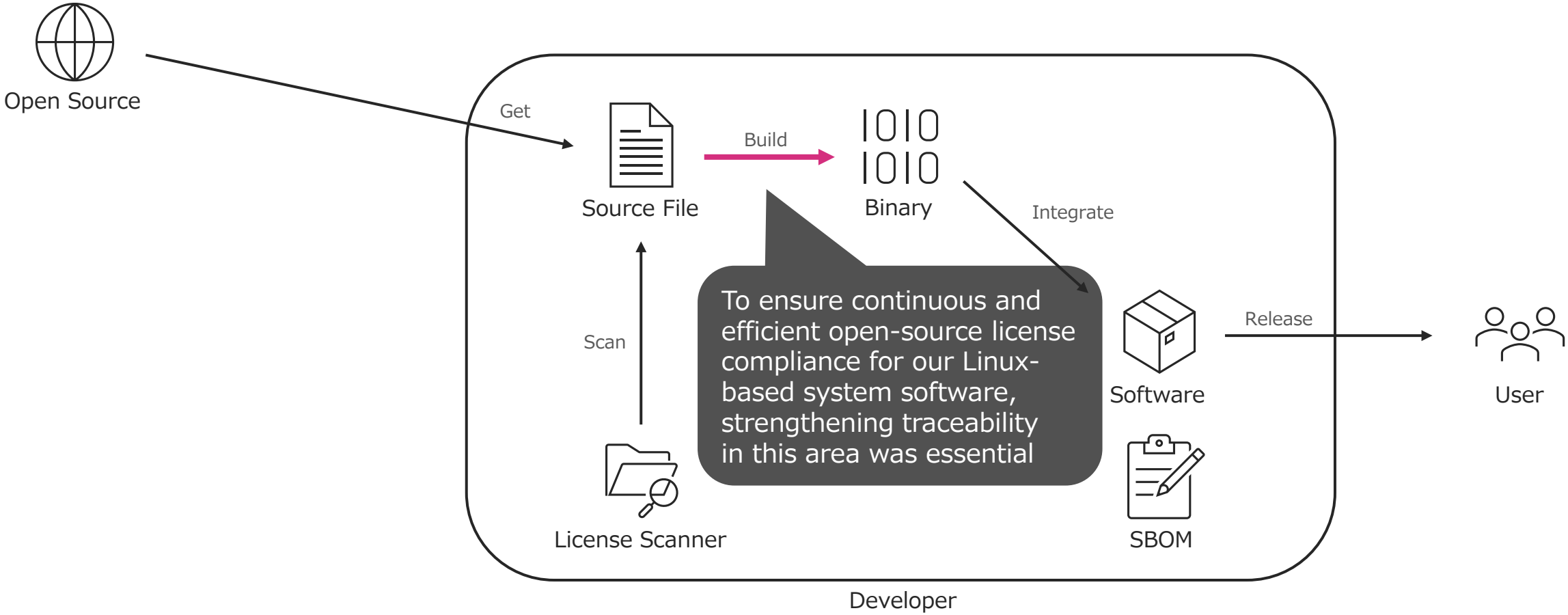


 ©2025 Sony Group Corporation

Challenge: Key to Solving

From the “Introduction” slide on p. 4

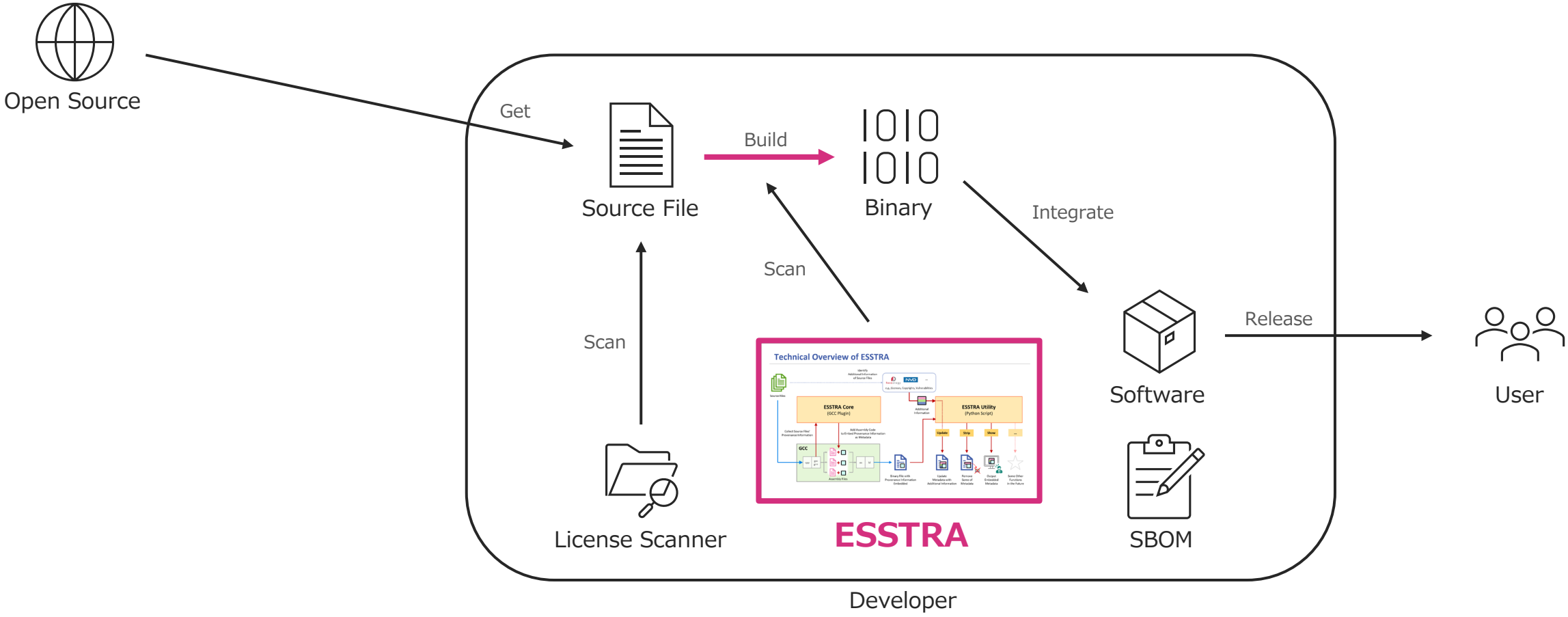
- When you use open-source software, do you ever wonder which source files were actually compiled into the binaries?



 ©2025 Sony Group Corporation

Solution: ESSTRA

- Enhancing Software Supply Chain Transparency and Traceability

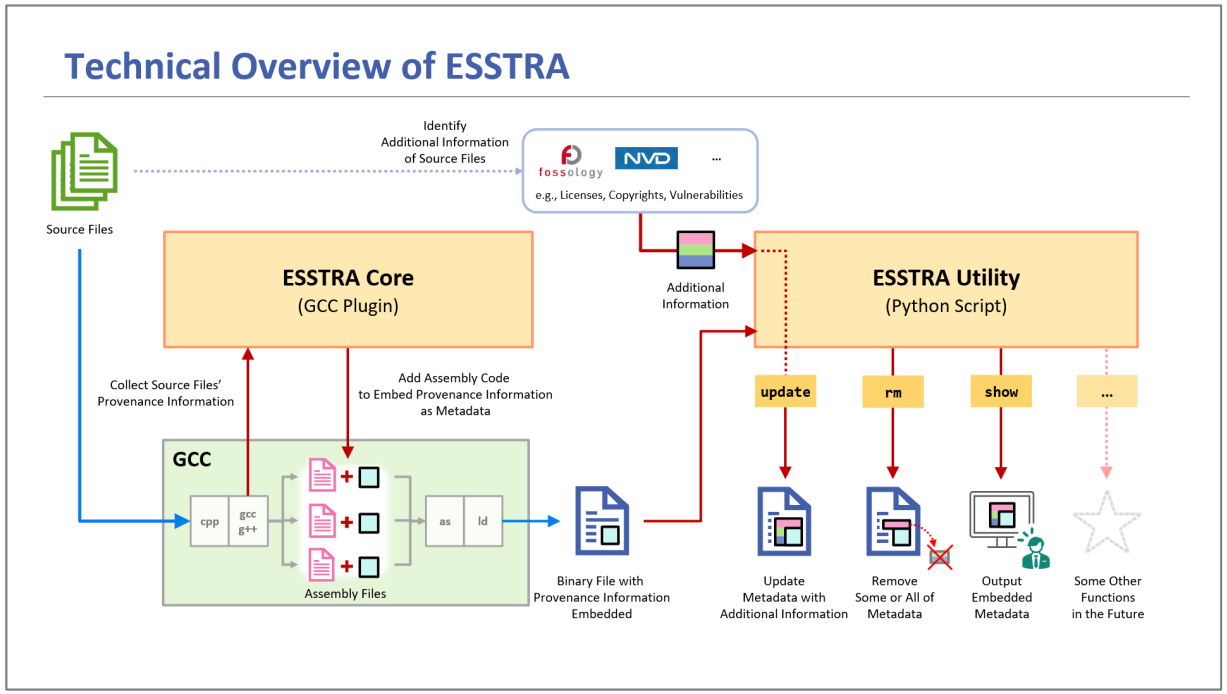


 ©2025 Sony Group Corporation

Back to: What's ESSTRA?

