Open Source in Industry:
Scanning, compliance and OSADL services

Legal Heidelberg OSADL Talks, April 28, 2020, Online Session 3

What is software license scanning and when is it needed?
License compliance as integral part of company compliance
OSADL License Compliance Audit (LCA)
Example of the OSADL legal FAQ collection
Example of a legal assessment

Some information on today's sessions

- Please provide feedback on Legal HOT using the online form
  - Use the quick link **osadl.org/FB** (FeedBack), same as osadl.org/?id=3323
- You may ask questions during the session to be answered online, if possible
  - The quick link URL is **osadl.org/AQ** (AskQuestion), same as osadl.org/?id=3321
- You may join an online discussion on all topics of today at 4 pm
  - The quick link URL is **osadl.org/OD** (OnlineDiscussion), same as jitsi.osadl.org
  - Meeting name **OSADLLegalHOT**
  - Username and password will be displayed here after the last presentation

(We will show this slide again at the end of this session)

What is „Scanning“?

The term „Scanning“ in the context of license compliance may refer to two completely different issues:

1. Extract typical lines of text from program source and other files possibly protected by copyright law. The main purpose is to collect obvious notices in plain text
   ⇒ Informational Scanning.
2. Discover non-obvious, hidden or even obfuscated software snippets that were incorporated from third parties and may not licensed correctly. For this purpose, certain criteria from suspicious software (“finger prints”) are matched against a usually large data base of the same criteria of known software components
   ⇒ Forensic Scanning.

Informational *vs* forensic scanning

Scanning	Effort	Duration	Needed by everybody?	Examples
Informational scanning	Relatively small	Minutes/hours	Probably yes	Grep, Ninka, Fossology, Scancode
Forensic scanning	Very big	Days/weeks	No, not necessarily	Black Duck, Palamida/Flexera BAT

Scanning and beyond ...

- Source code administration
  - Comprehensive table of licenses in use
  - History of licenses, documentation of license changes
  - Hints to obligations of detected licenses
  - Evaluation of license compatibility
- Batch-Processing
  - License scanning integrated into tool chain and build processes
  - Alerts (e.g. via email), if critical change detected
  - Documentation as a proof of implemented license compliance

BTW: Why do we need scanning?

GPL-2.0 Section 1:

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately **publish on each copy an appropriate copyright notice** and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

Fulfill obligation „Publish copyright notice(s)“

- Challenges
  - **Formal presentation not specified:**

    `Copyright © 2019 Employer LLC, author John Doe`

    could have been written as

    `Owned by Employer LLC, written by John Doe`
  - **Possible large number of copyright holders and authors**

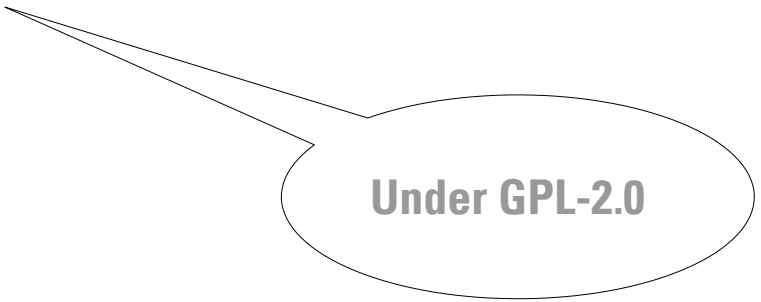
    At the time when some licenses were created, there were no large communities of distributed software development with more than thousend developers.

Formal presentation of copyright not specified

Other sources of information may need to be consulted such as the file „MAINTAINERS“ of the Linux kernel:

Formal presentation of copyright not specified

Other sources of information may need to be consulted such as the file „MAINTAINERS“ of the Linux kernel:



Formal presentation of copyright not specified

Other sources of information may need to be consulted such as the file „MAINTAINERS“ of the Linux kernel:

Step #1: Building a list of authors:

```
# grep "^M:" MAINTAINERS | sed 's/^M:[\x09 ]*//' | cut "-d<"
-f1 | tr -d '"' | grep -v @ | sort | uniq >maintainers
```

Formal presentation of copyright not specified

Other sources of information may need to be consulted such as the file „MAINTAINERS“ of the Linux kernel:

Step #1: Building a list of authors:

```
# grep "^M:" MAINTAINERS | sed 's/^M:[\x09 ]*//' | cut "-d<"
-f1 | tr -d '"' | grep -v @ | sort | uniq >maintainers
```

Step #2: Searching for authors and formal descriptors:

```
# grep -ir -f maintainers -e "copyright.*[12][90][0-9][0-9]"
-e "(c).*[12][90][0-9][0-9]" . | grep -v -e _AUTHOR -e
^./Documentation -e ^./tools -e ^./samples -e ^./patch -e
^./.git -e ^./.pc -e ./MAINTAINERS: >copyright-notices
```

A storage medium used to look like that

8-inch floppy disk:

- 1,6 MByte unformatted data
- 1,2 MByte formatted data
- 500 Kbit/s data transfer
- Few authors
- More than 400 cm² area for copyright notices

A storage medium may look like that today

For example USB storage:

- Up to 1 TByte capacity
- Up to 100 MByte/s data transfer
- Possibly more than thousand authors
- Very little area for copyright notices

Large number of copyright holders and authors

- The file `copyright-notices` (Linuxkernel 5.2.21-rt13):

```
# wc -l copyright-notices
65141 copyright-notices
```

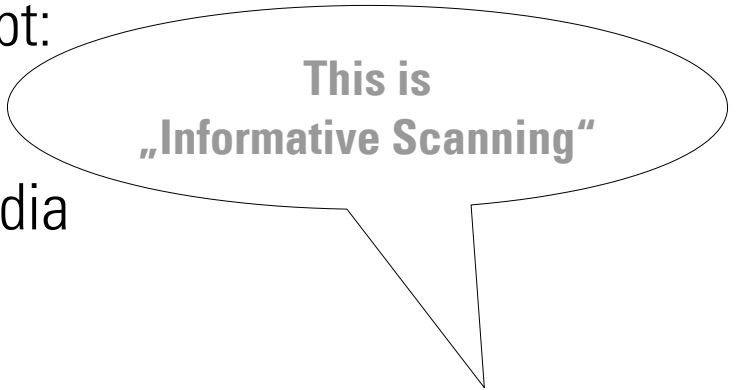
- As normal text document printed in 12 pt:
  1595 pages
- Can only be forwarded in electronic media
- Document cannot be created manually
- Instead of a self-made script (as in our example) established tools must be used.

Large number of copyright holders and authors

- The file `copyright-notices` (Linuxkernel 5.2.21-rt13):

```
# wc -l copyright-notices
65141 copyright-notices
```

- As normal text document printed in 12 pt:
  1595 pages
- Can only be forwarded in electronic media
- Document cannot be created manually
- Instead of a self-made script (as in our example) established tools must be used.



