Free and Open Source Software

Covering OSS Compliance with Software Tools

Dr. Catharina Maracke
Dr. Michael C. Jaeger
Software Compliance Academy

2018-06-12

SOFTWARE COMPLIANCE
ACADEMY

Overview

Main kinds of tools in the area of license compliance include *(but are not limited to)*:

- Source code scanning
- License scanning
- Binary scanning
- Dev Ops integration
- Component management

1. License Scanner

License Scanner: Introduction

Purpose:

Identifies licenses and license relevant statements

Other Identifications:

Copyright statements, author statements, acknowledgements

Also of interest:

Export control statements, more static code analysis

License Scanner: Solved Problem

Problem: Identify licensing in Open Source Software packages

Licensing in Open Source Software

- Licensing of OSS can be heterogeneous, different licensing applies to parts of OSS
- Licensing statements are not uniform
- Many licenses exist, number growing

-> Tool based licensing identification required for complicated licensing situations

License Scanner: Technical

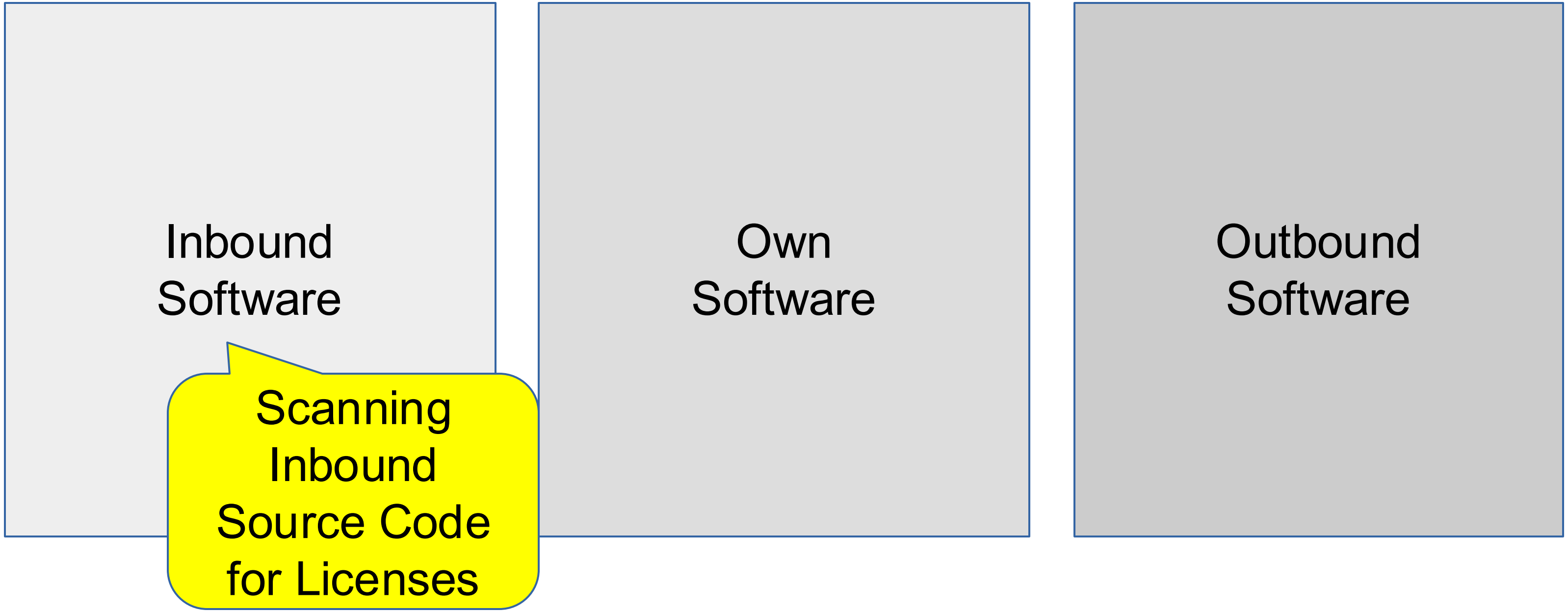
Mode of operation: Tool searches in content
for license relevant keywords, phrases, license texts

- Searching in every file of software uploaded:
  requires source code distribution
- Different approaches can be applied:
  regular expressions, text comparison, phrase collection
- Requires database of license texts, licensing statements
- Comparison with existing license texts enables exact identification
- Licensing information can summarized for open source packages