- A tool that collects information about the source files actually used during the build process of open-source software with GCC, and embeds that information into the resulting binary

https://github.com/sony/esstra (open-sourced in Oct. 2024)



Example: Showing information embedded in a binary file

```
#
# BinaryFileName: hello2
# BinaryPath: /home/tnamae/github.com/sony/esstra/samples/hello2/hello2
#
Headers:
  ToolName: ESSTRA Core
  ToolVersion: 0.1.1-develop
  DataFormatVersion: 0.1.0
  InputFileNames:
  - hello_main.c
  - hello_sub.c
SourceFiles:
- Directory: /home/tnamae/github.com/sony/esstra/samples/hello2
  Files:
  - File: hello_main.c
    SHA1: f7f5c447d68fd9685594a31cb10c8d8b1dd5ebd6
    LicenseInfo:
    - MIT
  - File: hello_sub.c
    SHA1: cfb72998ae0242237fa42c8bcf61ee5887137392
    LicenseInfo:
    - BSD-3-Clause
  - File: hello_sub.h
    SHA1: 3e5b3ed1aed966c0e0c183eac8fe6ea02dfa62a0
    LicenseInfo:
    - LGPL-2.1-or-later
- Directory: /usr/include
  Files:
  - File: features-time64.h
```

You can check which source files and licenses come with each binary

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate
2. No impact on the execution of binaries,
   minimal changes to build settings
3. Structured metadata suitable for SBOM use
4. SBOM data can be retained in release binaries
5. Post-processing support for additional metadata

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate
   - ESSTRA collects source file hashes during compilation, independent of the build system
   - Ensures reliable evidence of file identity, even in complex or customized environments
   - Embeds the collected metadata directly into the binary during compilation, ensuring it travels together as verifiable evidence
   - Easy to integrate into existing CI/CD systems and workflows, with minimal setup and impact
   - Avoids reliance on large frameworks or special environments, offering a simple and portable structure
2. No impact on the execution of binaries, minimal changes to build settings
3. Structured metadata suitable for SBOM use
4. SBOM data can be retained in release binaries
5. Post-processing support for additional metadata

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate

2. No impact on the execution of binaries,
   minimal changes to build settings
   - ESSTRA Core works as a GCC plugin and collects source file information during compilation
   - Requires only a small change: add the -fplugin option to the build configuration
   - Adds a dedicated ELF section (.esstra); no changes to executable code or linking
   - Works with both production and debug builds without affecting binary quality
   - In production builds, ESSTRA can avoid binary behavior changes caused by debug options

3. Structured metadata suitable for SBOM use

4. SBOM data can be retained in release binaries

5. Post-processing support for additional metadata

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate

2. No impact on the execution of binaries,
   minimal changes to build settings

3. Structured metadata suitable for SBOM use
   - Collected source file information is structured in YAML format and stored in the .esstra section
   - Records source file names, hashes, license information, etc.
   - Compatible with SPDX and other SBOM tools

4. SBOM data can be retained in release binaries

5. Post-processing support for additional metadata

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate

2. No impact on the execution of binaries,
   minimal changes to build settings

3. Structured metadata suitable for SBOM use

4. SBOM data can be retained in release binaries
   - The .esstra section, unlike debug info, is not stripped and remains in production build binaries
   - Embedding SBOM metadata directly into the binary allows it to be delivered together with the product
   - When using the binary in production, the .esstra section can be safely removed without side effects

5. Post-processing support for additional metadata

©2025 Sony Group Corporation

Key Concepts

1. Build framework independent and easy to integrate

2. No impact on the execution of binaries,
   minimal changes to build settings

3. Structured metadata suitable for SBOM use

4. SBOM data can be retained in release binaries

5. Post-processing support for additional metadata
   - ESSTRA Utility (Python Script) can update or remove the .esstra section in the binary
   - Additional metadata—such as license or vulnerability data—can be linked to the source file information stored in .esstra after the build