The file copyright-notices, page #1

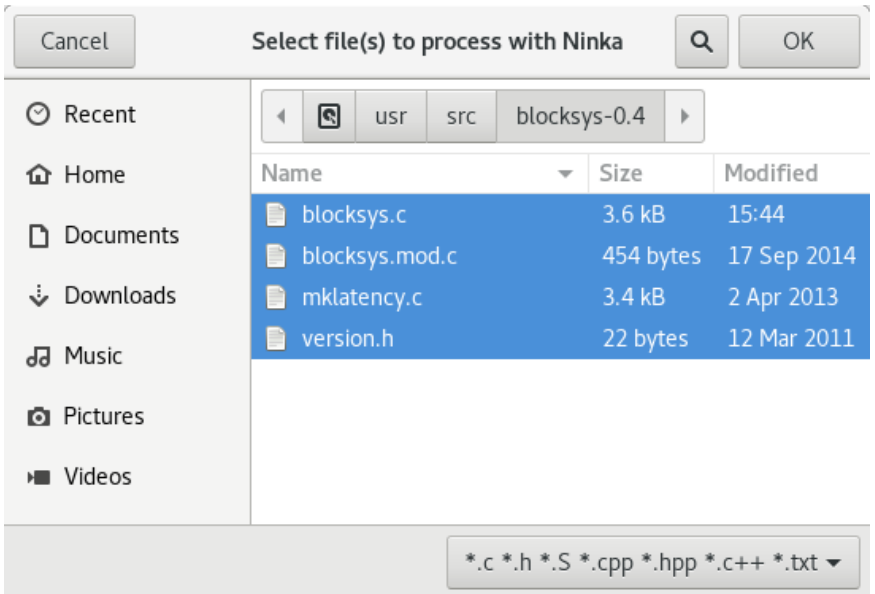
```
./kernel/softirq.c: * Copyright (C) 1992 Linus Torvalds
./kernel/futex.c: * (C) Rusty Russell, IBM 2002
./kernel/futex.c: * (C) Copyright 2003 Red Hat Inc, All Rights Reserved
./kernel/futex.c: * (C) Copyright 2003, 2004 Jamie Lokier
./kernel/futex.c: * (C) Copyright 2006 Red Hat Inc, All Rights Reserved
./kernel/futex.c: * Thanks to Thomas Gleixner for suggestions, analysis and fixes.
./kernel/futex.c: * PI-futex support started by Ingo Molnar and Thomas Gleixner
./kernel/futex.c: * Copyright (C) 2006 Red Hat, Inc., Ingo Molnar <mingo@redhat.com>
./kernel/futex.c: * Copyright (C) 2006 Timesys Corp., Thomas Gleixner <tglx@timesys.com>
./kernel/futex.c: * Copyright (C) 2007 Eric Dumazet <dada1@cosmosbay.com>
./kernel/futex.c: * Requeue-PI support by Darren Hart <dvhltc@us.ibm.com>
./kernel/futex.c: * Copyright (C) IBM Corporation, 2009
./kernel/futex.c: * Thanks to Thomas Gleixner for conceptual design and careful reviews.
./kernel/irq/resend.c: * Copyright (C) 1992, 1998-2006 Linus Torvalds, Ingo Molnar
./kernel/irq/resend.c: * Copyright (C) 2005-2006, Thomas Gleixner
./kernel/irq/affinity.c: * Copyright (C) 2016 Thomas Gleixner.
./kernel/irq/affinity.c: * Copyright (C) 2016-2017 Christoph Hellwig.
./kernel/irq/autoprobe.c: * Copyright (C) 1992, 1998-2004 Linus Torvalds, Ingo Molnar
./kernel/irq/debugfs.c:// Copyright 2017 Thomas Gleixner <tglx@linutronix.de>
./kernel/irq/dummychip.c: * Copyright (C) 1992, 1998-2006 Linus Torvalds, Ingo Molnar
./kernel/irq/dummychip.c: * Copyright (C) 2005-2006, Thomas Gleixner, Russell King
```

The file `copyright-notices`, page #1595

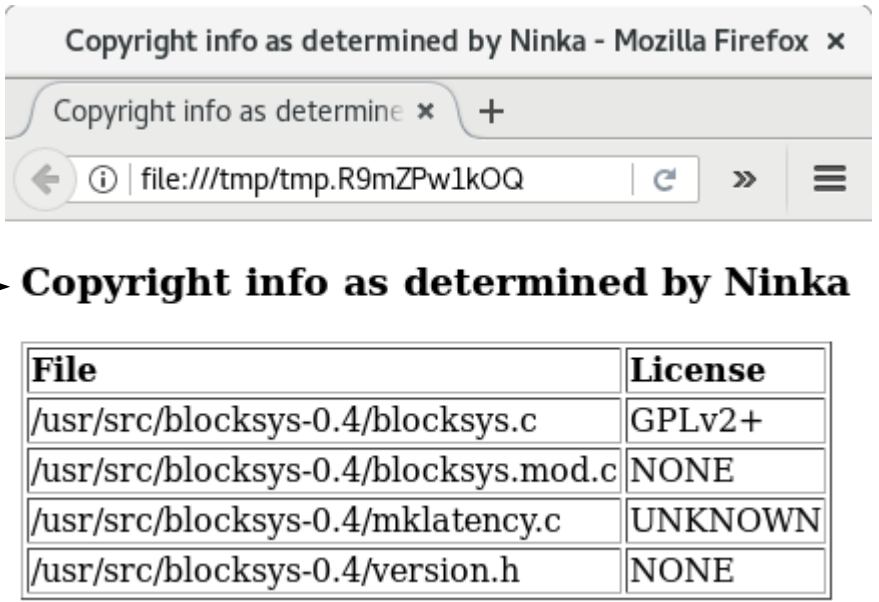
```
./fs/affs/dir.c: * (c) 1996 Hans-Joachim Widmaier - Rewritten
./fs/affs/dir.c: * (C) 1993 Ray Burr - Modified for Amiga FFS filesystem.
./fs/affs/dir.c: * (C) 1992 Eric Youngdale Modified for ISO 9660 filesystem.
./fs/affs/dir.c: * (C) 1991 Linus Torvalds - minix filesystem
./fs/affs/inode.c: * (c) 1996 Hans-Joachim Widmaier - Rewritten
./fs/affs/inode.c: * (C) 1993 Ray Burr - Modified for Amiga FFS filesystem.
./fs/affs/inode.c: * (C) 1992 Eric Youngdale Modified for ISO9660 filesystem.
./fs/affs/inode.c: * (C) 1991 Linus Torvalds - minix filesystem
./fs/affs/file.c: * (c) 1996 Hans-Joachim Widmaier - Rewritten
./fs/affs/file.c: * (C) 1993 Ray Burr - Modified for Amiga FFS filesystem.
./fs/affs/file.c: * (C) 1992 Eric Youngdale Modified for ISO 9660 filesystem.
./fs/affs/file.c: * (C) 1991 Linus Torvalds - minix filesystem
./fs/affs/super.c: * (c) 1996 Hans-Joachim Widmaier - Rewritten
./fs/affs/super.c: * (C) 1993 Ray Burr - Modified for Amiga FFS filesystem.
./fs/affs/super.c: * (C) 1992 Eric Youngdale Modified for ISO 9660 filesystem.
./fs/affs/super.c: * (C) 1991 Linus Torvalds - minix filesystem
./LICENSES/preferred/LGPL-2.0:Copyright (C) 1991 Free Software Foundation, Inc.
./LICENSES/preferred/LGPL-2.1:Copyright (C) 1991, 1999 Free Software Foundation, Inc.
./LICENSES/preferred/GPL-2.0: Copyright (C) 1989, 1991 Free Software Foundation, Inc.
./LICENSES/deprecated/GPL-1.0: Copyright (C) 1989 Free Software Foundation, Inc.
./LICENSES/deprecated/X11:Copyright (C) 1996 X Consortium
```