License Scanner: More remarks

- License scanning does not require huge database
- However, updates are necessary as licensing statements evolve and new licenses are still created
- Identified licensing information of a software package can be exchanged using SPDX files
- Approach makes sense for OSS licenses, commercial licensing is even more heterogeneous
- License identification precision depends on available licensing information and may require expert knowledge for analysis

License Scanner Main Usage



Inbound Software

Own Software

Outbound Software

Scanning Inbound Source Code for Licenses

2. Binary Scanner

Binary Scanner: Introduction

Purpose:

- Identifies used software packages in software binaries

Other identifications:

- Can also determine the versions of software packages

Also of interest:

- Identifying used software packages for creating the binary also enables identification of vulnerabilities

Binary Scanner: Solved Problem

Problem: A binary is comprised of different software packages, but if not declared, not obvious to determine

- Applies in compiled programming languages: programming language code is translated (=compiled) into machine executable code (machine = processor)
- Script languages (e.g. JavaScript) are not compiled
- Binaries are usually not readable, understanding contents difficult
- However, identification of contents can be inevitable for understanding required license compliance tasks

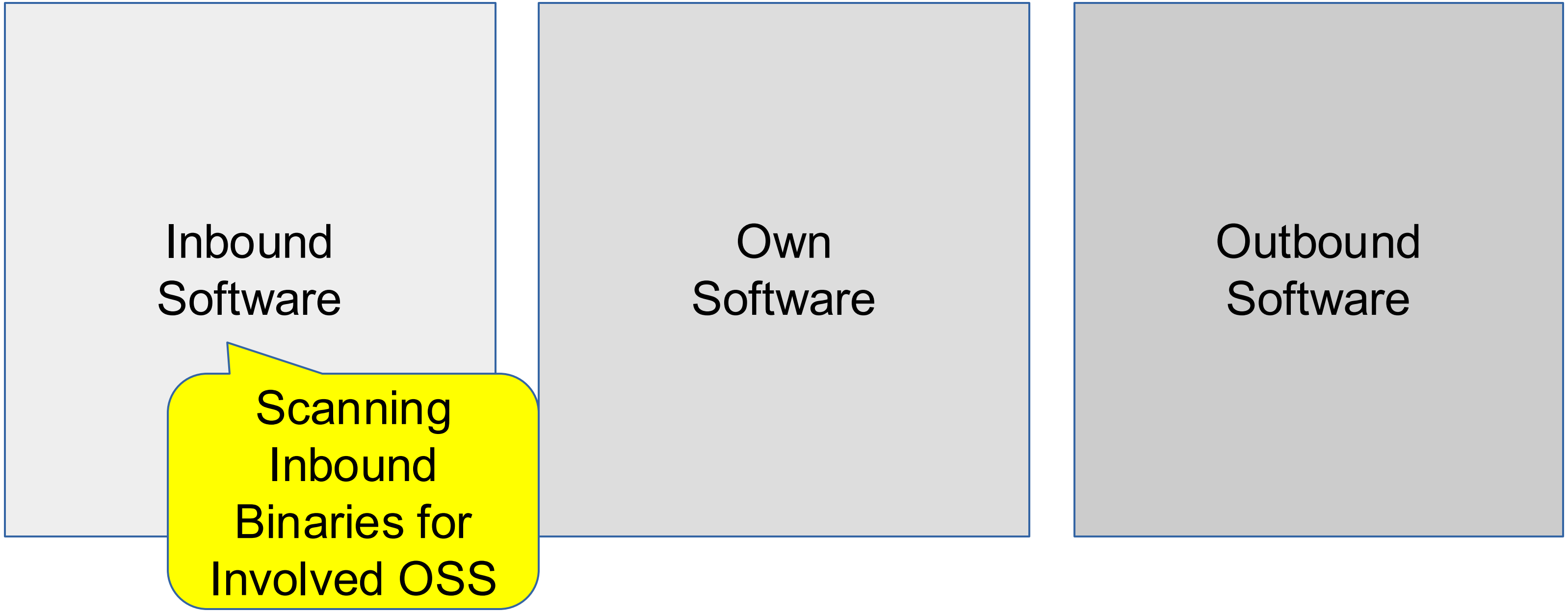
Binary Scanner: Technical

- Compiled machine language
  can contain characteristic elements
- For example used string variables (=text)
  or other content compiled into the binary
- Simpler method: capturing file names,
  or for run-time code (e.g. Java): method and field names
- Requires database of mapping
  from source code to resulting artifacts in binary

Binary Scanner: Remarks

- Binary scanning is a heuristic,
  secure mapping not supported for every possible binary
- Topic connected with reproducible builds
  (then, binaries can be compared more efficiently)
- Database requires updates because,
  because new software is published every day
    - (similar with source code scanning)