©2025 Sony Group Corporation

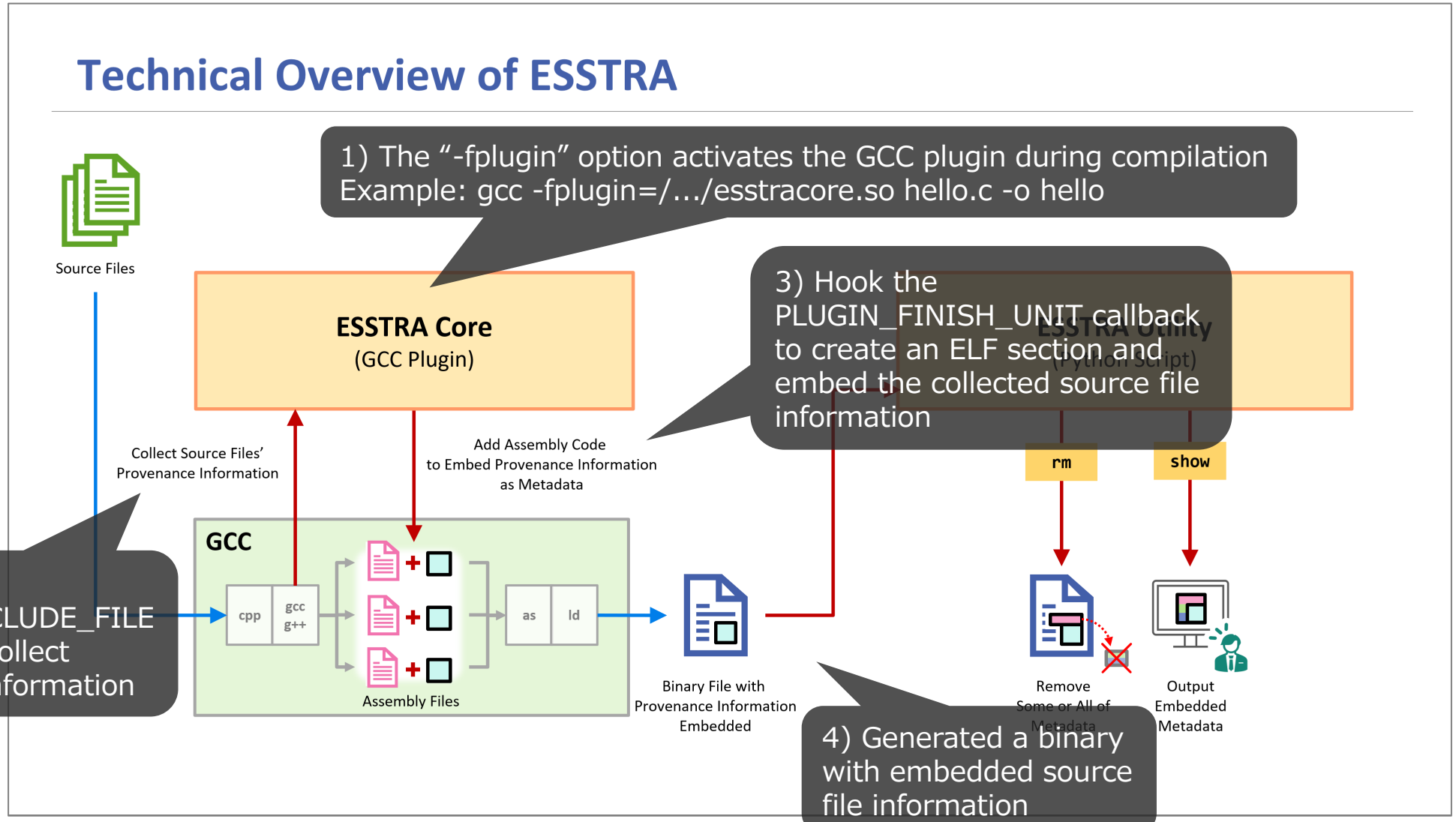
Key Concepts

1. Build framework independent and easy to integrate
   - ESSTRA collects source file hashes during compilation, independent of the build system
   - Ensures reliable evidence of file identity, even in complex or customized environments
   - Embeds the collected metadata directly into the binary during compilation, ensuring it travels together as verifiable evidence
   - Easy to integrate into existing CI/CD systems and workflows, with minimal setup and impact
   - Avoids reliance on large frameworks or special environments, offering a simple and portable structure
2. No impact on the execution of binaries, minimal changes to build settings
   - ESSTRA Core works as a GCC plugin and collects source file information during compilation
   - Requires only a small change: add the -fplugin option to the build configuration
   - Adds a dedicated ELF section (.esstra); no changes to executable code or linking
   - Works with both production and debug builds without affecting binary quality
   - In production builds, ESSTRA can avoid binary behavior changes caused by debug options
3. Structured metadata suitable for SBOM use
   - Collected source file information is structured in YAML format and stored in the .esstra section
   - Records source file names, hashes, license information, etc.
   - Compatible with SPDX and other SBOM tools
4. SBOM data can be retained in release binaries
   - The .esstra section, unlike debug info, is not stripped and remains in production build binaries
   - Embedding SBOM metadata directly into the binary allows it to be delivered together with the product
   - When using the binary in production, the .esstra section can be safely removed without side effects
5. Post-processing support for additional metadata
   - ESSTRA Utility (Python Script) can update or remove the .esstra section in the binary
   - Additional metadata—such as license or vulnerability data—can be linked to the source file information stored in .esstra after the build

©2025 Sony Group Corporation

Technical Overview: Basics - Collecting Source File Info

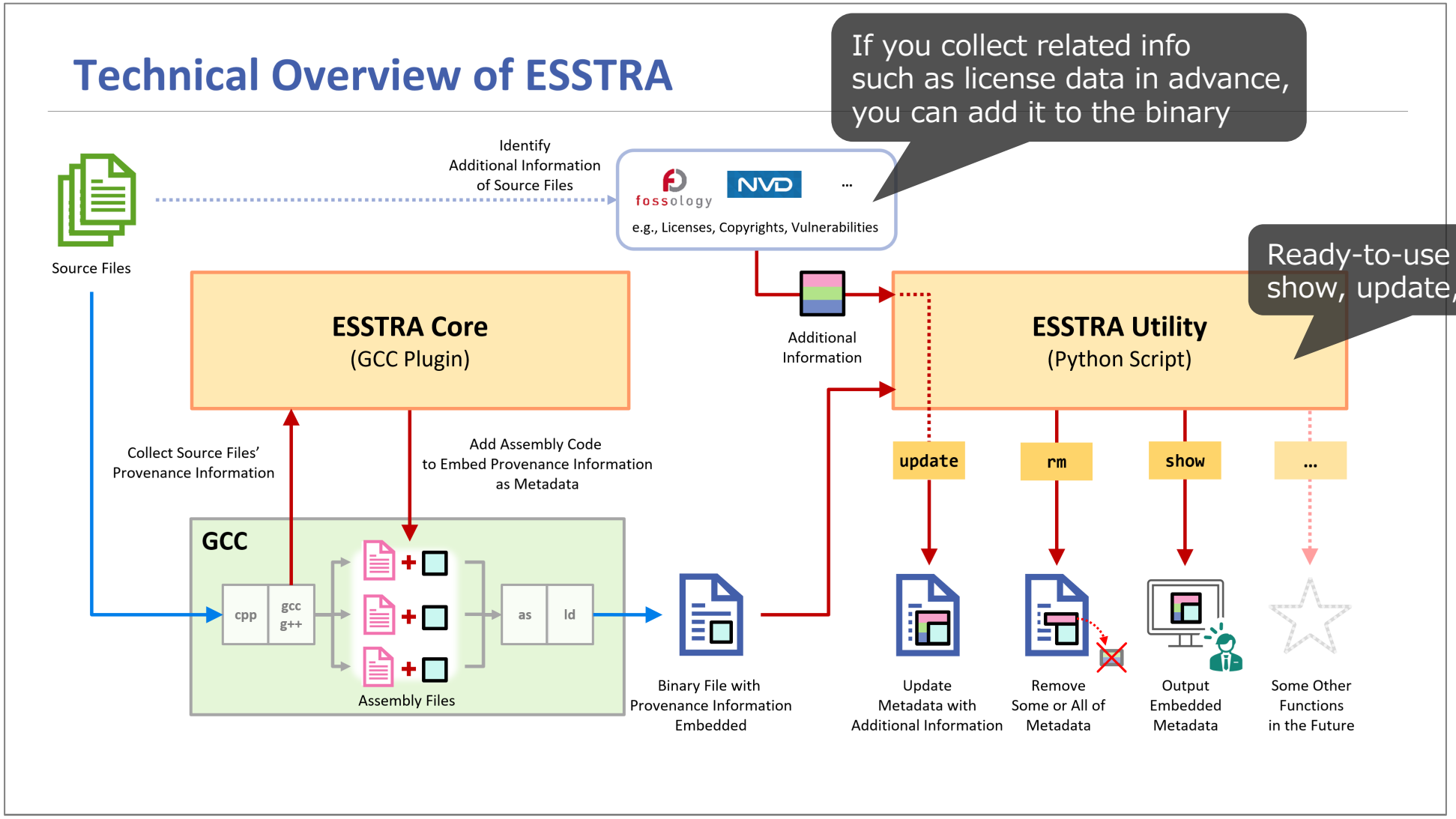
https://github.com/sony/esstra/tree/main/core



 ©2025 Sony Group Corporation

Technical Overview: Advanced - Adding Related Info

https://github.com/sony/esstra/tree/main/util



 ©2025 Sony Group Corporation

ELF Section Comparison for Embedding ESSTRA Data

Comparison Item	.note.esstra(SHT_NOTE)	.esstra(SHT_PROGBITS)	.debug_info(SHT_PROGBITS)	Key Stakeholders
Primary Purpose	👍 Lightweight identifiers	✅ Detailed SBOM composition	⚠️ Debugging support	SBOM creators, integrators
Identifiability	✅ Clear via name + type	👍 Identifiable via section name	⚠️ Requires DWARF-aware tools	SBOM creators, auditors
Structured Data Support	⚠️ Limited by NOTE format (custom format needed for SBOM)	✅ Supports YAML/JSON freely	⚠️ DWARF is complex and non-structured	CI/CD engineers, SBOM creators
Extensibility	⚠️ Multiple entries possible, but limited flexibility	✅ Fully extensible	⚠️ Bound by DWARF specification	CI/CD engineers
Size Tolerance	⚠️ Realistically up to a few hundred KB	✅ Several KB to tens of KB (compression possible)	⚠️ Can grow to several MB	Integrators
Production Build Integration	👍 Possible (for lightweight identifiers)	✅ Suitable for embedding SBOM data	⚠️ Not suitable (typically stripped)	Integrators
Debug Build Integration	✅ Fully supported	✅ Fully supported	✅ Fully supported	Developers, CI/CD engineers
Impact on Binary	👍 Added to NOTE section, may conflict with other entries	✅ Dedicated section, isolated from existing code	⚠️ Merged with debug info, may increase complexity	Integrators, CI/CD engineers
Runtime Memory Impact	✅ Not loaded (no SHF_ALLOC)	✅ Not loaded (no SHF_ALLOC)	✅ Not loaded (typically stripped)	Integrators
Ownership & Control	⚠️ May be auto-generated by toolchain	✅ Fully developer-controlled	⚠️ Compiler-generated, hard to control	CI/CD engineers, SBOM creators
Retention Target	👍 Can be retained in product binaries (lightweight info)	✅ Should be retained as SBOM evidence	⚠️ Temporary, usually stripped	Integrators, auditors
Maintainability	⚠️ Rigid structure, hard to extend	✅ Easy to maintain, CI/CD friendly	⚠️ Complex and hard to maintain	CI/CD engineers
Tool Compatibility	👍 Recognized by readelf/objdump (may display incorrectly)	👍 Supported via ESSTRA Utility	👍 Widely supported by gdb (not SBOM-oriented)	Auditors, external tools
Standardization & Future Outlook	✅ ELF-compliant	👍 ELF-compliant, adaptable to future .sbom section	✅ DWARF is standardized (for debugging)	All stakeholders