Ninka

File selector

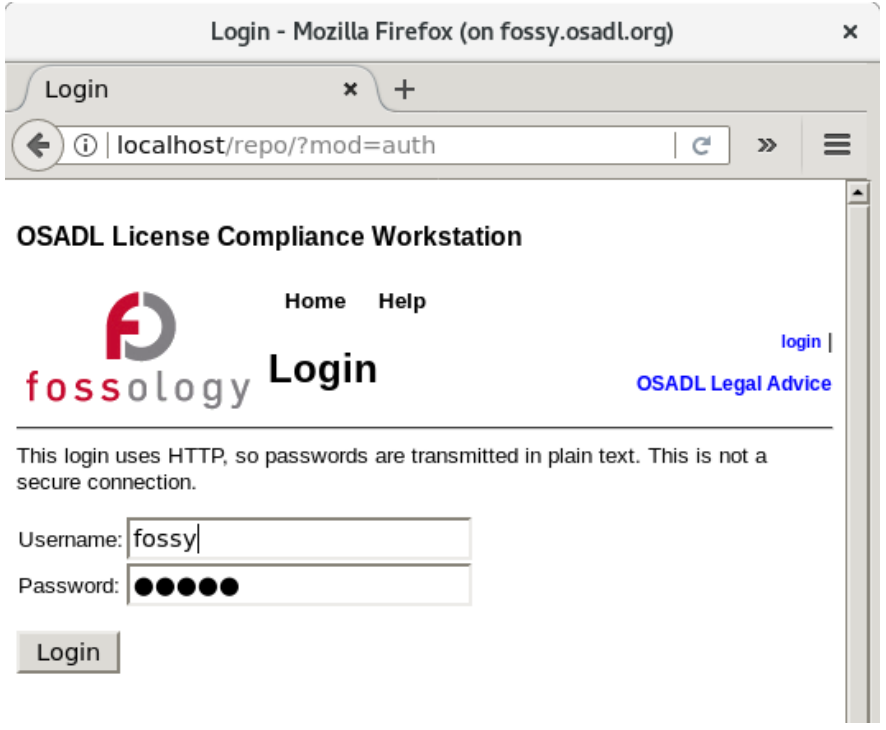


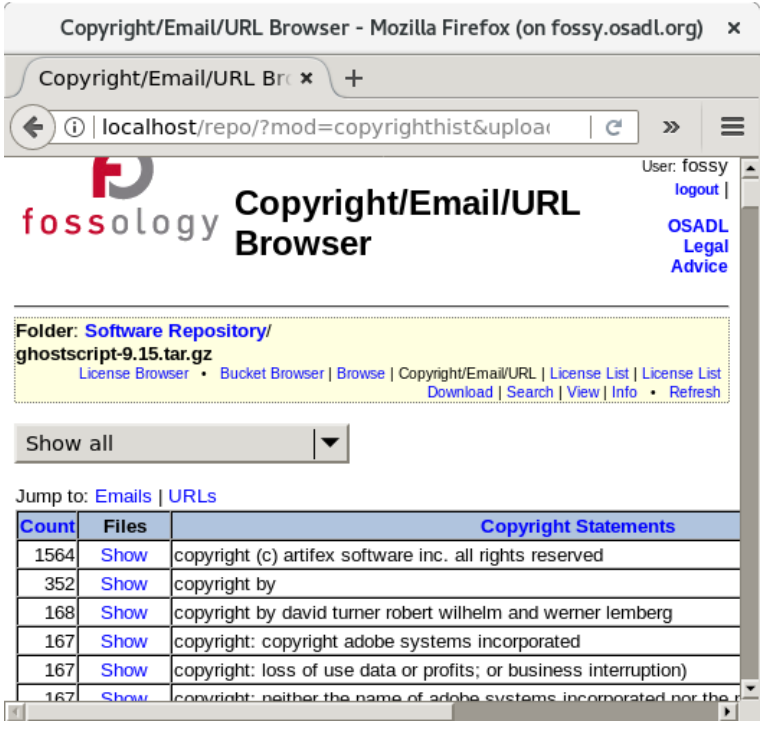
Result in browser



Fossology

Web based license management

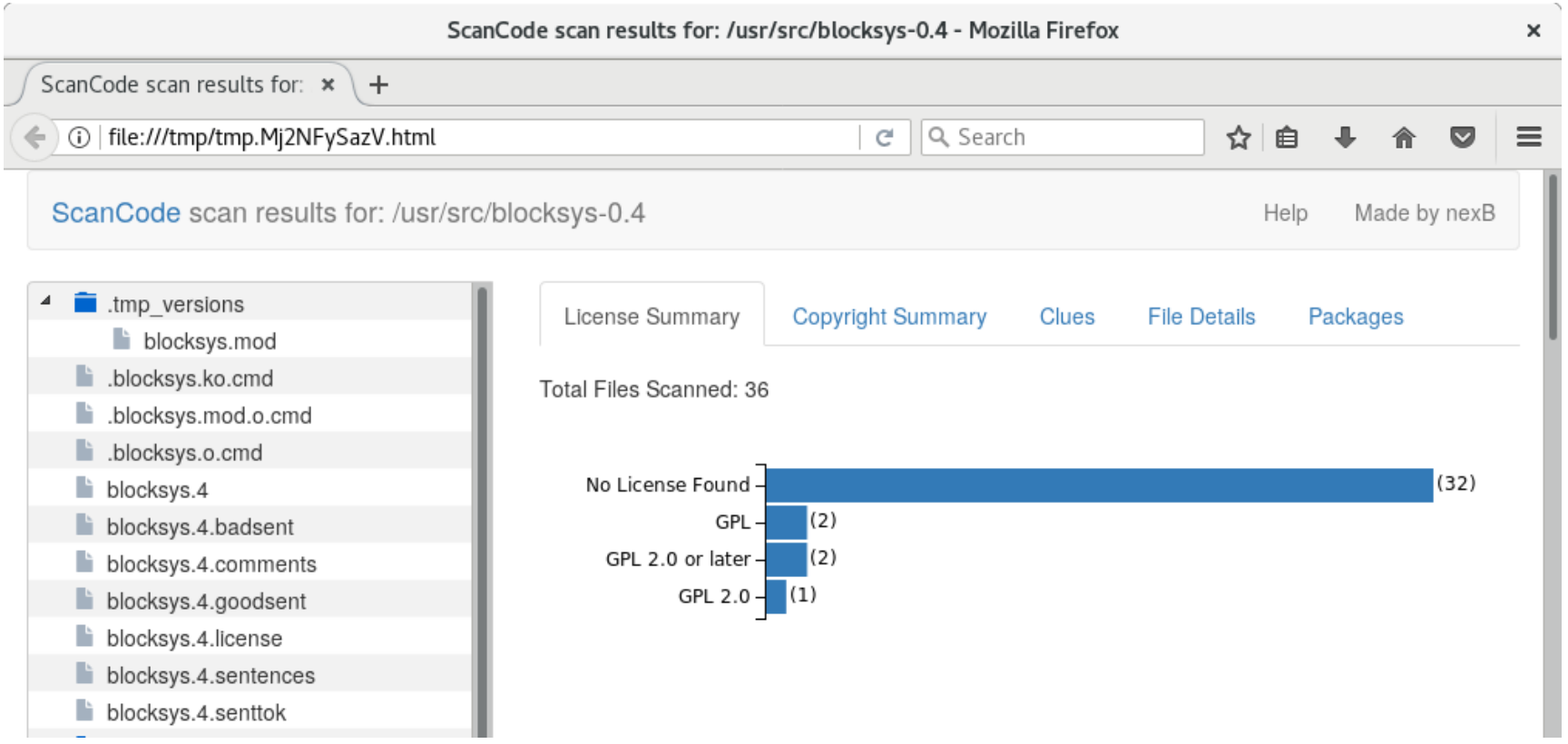




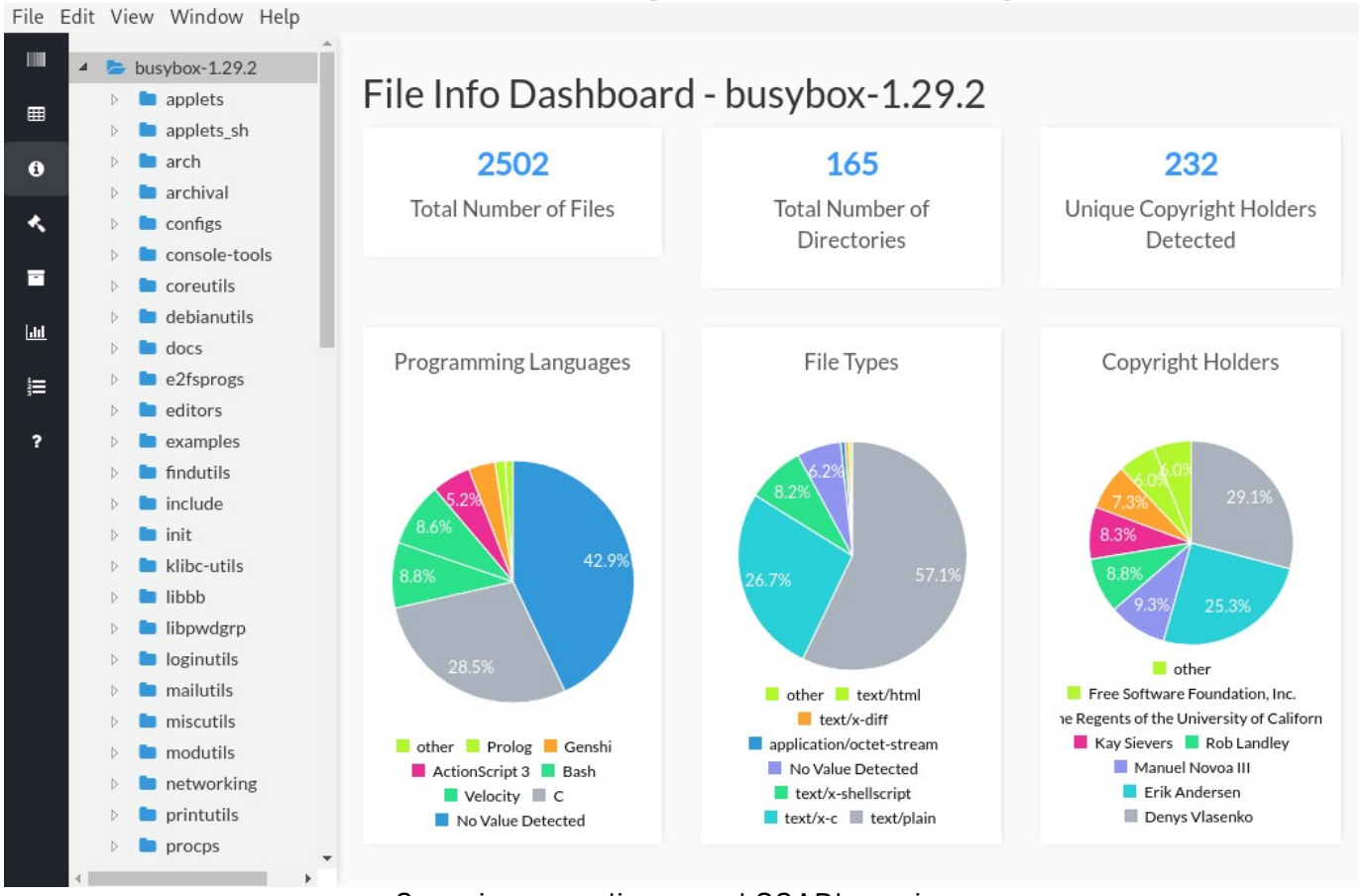
Scancode (command line tool)

```
Usage: scancode [OPTIONS] <input> <output_file>

  scan the <input> file or directory for origin clues and license and save results to the <output_file>.

  The scan results are printed to stdout if <output_file> is not provided. Error and progress is printed to stderr.

Options:
  -c, --copyright          Scan <input> for copyrights. [default]
  -l, --license            Scan <input> for licenses. [default]
  -p, --package            Scan <input> for packages. [default]
  -e, --email              Scan <input> for emails.
  -u, --url                Scan <input> for urls.
  -i, --info               Include information such as size, type, etc.
  --license-score INTEGER  Do not return license matches with scores lower than this score. A number between 0 and 100.
                           [default: 0]
  --license-text           Include the detected licenses matched text. Has no effect unless --license is requested.
  -f, --format <style>     Set <output_file> format <style> to one of the standard formats: json or json-pp or html or
                           html-app or spdx-tv or spdx-rdf or the path to a custom template  [default: json]
  --verbose                Print verbose file-by-file progress messages.
  --quiet                  Do not print summary or progress messages.
  -n, --processes INTEGER  Scan <input> using n parallel processes.  [default: 1]
  -h, --help               Show this message and exit.
  --examples               Show command examples and exit.
  --about                  Show information about ScanCode and licensing and exit.
  --version                Show the version and exit.
  --diag                   Include additional diagnostic information such as error messages or result details.
  --timeout INTEGER        Stop scanning a file if scanning takes longer than a timeout in seconds.  [default: 120]
  --max-memory INTEGER     Stop scanning a file if scanning requires more than a maximum amount of memory in megabytes.
                           [default: 1000]
```