Binary Scanner Main Usage



Inbound Software

Own Software

Outbound Software

Scanning Inbound Binaries for Involved OSS

3. Source Code Scanner

Source Code Scanner: Introduction

Purpose:

Can identify published origin of source code and other files

Other Identifications:

Icons, images, style descriptions, XML schemes, documentation

Also of interest:

Programming examples, from blogs and best practise Websites

Source Code Scanner: Solved Problem

Problem: how to understand that source code or other files have been taken from elsewhere, not self-created, and not declared

If "own" software is not entirely own software and not understood:

- Missing rights for business case in "own" software
- But distribution requires distribution rights are available
- Identification of origin is first step to understand available rights

Source Code Scanner: Technical

Mode of operation: upload source code or just files or fingerprints of it, get origin in case it is captured by database

- File contents are compared
  with contents from (huge) database of published contents
- Fingerprinting of file contents (“hashing”)
  allow for accelerated search and storage in database
- Not only coverage of entire files, but fragments of it
- Database requires updates: every day new published OSS
- Content is large (e.g. the entire GitHub)

Source Code Scanner: More Remarks

Once origin of source is identified,
more metadata can be made available:

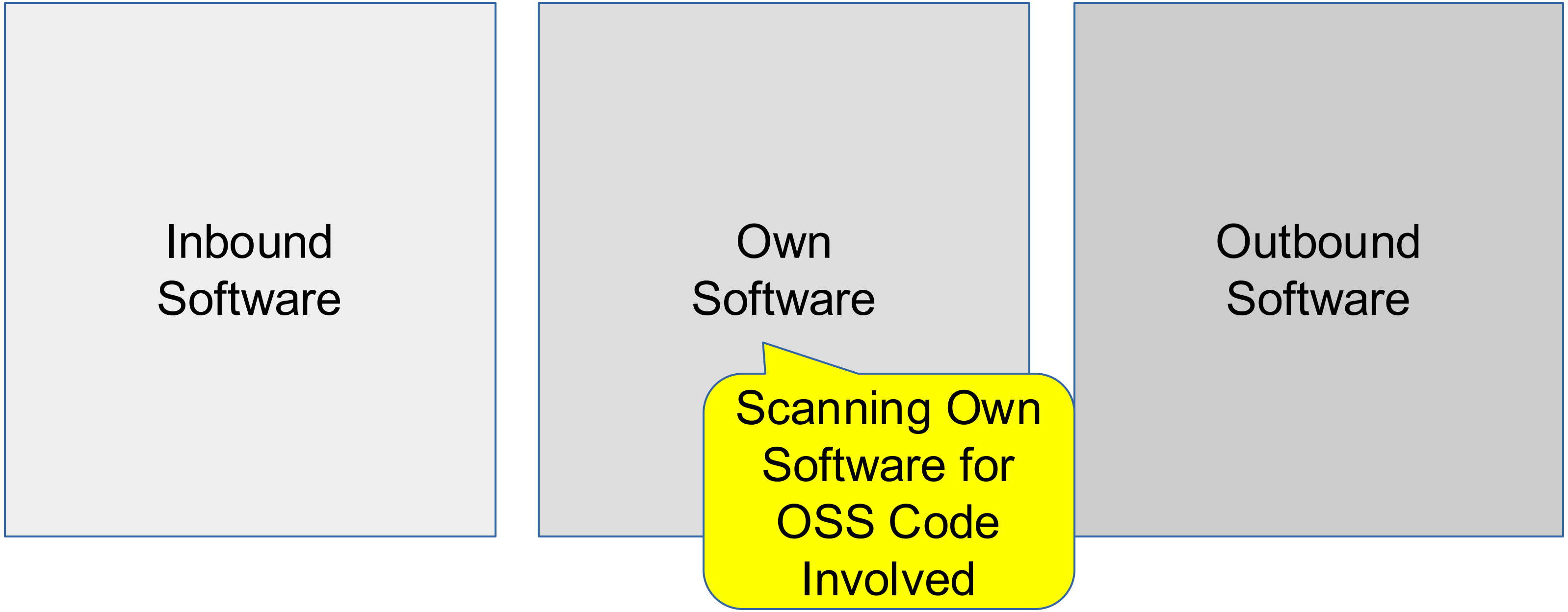
- Licensing
- Vulnerabilities

Potential for integration:

- Development toolchain
- Reporting, BOM

Matched content may require expert knowledge
to determine relevance

Source Scanner Main Usage



Inbound Software

Own Software

Outbound Software

Scanning Own Software for OSS Code Involved

4. Dev Ops Integration

Dev Ops Integrations: Introduction

Purpose:

- Uses the information from building the software to determine OSS used

Other identifications:

- Can be combined with source code scanning, license scanning, binary scanning

Also of interest:

- Resulting identification of elements during building the software enables the creation of a bill of material (BOM)

Dev Ops Integrations: Solved problem

Problem: for larger software projects
a tool based approach is inevitable to understand involved OSS

- Modern software building environments have defined dependencies
- During compilation, dependencies can be captured
  to understand used dependencies
- License compliance integrated
  into the Dev Ops tooling implements automation
- Reporting as part of Dev Ops tooling reduces manual efforts
- Enables short release cycles in an agile environment

Dev Ops Integrations: Technical

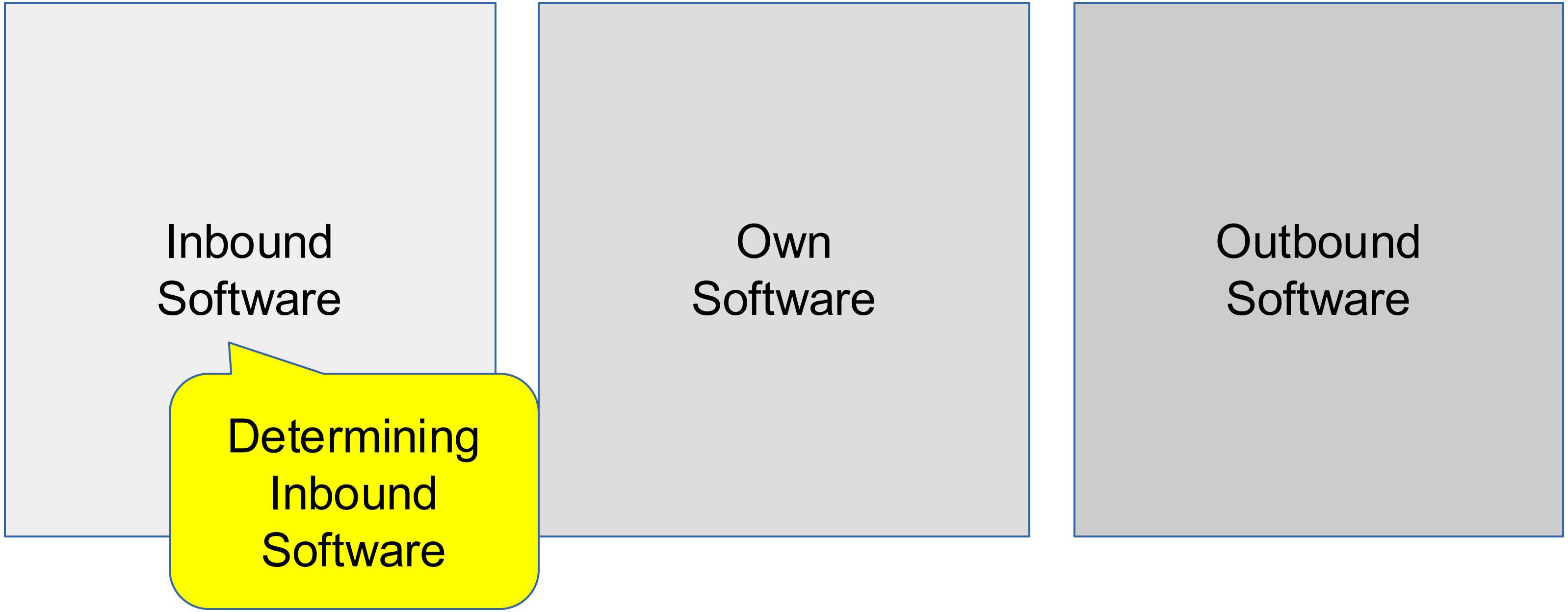
Integration into Dev Ops tooling requires customization

- Building software depends on used technology
  as well as individually setup tooling
- Additional efforts, if software is comprised of different technologies
- Today, building environments sometimes contain already metadata about licensing of involved OSS software
- Identified software elements may require additional checks to determine actual licensing information
  (in case of heterogeneous licensing)

Dev Ops Integrations: Remarks

- Today, a custom task, nothing to "download and double-click"
- Tooling approach allows for differential approach: once setup and checked, only new dependencies require additional coverage