Legend: ✅ Excellent: strongly recommended 👍 Good: recommended ⚠️ Limited: acceptable with caveats

©2025 Sony Group Corporation

Why a dedicated ELF section: .esstra (SHT_PROGBITS)?

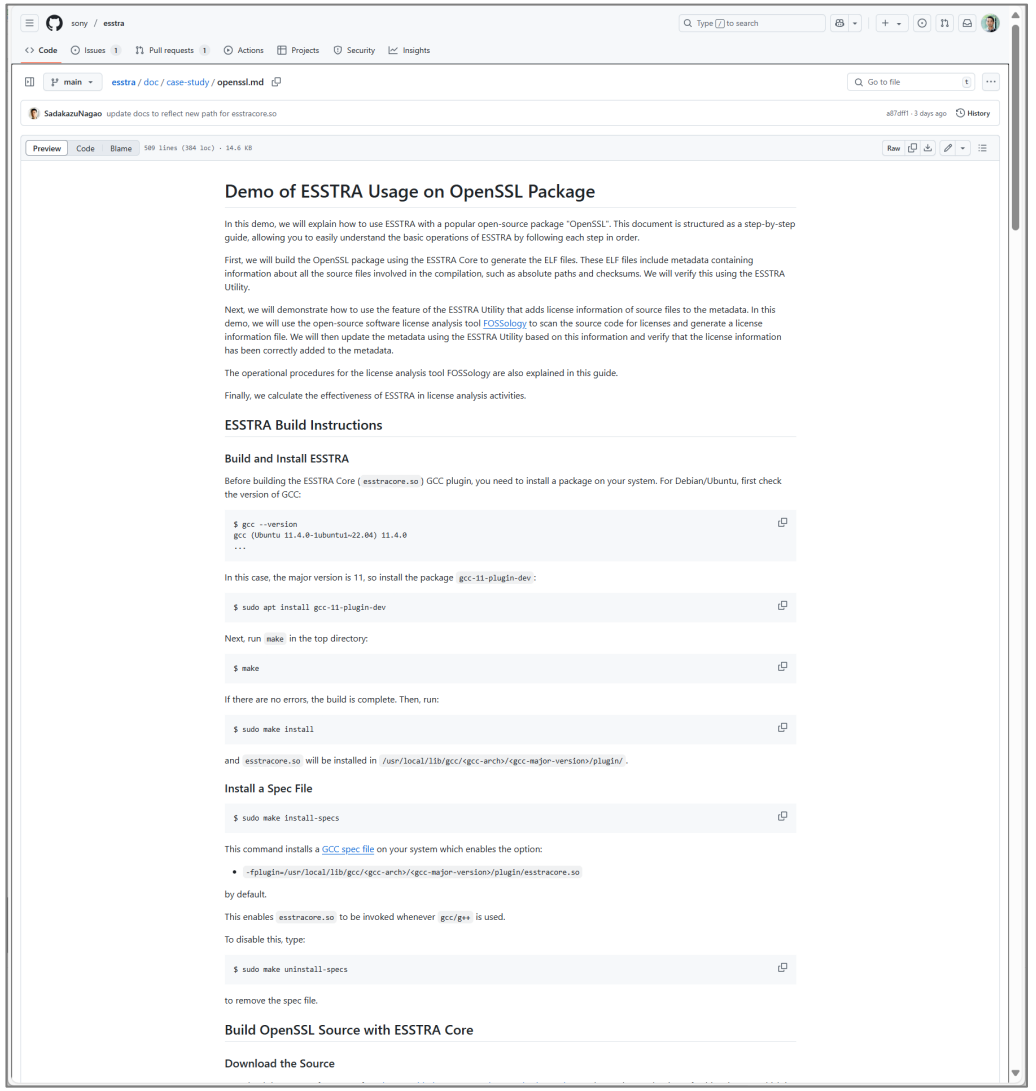
- Clean separation from existing sections, with no impact on executable code
  - .note (SHT_NOTE): Limited structure; may conflict with other NOTE entries
  - .debug_info (SHT_PROGBITS): Merged with debug info; can become structurally complex
- Flexible storage of structured metadata (YAML/JSON) for SBOM use
  - .note: Fixed format; custom structure needed for SBOM
  - .debug_info: DWARF format is complex and non-structured
- Retained after stripping, allowing SBOM data to be delivered with the binary
  - .note: Can be retained, but lacks detailed SBOM data
  - .debug_info: Typically stripped; not suitable for production
- Adaptable to future standards, such as a possible .sbom section

©2025 Sony Group Corporation

Use Case: OpenSSL

- Process Overview
  1. Build and install ESSTRA
  2. Build OpenSSL using ESSTRA
  3. Analyze licenses with license scanner
  4. Add license information to metadata

https://github.com/sony/esstra/blob/main/doc/case-study/openssl.md



sony / esstra

Code Issues 1 Pull requests 1 Actions Projects Security Insights

main esstra / doc / case-study / openssl.md

SadakazuNagao update docs to reflect new path for esstracore.so

Preview Code Blame 509 lines (384 loc) · 14.6 KB

Demo of ESSTRA Usage on OpenSSL Package

In this demo, we will explain how to use ESSTRA with a popular open-source package "OpenSSL". This document is structured as a step-by-step guide, allowing you to easily understand the basic operations of ESSTRA by following each step in order.

First, we will build the OpenSSL package using the ESSTRA Core to generate the ELF files. These ELF files include metadata containing information about all the source files involved in the compilation, such as absolute paths and checksums. We will verify this using the ESSTRA Utility.

Next, we will demonstrate how to use the feature of the ESSTRA Utility that adds license information of source files to the metadata. In this demo, we will use the open-source software license analysis tool FOSSology to scan the source code for licenses and generate a license information file. We will then update the metadata using the ESSTRA Utility based on this information and verify that the license information has been correctly added to the metadata.

The operational procedures for the license analysis tool FOSSology are also explained in this guide.

Finally, we calculate the effectiveness of ESSTRA in license analysis activities.