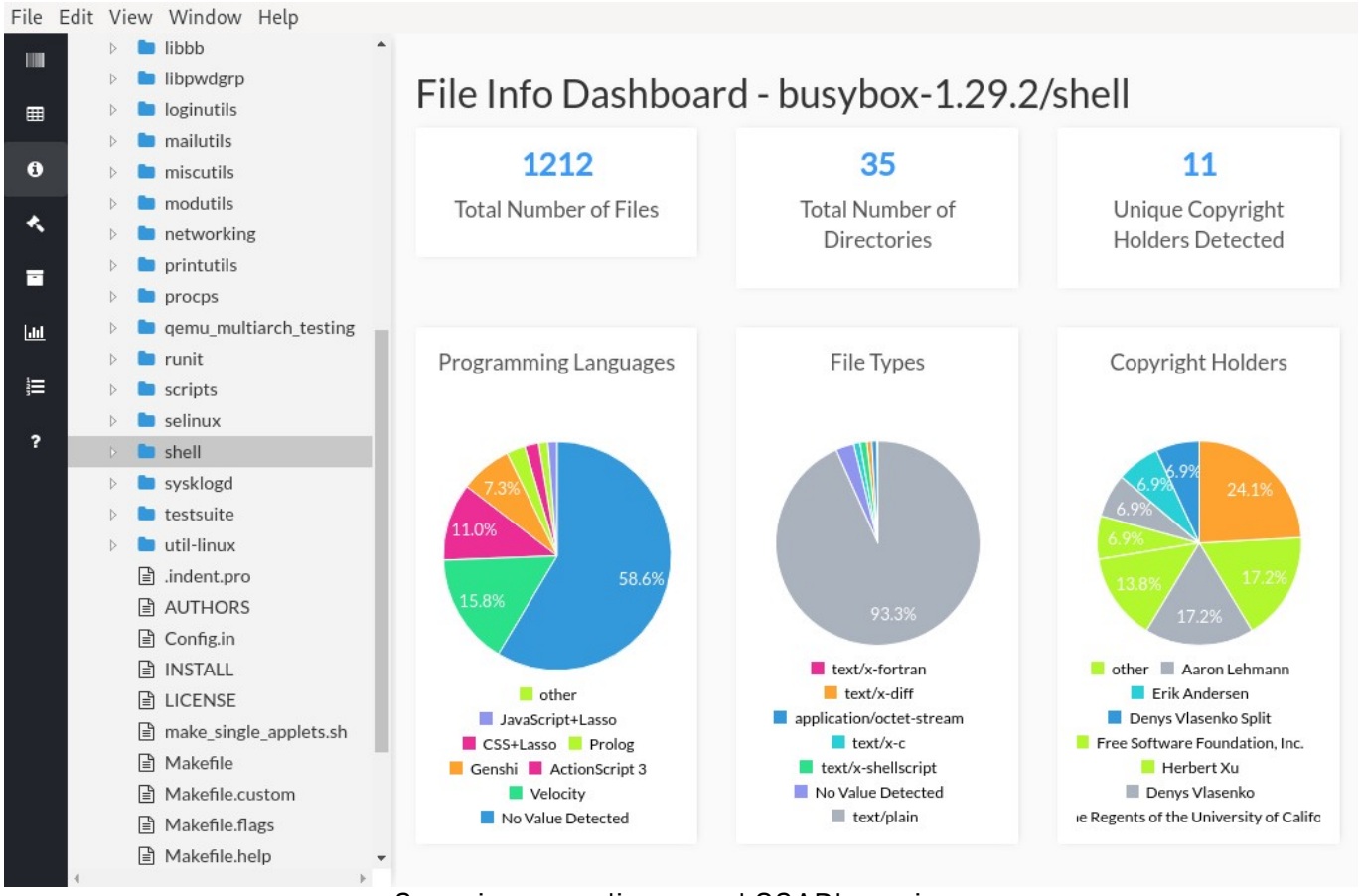
Scancode (result formatted for browser)



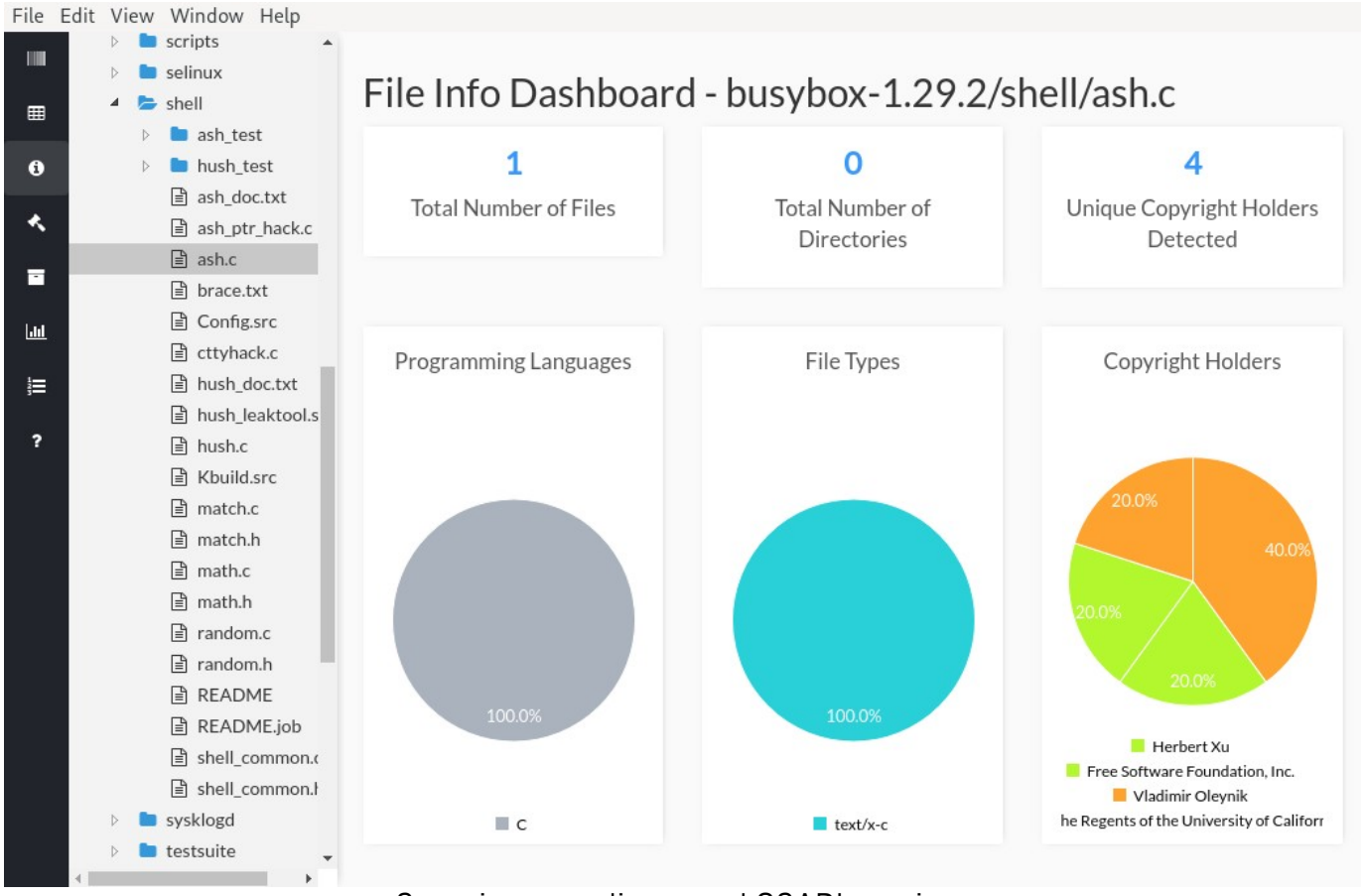
Scancode (workbench)