Dev Ops Integration Main Usage



Inbound Software

Own Software

Outbound Software

Determining Inbound Software

5. Component Catalogue

Component Catalogue: Introduction

Purpose:

- Collect information about used software components and their use in products or projects is centrally collected and can be reused

Other purposes:

- A component catalogue captures also the used components in a product or project, maintains a so-named BOM

Also interesting:

- Enables also vulnerability management or reuse of export classifications

Component Catalogue: Solved Problem

Problem: Once analysed component w.r.t. license compliance shall not require repeated analyses, but reuse of information shall be possible

Component catalogue:

- Maps component usage in products or projects
- Makes sense if an organisation has actually multiple products
- Shows organisation the important software components
- Allows for a comprehensive overview about involved licensing per product

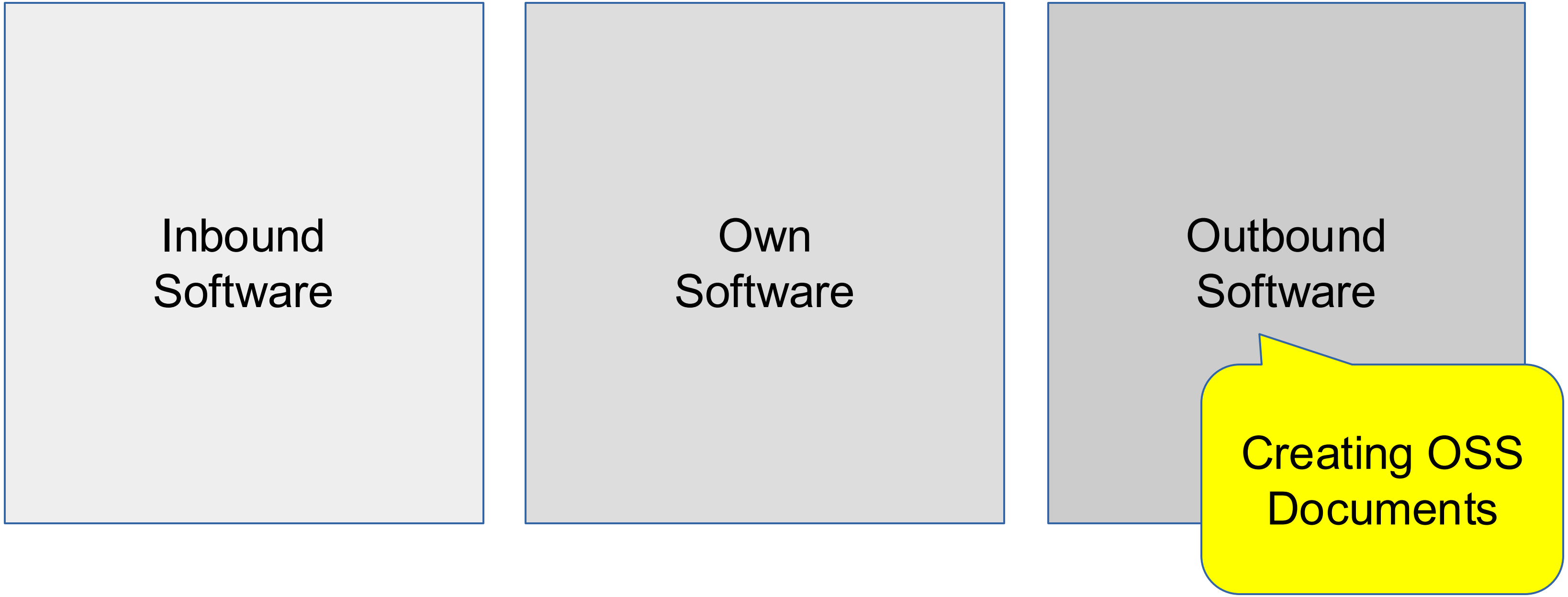
Component Catalogue: Technical

- A component catalogue can be viewed as a portal
- Database holding the catalogue information
- Another use case is archiving OSS distributions / source code
- Storing also multiple other files,
  for example license analysis reports, SPDX files
- Provides reporting output, for example OSS product documentation
- Component catalogue can be implemented as Web portal, thus accessible from various client computers in organisation

Component Catalogue: Remarks

- Component catalogue can be integrated with other license compliance tooling: scanners can directly feed the analyses
- Also integration in Dev Ops tooling is useful to automatically create BOM of products
- Component catalogues can also serve uses cases for vulnerability management
- Another related topic is license management and license metadata

Component Catalogue Usage



Inbound Software

Own Software

Outbound Software

Creating OSS Documents

Questions?

office@scompliance.com