ESSTRA Build Instructions

Build and Install ESSTRA

Before building the ESSTRA Core (`esstracore.so`) GCC plugin, you need to install a package on your system. For Debian/Ubuntu, first check the version of GCC:

```
$ 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 `gcc-11-plugin-dev` :

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

Next, run `make` in the top directory:

```
$ make
```

If there are no errors, the build is complete. Then, run:

```
$ sudo make install
```

and `esstracore.so` will be installed in `/usr/local/lib/gcc/<gcc-arch>/<gcc-major-version>/plugin/` .

Install a Spec File

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

This command installs a GCC spec file on your system which enables the option:

- `-fplugin=/usr/local/lib/gcc/<gcc-arch>/<gcc-major-version>/plugin/esstracore.so`

by default.

This enables `esstracore.so` to be invoked whenever `gcc/g++` is used.

To disable this, type:

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

to remove the spec file.

Build OpenSSL Source with ESSTRA Core

Download the Source

©2025 Sony Group Corporation

Use Case: OpenSSL

- Step1. Build and install ESSTRA

```
~$ gcc --version
gcc (Debian 12.2.0-14) 12.2.0
~$ sudo apt install gcc-12-plugin-dev

esstra/core$ make
esstra/core$ ls -l esstracore.so
-rwxr-xr-x 1 tnamae tnamae 80104 Jun  2 10:40 esstracore.so

esstra/core$ sudo make install
esstra/core$ ls -l /usr/local/lib/gcc/x86_64-linux-gnu/12/plugin/
-rwxr-xr-x 1 root root 80104 Jun  2 10:43 esstracore.so

esstra/core$ sudo make install-specs
esstra/core$ grep esstra /usr/lib/gcc/x86_64-linux-gnu/12/specs
...(snip)... %{!fplugin:-fplugin=/usr/local/lib/gcc/x86_64-linux-gnu/12/plugin/esstracore.so}
```

The GCC spec file hooks ESSTRA Core (GCC plugin) into every gcc/g++ compilation

©2025 Sony Group Corporation

Use Case: OpenSSL

- Step2. Build OpenSSL using ESSTRA

```
~$ wget https://github.com/openssl/openssl-3.4.1/releases/download/openssl-3.4.1.tar.gz
~$ tar -xf openssl-3.4.1.tar.gz

openssl-3.4.1$ sudo apt install -y perl build-essential
openssl-3.4.1$ ./Configure no-tests
openssl-3.4.1$ make

openssl-3.4.1$ readelf -x .esstra libssl.so

Hex dump of section '.esstra':
  0x00000000 2d2d2d0a 48656164 6572733a 0a202054 ---.Headers:.  T
  0x00000010 6f6f6c4e 616d653a 20455353 54524120 oolName: ESSTRA
  0x00000020 436f7265 0a202054 6f6f6c56 65727369 Core.  ToolVersi
  0x00000030 6f6e3a20 302e312e 312d6465 76656c6f on: 0.1.1-develo
  0x00000040 700a2020 44617461 466f726d 61745665 p.  DataFormatVe
  0x00000050 7273696f 6e3a2030 2e312e30 0a202049 rsion: 0.1.0.  I
  0x00000060 6e707574 46696c65 4e616d65 3a206372 nputFileName: cr
  0x00000070 7970746f 2f637479 70652e63 0a536f75 ypto/ctype.c.Sou
  0x00000080 72636546 696c6573 3a0a2d20 44697265 rceFiles:.- Dire
  0x00000090 63746f72 793a202f 686f6d65 2f746e61 ctory: /home/tna
  0x000000a0 6d61652f 6f70656e 73736c2d 332e342e mae/openssl-3.4.
  0x000000b0 312f6372 7970746f 0a202046 696c6573 1/crypto.  Files
  0x000000c0 3a0a2020 2d204669 6c653a20 63747970 :.  - File: ctyp
      :
```

No build configuration changes are needed—it's all handled by the GCC spec file

Looks like metadata has been embedded in the ELF section ".esstra" of libssl.so

©2025 Sony Group Corporation

Use Case: OpenSSL

- Step2. Build OpenSSL using ESSTRA

```
esstra/util$ sudo apt install python3-yaml
esstra/util$ sudo make install

openssl-3.4.1$ esstra.py show libssl.so
#
# BinaryFileName: libssl.so
# BinaryPath: /home/tnamae/openssl-3.4.1/libssl.so.3
#
Headers:
  ToolName: ESSTRA Core
  ToolVersion: 0.1.1-develop
  DataFormatVersion: 0.1.0
  InputFileNames:
  - crypto/ctype.c
  - crypto/getenv.c
  - crypto/packet.c
  - crypto/quic_vlint.c
  - crypto/time.c
  - crypto/thread/arch/thread_none.c
  - crypto/thread/arch/thread_posix.c
  - crypto/thread/arch/thread_win.c
  - crypto/thread/arch.c
  - ssl/bio_ssl.c
  - ssl/d1_lib.c
  - ssl/d1_msg.c
      :
```

YAML-formatted metadata is embedded in the ELF section ".esstra"

Use "esstra.py show" command to dump it

Source files included in libssl.so are listed here

```
      :
  - ssl/statem/statem_dtls.c
  - ssl/statem/statem_lib.c
  - ssl/statem/statem_srvr.c
SourceFiles:
- Directory: /home/tnamae/openssl-3.4.1/crypto
  Files:
  - File: ctype.c
    SHA1: df5041387a4879f9ae985bda008039b96011fa06
  - File: getenv.c
    SHA1: 2681da5a2a6d344d6a095d330ad412ed93155c1c
  - File: packet.c
    SHA1: a3755e1c7050f151becbb74bf71764b20947c085
  - File: quic_vlint.c
    SHA1: acbb334a766b50d81b5592fc4191441e6558e288
  - File: time.c
    SHA1: d4b1475e39237b841a9ae19a04a0a97a036a597e
- Directory: /home/tnamae/openssl-3.4.1/crypto/thread
  Files:
  - File: arch.c
    SHA1: 2282faf9e30780e3584b935122407a099d4f34b7
- Directory: /home/tnamae/openssl-3.4.1/crypto/thread/arch
  Files:
  - File: thread_none.c
      :
  - File: ssl_lib.c
    SHA1: 1a7b0fb01a3c13a40e159c7aa704a323b667b3db
      :
```


©2025 Sony Group Corporation

Use Case: OpenSSL

- Step3. Analyze licenses with license scanner

An SPDX v2 tag-value file with “LicenseInfoInFile:” tags is required

https://raw.githubusercontent.com/sony/esstra/refs/heads/main/doc/case-study/output-examples/openssl/SPDX2TV_openssl-3.4.1.tar.gz.spdx

```
SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0

##-------------------------
## Document Information
##-------------------------

DocumentNamespace: http://1f9d9e80ed71/repo/SPDX2TV_openssl-3.4.1.tar.gz.spdx
DocumentName: /srv/fossology/repository/report
SPDXID: SPDXRef-DOCUMENT

    :

##File

FileName: openssl-3.4.1.tar.gz/openssl-3.4.1.tar/openssl-3.4.1/ssl/ssl_lib.c
SPDXID: SPDXRef-item26070
FileChecksum: SHA1: 1a7b0fb01a3c13a40e159c7aa704a323b667b3db
FileChecksum: SHA256: 4ebc88bd5ef3522d479d57bab25a15df65a9b2e4d13fd5fc4e0da18450b4a3eb
FileChecksum: MD5: 0892c3d84de8e5dfadb5a7ba218d63c2
LicenseConcluded: NOASSERTION
LicenseInfoInFile: OpenSSL
LicenseInfoInFile: Apache-2.0
FileCopyrightText: <text> Copyright 1995-2024 The OpenSSL Project Authors. All Rights Reserved.
FileCopyrightText: <text> Copyright 1995-2024 The OpenSSL Project Authors. All Rights Reserved.
Copyright (c) 2002, Oracle and/or its affiliates. All rights reserved
Copyright 2005 Nokia. All rights reserved. </text>
```

 ©2025 Sony Group Corporation

Use Case: OpenSSL

- Step4. Add license information to metadata

```
openssl-3.4.1$ esstra.py update libssl.so -i
~/.../SPDX2TV_openssl-3.4.1.tar.gz.spdx