Scancode (workbench)



Scancode (result formatted for browser)



What is „forensic scanning“ and how does it work?

Step #1: Create data base and tool for forensic scanning:

- Collect each and every piece of Open Source software ever published (could be up to several hundreds of TBytes)
- Determine “meaningful” source code snippets and create hashes of them
- Store hashes along with original source code information in a data base

What is „forensic scanning“ and how does it work?

Step #2: Use the tool to discover yet unidentified code in own software (by negligence or by fraud):

- Determine “meaningful” source code snippets of own software and create hashes of them (same procedure as with foreign code in step #1)
- Search for the hashes in the data base
- Manually check the matches and remove false positive ones (this may be labor intensive)
- Take care of the correct findings (license/remove/rewrite code)

Conclusion

- Informational scanning is feasible with limited effort and provides all information that normally is needed to compliantly copy and distribute Open Source software. It, therefore, is generally recommended ("knowing your files").
- Forensic scanning usually requires a big effort, but certainly may provide crucial information, if needed. Forensic scanning, thus, should only be employed, if the individual conditions of software procurement let this appear meaningful ("knowing your enemy").

License conformance as part of a company's compliance procedures

HOT HEIDELBERG OSADL TALKS 2020

Scanning, compliance and OSADL services
Legal Heidelberg OSADL Talks, April 28, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg

OSADL
Open Source Automation Development Lab eG

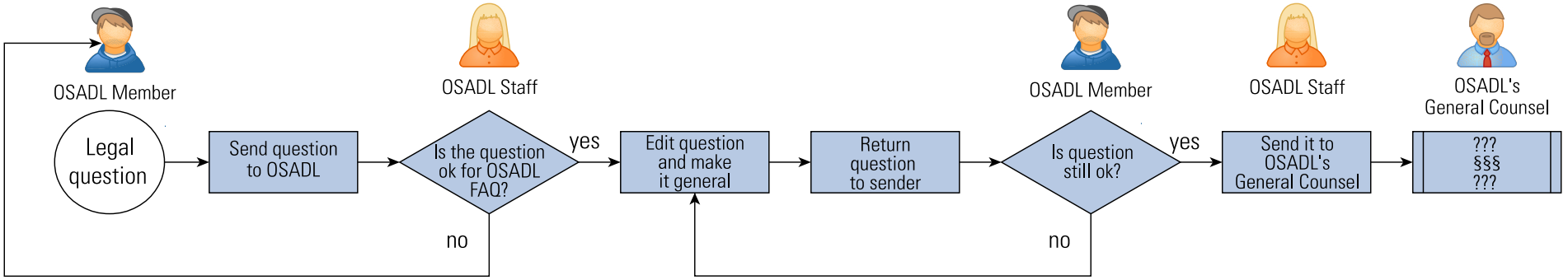
How can OSADL help with license compliance?

- OSADL FAQ
- OSADL Legal assessments
- OSADL Scanbook
- OSADL License Compliance Audit (LCA)
- OSADL Open Source License Obligations Checklists

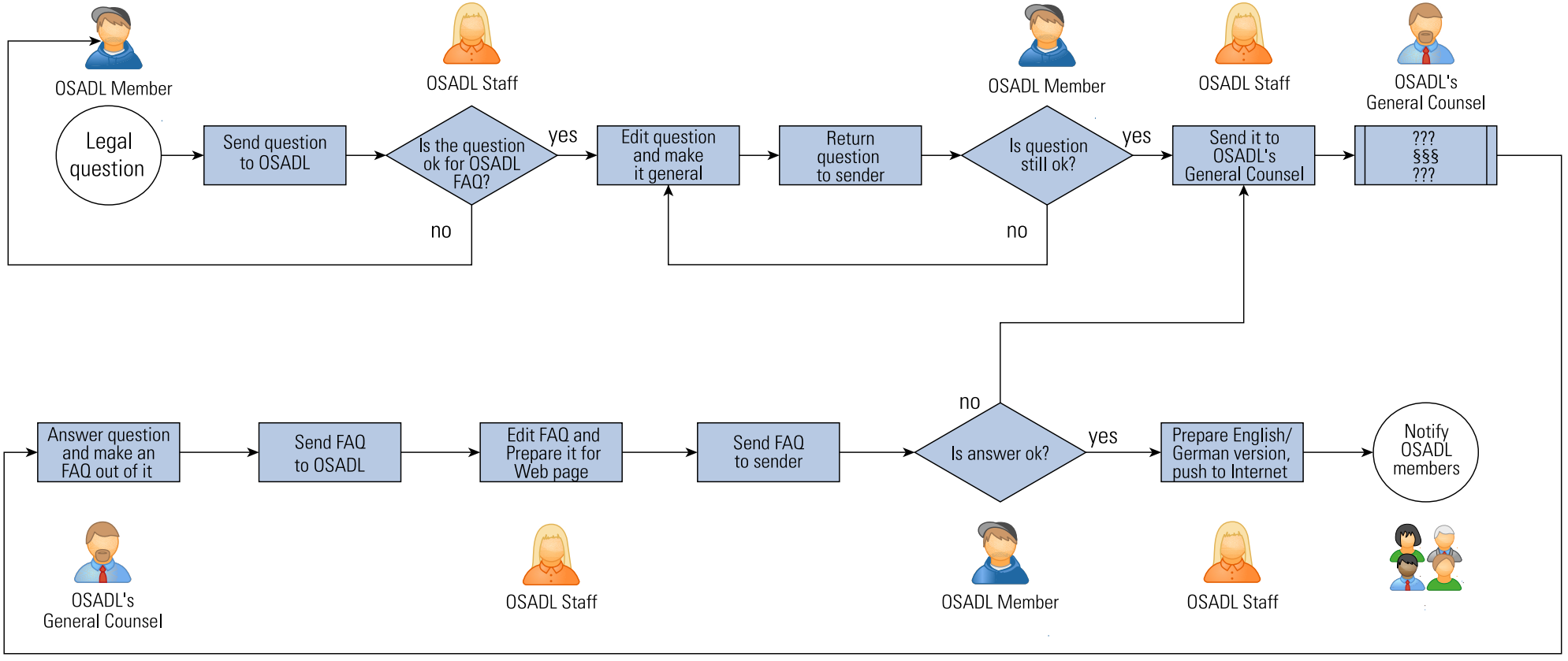
What are OSADL FAQ?