openssl-3.4.1$ esstra.py show libssl.so
#
# BinaryFileName: libssl.so
# BinaryPath: /home/tnamae/openssl-3.4.1/libssl.so.3
#
Headers:
  ToolName: ESSTRA Core
  ToolVersion: 0.1.1-develop
  DataFormatVersion: 0.1.0
  InputFileNames:
  - crypto/ctype.c
  - crypto/getenv.c
  - crypto/packet.c
  - crypto/quic_vlint.c
  - crypto/time.c
  - crypto/thread/arch/thread_none.c
  - crypto/thread/arch/thread_posix.c
  - crypto/thread/arch/thread_win.c
  - crypto/thread/arch.c
  - ssl/bio_ssl.c
  - ssl/d1_lib.c
  - ssl/d1_msg.c
      :
```

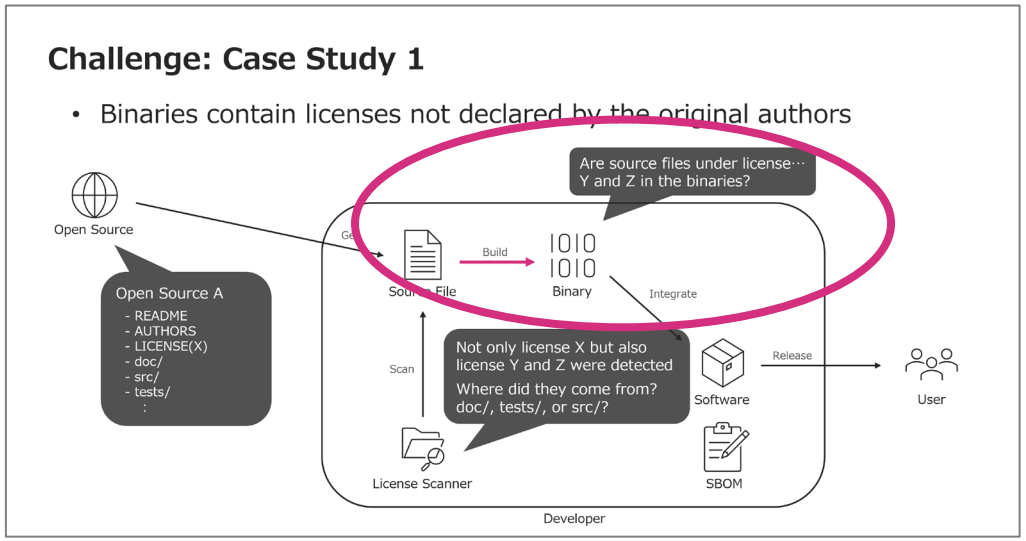
Source files included in libssl.so are listed here, each with license information

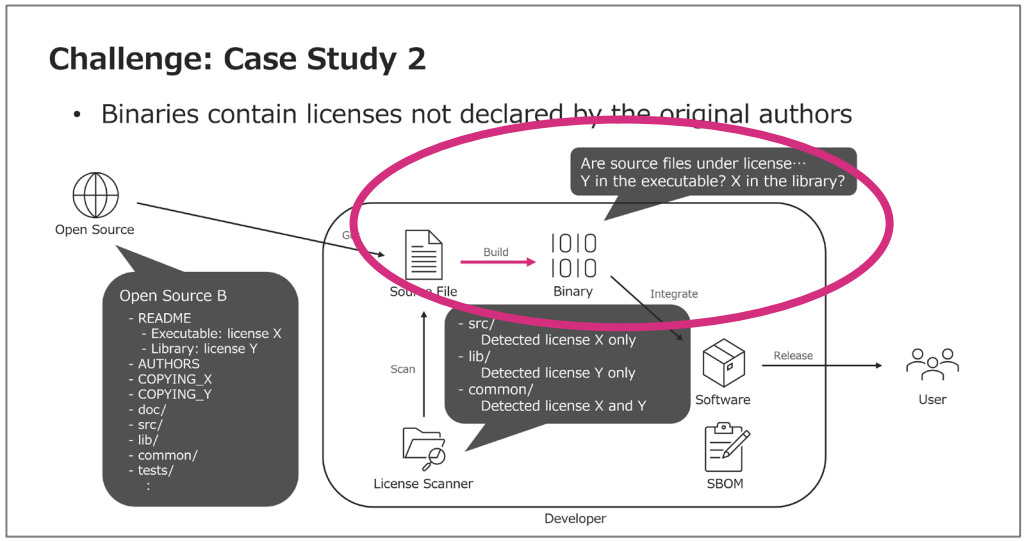
```
      :
  - ssl/statem/statem_srvr.c
SourceFiles:
- Directory: /home/tnamae/openssl-3.4.1/crypto
  Files:
  - File: ctype.c
    SHA1: df5041387a4879f9ae985bda008039b96011fa06
    LicenseInfo:
    - Apache-2.0
    - OpenSSL
  - File: getenv.c
    SHA1: 2681da5a2a6d344d6a095d330ad412ed93155c1c
    LicenseInfo:
    - Apache-2.0
    - OpenSSL
  - File: packet.c
    SHA1: a3755e1c7050f151becbb74bf71764b20947c085
    LicenseInfo:
    - OpenSSL
    - Apache-2.0
  - File: quic_vlint.c
      :
  - File: ssl_lib.c
    SHA1: 1a7b0fb01a3c13a40e159c7aa704a323b667b3db
    LicenseInfo:
    - OpenSSL
    - Apache-2.0
      :
```


©2025 Sony Group Corporation

Measured Impact

- Efficient and effective investigations made possible by ESSTRA
  - Reference Data
    - Target: 187 Debian binary packages for Linux-based system software
    - Total number of source files in…
      - The Debian source packages: 352,986 (A)
      - The binaries for the software: 31,159 (B), Detected by ESSTRA
    - Utilization rate of source files: (B / A) × 100 = 8.8%

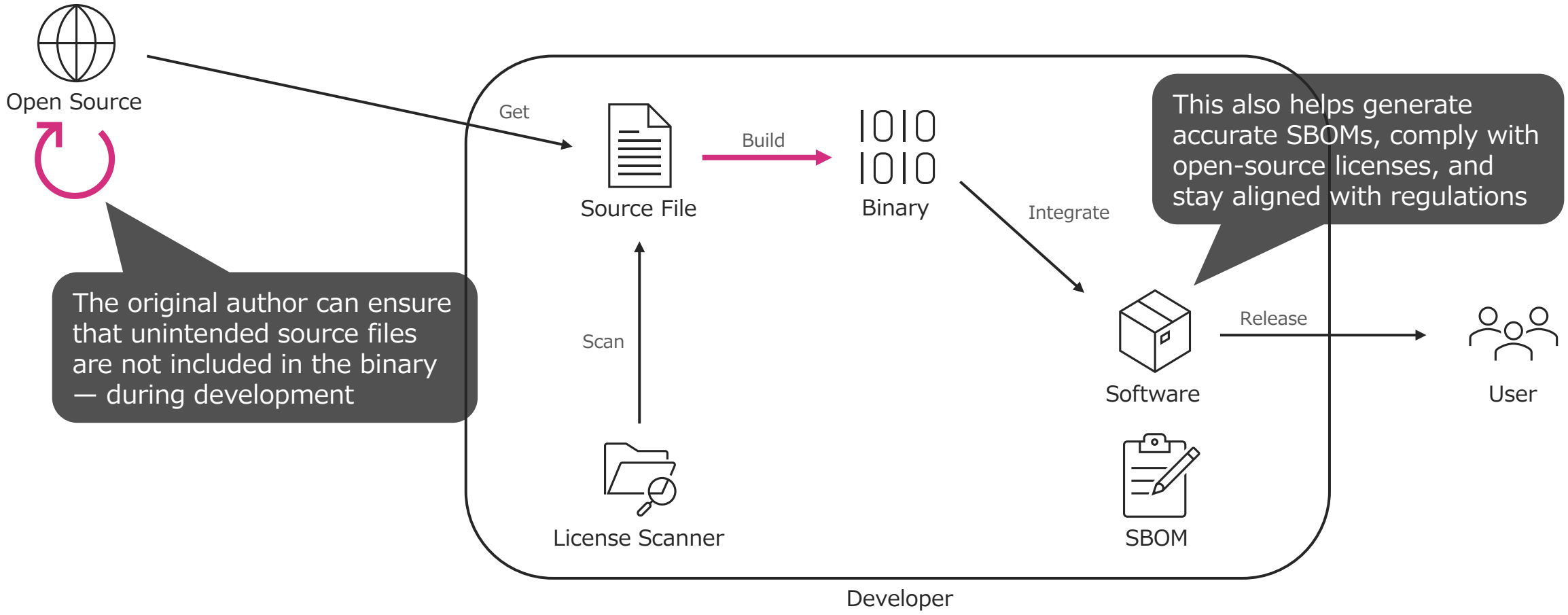




 ©2025 Sony Group Corporation

Why We Open-Sourced

- No matter where you are in the software supply chain, ESSTRA is here to support you — give it a try!

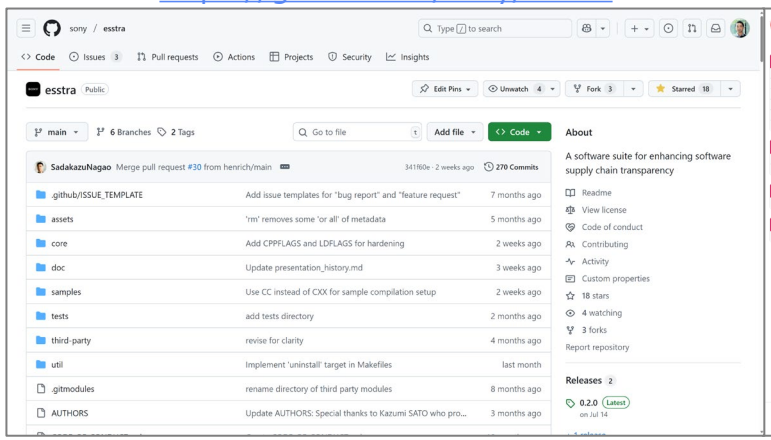


©2025 Sony Group Corporation

Where to Get

GitHub

https://github.com/sony/esstra

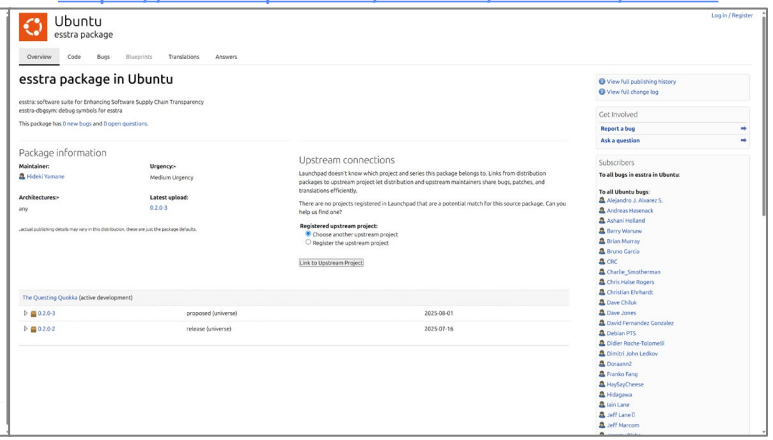


Debian

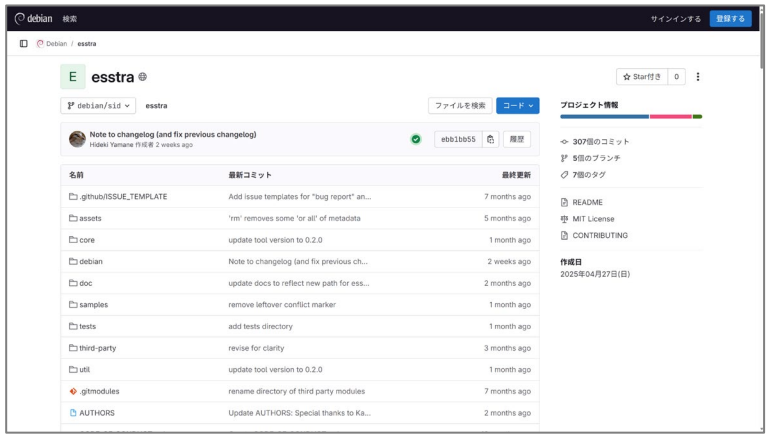
https://tracker.debian.org/pkg/esstra

Ubuntu

https://launchpad.net/ubuntu/+source/esstra



https://salsa.debian.org/debian/esstra



Debian packaged and maintained by

Hideki Yamane

SIOS Technology, Inc.
Debian Developer
https://contributors.debian.org/contributor/henrich/
https://qa.debian.org/developer.php?email=henrich%40debian.org

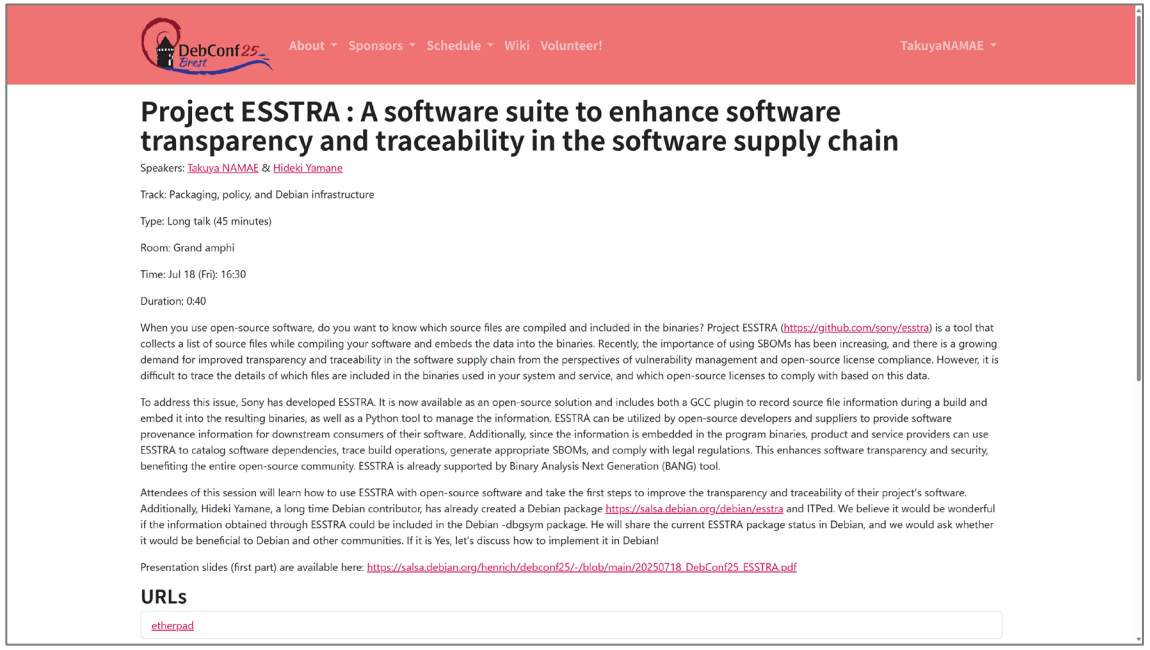
 ©2025 Sony Group Corporation

Provenance Info Proposal for Debug Packages at DebConf25

- Joint presentation with Hideki Yamane
- Received valuable feedback from Debian Developers

https://debconf25.debconf.org/talks/118-project-esstra-a-software-suite-to-enhance-software-transparency-and-traceability-in-the-software-supply-chain/

https://salsa.debian.org/debconf-team/public/share/debconf25/-/tree/main/slides/118-project-esstra-a-software-suite-to-enhance-software-transparency-and-traceability-in-the-software-supply-chain



DebConf25 Brest — About · Sponsors · Schedule · Wiki · Volunteer! · TakuyaNAMAE

Project ESSTRA : A software suite to enhance software transparency and traceability in the software supply chain

Speakers: Takuya NAMAE & Hideki Yamane

Track: Packaging, policy, and Debian infrastructure

Type: Long talk (45 minutes)

Room: Grand amphi

Time: Jul 18 (Fri): 16:30

Duration: 0:40

When you use open-source software, do you want to know which source files are compiled and included in the binaries? Project ESSTRA (https://github.com/sony/esstra) is a tool that collects a list of source files while compiling your software and embeds the data into the binaries. Recently, the importance of using SBOMs has been increasing, and there is a growing demand for improved transparency and traceability in the software supply chain from the perspectives of vulnerability management and open-source license compliance. However, it is difficult to trace the details of which files are included in the binaries used in your system and service, and which open-source licenses to comply with based on this data.

To address this issue, Sony has developed ESSTRA. It is now available as an open-source solution and includes both a GCC plugin to record source file information during a build and embed it into the resulting binaries, as well as a Python tool to manage the information. ESSTRA can be utilized by open-source developers and suppliers to provide software provenance information for downstream consumers of their software. Additionally, since the information is embedded in the program binaries, product and service providers can use ESSTRA to catalog software dependencies, trace build operations, generate appropriate SBOMs, and comply with legal regulations. This enhances software transparency and security, benefiting the entire open-source community. ESSTRA is already supported by Binary Analysis Next Generation (BANG) tool.

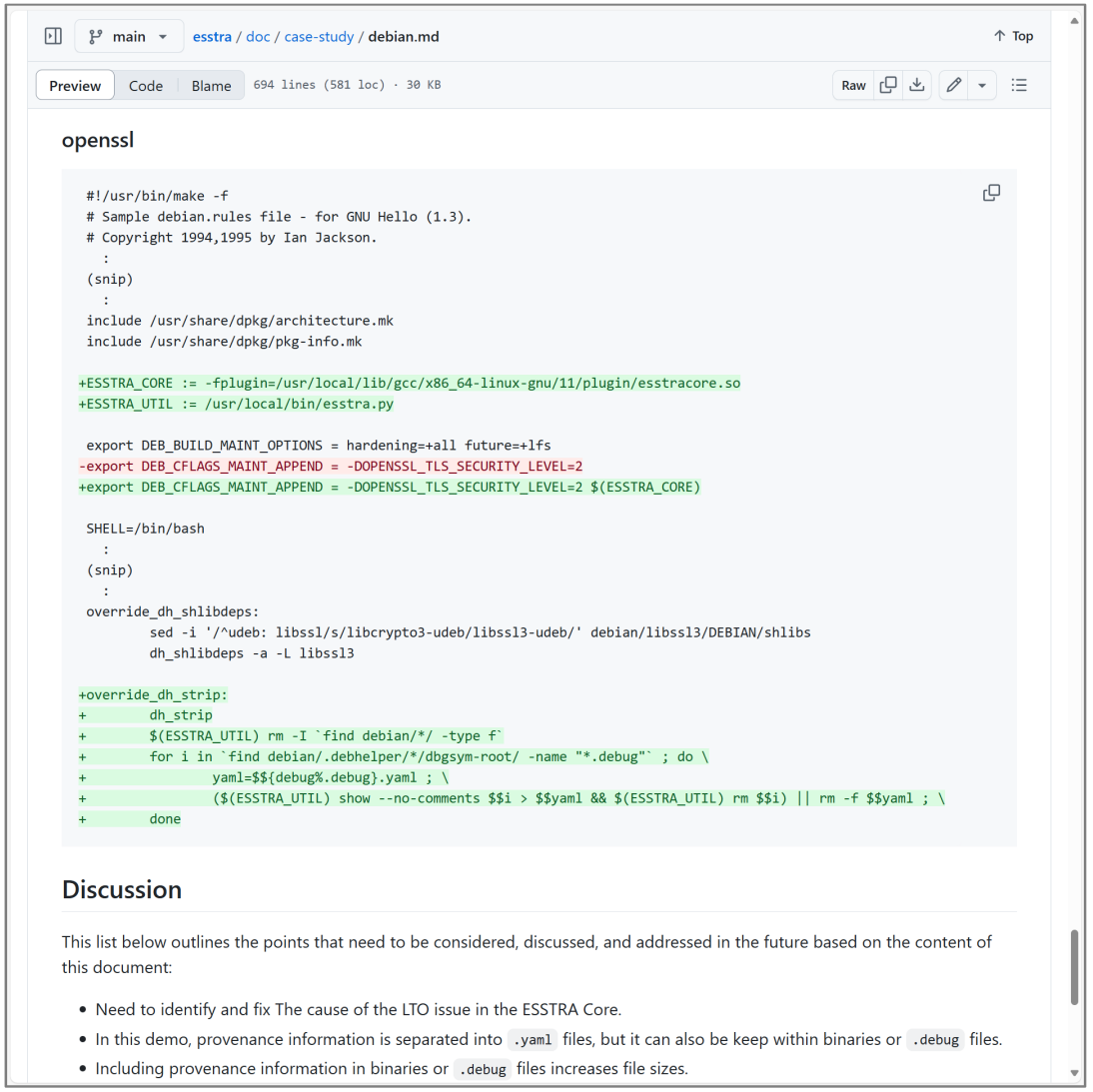
Attendees of this session will learn how to use ESSTRA with open-source software and take the first steps to improve the transparency and traceability of their project's software. Additionally, Hideki Yamane, a long time Debian contributor, has already created a Debian package https://salsa.debian.org/debian/esstra and ITPed. We believe it would be wonderful if the information obtained through ESSTRA could be included in the Debian -dbgsym package. He will share the current ESSTRA package status in Debian, and we would ask whether it would be beneficial to Debian and other communities. If it is Yes, let's discuss how to implement it in Debian!

Presentation slides (first part) are available here: https://salsa.debian.org/henrich/debconf25/-/blob/main/20250718_DebConf25_ESSTRA.pdf

URLs

etherpad

https://github.com/sony/esstra/blob/main/doc/case-study/debian.md



main · esstra / doc / case-study / debian.md · Top

Preview | Code | Blame · 694 lines (581 loc) · 30 KB · Raw

openssl