- Every employee of an OSADL member company may submit legal and technical questions of general interest to OSADL (office@osadl.org).
- Legal questions are edited and forwarded to OSADL's General Counsel, Dr. Till Jaeger, certified copyright and media law attorney.
- Dr. Jaeger's answer is then sent to the initial questioner and asked whether he or she is satisfied with the answer.
- If so, the question is added to OSADL's collection of legal FAQ in English and German language, and all OSADL members are notified. OSADL covers the attorney's fee. If not, OSADL tries to improve question and answer.

Receiving and editing the question ...



… processing the question and publishing it as FAQ



Selected example FAQ

- *General Aspects of License Agreements*

  Is it sufficient to specify a URL for the license text or does the complete license text have to be supplied to the customer? Is it otherwise useful to work with URLs in contracts?

- *Derivative Work*

  What is the impact of the copyleft of the GPL, if two independent software components (e.g. application and Linux kernel) are distributed together in a common file such as a zip archive file, .iso file, VM image file or an installation file for an embedded firmware?

General Aspects of License Agreements

"Is it sufficient to specify a URL for the license text or does the complete license text have to be supplied to the customer? Is it otherwise useful to work with URLs in contracts?"

Answer

Open Source licenses deal with the question if the license text has to be supplied with the product in paper form or as a file, or whether it is sufficient to specify a URL differently. Most licenses, like the GPL, require that the license text is supplied together with the product. The Landgericht München (Regional Court Munich) has explicitly deemed this as necessary in a judgment.

Also irrespective of the license terms requiring the inclusion of license texts in the product, the use of URLs is not recommended in contracts. Contracts are often created for long-term use, so that amendments to a URL can lead to a loss of relevant parts of the contract. In addition, the legal relevance of the URL is greatly reduced since it has to be proven which text was located under the URL at the time when the contract was concluded. This will often lead to practical problems as well as to easy manipulation.

Derivative Work

"What is the impact of the copyleft of the GPL, if two independent software components (e.g. application and Linux kernel) are distributed together in a common file such as a zip archive file, .iso file, VM image file or an installation file for an embedded firmware?"

Answer

Copyleft may also be relevant for completely independent programs, i.e. they must be licensed under the GPL altogether, if they are not distributed as separate works. This is particularly the case when the independent parts can no longer be separated easily so, in fact, a single work is created (e.g. in a single binary file).

However, the GPL makes it also clear that the "mere aggregation" of independent software components on the same storage or distribution medium does not result in a situation where copyleft becomes effective. This normally applies to archive and image files provided they can be unpacked easily in such a way that the original independent files become available again.

What are OSADL legal assessments?

- Should an FAQ be too complex to be answered in a couple of sentences, OSADL may decide to order a complete legal assessment on the topic.
- Currently, the following legal assessments are available:
  - February 13, 2009: Liability of a licensor of safety-critical Open Source software by Dr. Till Jaeger and Prof. Axel Metzger
  - September 16, 2011: GPL assessment with reference to “Hypervisor” by Dr. Till Jaeger
  - November 18: Business risk associated with participation in the OIN patent pool by Johanna Schwarz and colleagues at JBB, and Mishi Choudhary
  - December 18, 2018: Linux distributions by Dr. Till Jaeger
  - December 18: LGPL and third-party software by Dr. Till Jaeger
  - January 19: License obligations of Open Source software in the so-called “Cloud” by Dr. Till Jaeger and Prof. Axel Metzger

What is the OSADL scanbook?

- Standard scan tools such as scancode and fossology
- Armijn Hemel's Linux kernel delta scan
  - Trust *kernel.org*, but do not trust other code
  - Generate a hash data base of all original Linux code
  - Only scan code that does not belong to a valid hash
- Available as image or in a ready-to-use notebook
- Example of a license compliant generic redistribution of a Linux distribution

What is the OSADL License Compliance Audit (LCA)?

- To be audited and possibly certified:
  - Delivery of the product and accompanying documents
  - Linux kernel
  - C library
  - Relevant company documents
- Audited, but not certified:
  - Proprietarily licensed user-space applications
- Should irregularities be found:
  - Analysis of company processes and proposals for improvement

Example of an OSADL Legal Assessment:

License obligations under the LGPL-2.1 when linked to proprietary third-party software

HOT HEIDELBERG OSADL TALKS 2020

Scanning, compliance and OSADL services
Legal Heidelberg OSADL Talks, April 28, 2020, Online Session 3
Open Source Automation Development Lab (OSADL), Heidelberg