```
#!/usr/bin/make -f
# Sample debian.rules file - for GNU Hello (1.3).
# Copyright 1994,1995 by Ian Jackson.
  :
(snip)
  :
include /usr/share/dpkg/architecture.mk
include /usr/share/dpkg/pkg-info.mk

+ESSTRA_CORE := -fplugin=/usr/local/lib/gcc/x86_64-linux-gnu/11/plugin/esstracore.so
+ESSTRA_UTIL := /usr/local/bin/esstra.py

 export DEB_BUILD_MAINT_OPTIONS = hardening=+all future=+lfs
-export DEB_CFLAGS_MAINT_APPEND = -DOPENSSL_TLS_SECURITY_LEVEL=2
+export DEB_CFLAGS_MAINT_APPEND = -DOPENSSL_TLS_SECURITY_LEVEL=2 $(ESSTRA_CORE)

 SHELL=/bin/bash
  :
(snip)
  :
override_dh_shlibdeps:
        sed -i '/^udeb: libssl/s/libcrypto3-udeb/libssl3-udeb/' debian/libssl3/DEBIAN/shlibs
        dh_shlibdeps -a -L libssl3

+override_dh_strip:
+       dh_strip
+       $(ESSTRA_UTIL) rm -I `find debian/*/ -type f`
+       for i in `find debian/.debhelper/*/dbgsym-root/ -name "*.debug"` ; do \
+               yaml=$${debug%.debug}.yaml ; \
+               ($(ESSTRA_UTIL) show --no-comments $$i > $$yaml && $(ESSTRA_UTIL) rm $$i) || rm -f $$yaml ; \
+       done
```

Discussion

This list below outlines the points that need to be considered, discussed, and addressed in the future based on the content of this document:

- Need to identify and fix The cause of the LTO issue in the ESSTRA Core.
- In this demo, provenance information is separated into `.yaml` files, but it can also be keep within binaries or `.debug` files.
- Including provenance information in binaries or `.debug` files increases file sizes.

©2025 Sony Group Corporation

Future Work

- Ongoing work by our team
  - Reproducible build support
    - Introduce path normalization to avoid embedding absolute paths in ESSTRA metadata
  - Linker plugin
    - Optimize metadata during linking by removing duplicates across object files
  - Tests
    - Topic branch under preparation; implementation work in progress
- Areas where community support is welcome
  - Support for other compilers
    - Extend ESSTRA compatibility to LLVM/Clang, Rust, and Go toolchains, etc.
  - Vulnerability metadata support
    - Explore how vulnerability data could be embedded and linked in .esstra metadata
  - LTO metadata issue
    - Investigate incomplete metadata in final binaries when compiled with Link Time Optimization

©2025 Sony Group Corporation

Special Thanks

- Armijn Hemel
  - Provided technical advice soon after releasing ESSTRA as open-source last year
  - Extended his Binary Analysis Next Generation (BANG) tool to support ESSTRA
- Hideki Yamane
  - Showed interest in ESSTRA and suggested that we present it at DebConf25
  - Packaged ESSTRA for Debian and currently maintains the package
- Sony Group OSPO
  - Always offered dedicated support
  - Their long-term engagement in the open-source community made ESSTRA's open-source work possible
- Team Members
  - My colleagues in Japan and India, mentor, and managers who supported the project throughout

©2025 Sony Group Corporation

Q&A

Thank you for listening

I’m happy to take your questions and feedback

 ©2025 Sony Group Corporation

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.