OSADL
Open Source Automation Development Lab eG

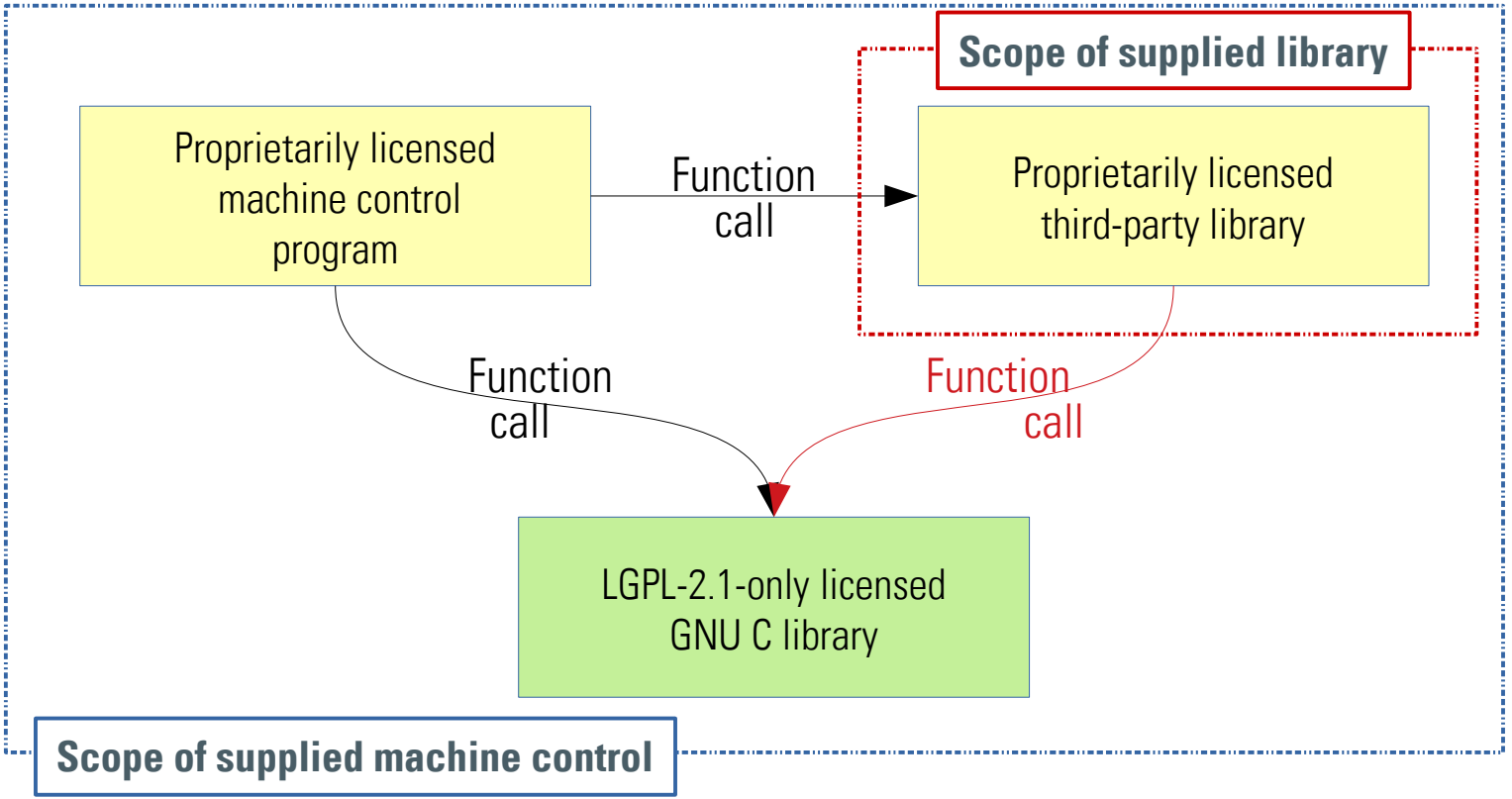
The OSADL member asked:

*With our product we are using a **binary proprietary third-party software** that requires an **LGPL-2.1 library** at runtime.*

*To distribute this third-party software along with the LGPL library we need to fulfill the **license obligations of the LGPL for linked works** also for this third-party software. The **supplier** of the third-party software however **refuses** to grant the permissions required to fulfill the obligations.*

What can we do?

Scenario


Scope of supplied library
Proprietarily licensed machine control program
Function call
Proprietarily licensed third-party library
Function call
Function call
LGPL-2.1-only licensed GNU C library
Scope of supplied machine control

Digression: Function calls

Digression: Function calls

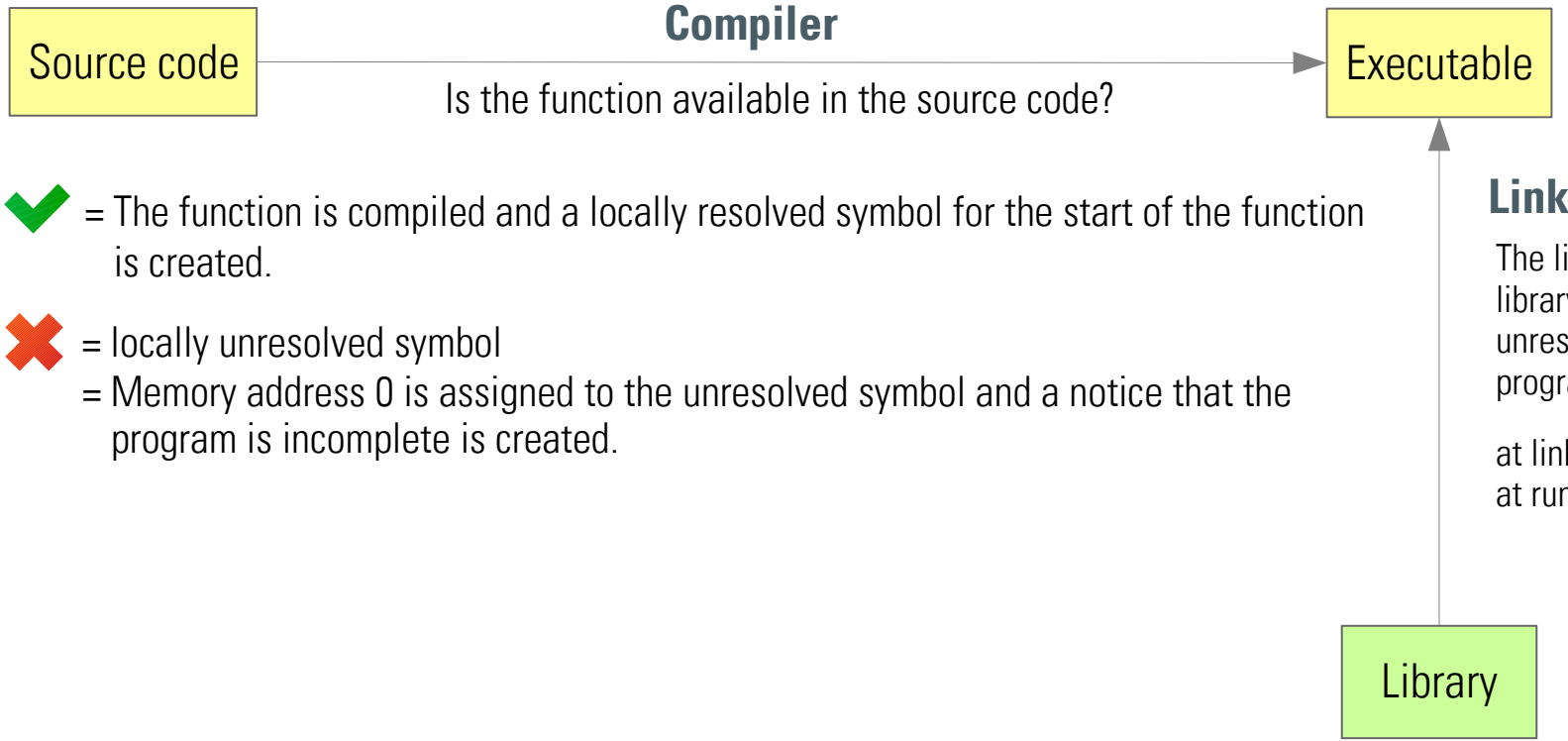
✔ = The function is compiled and a locally resolved symbol for the start of the function is created.

Digression: Function calls

✔ = The function is compiled and a locally resolved symbol for the start of the function is created.

✖ = locally unresolved symbol
= Memory address 0 is assigned to the unresolved symbol and a notice that the program is incomplete is created.

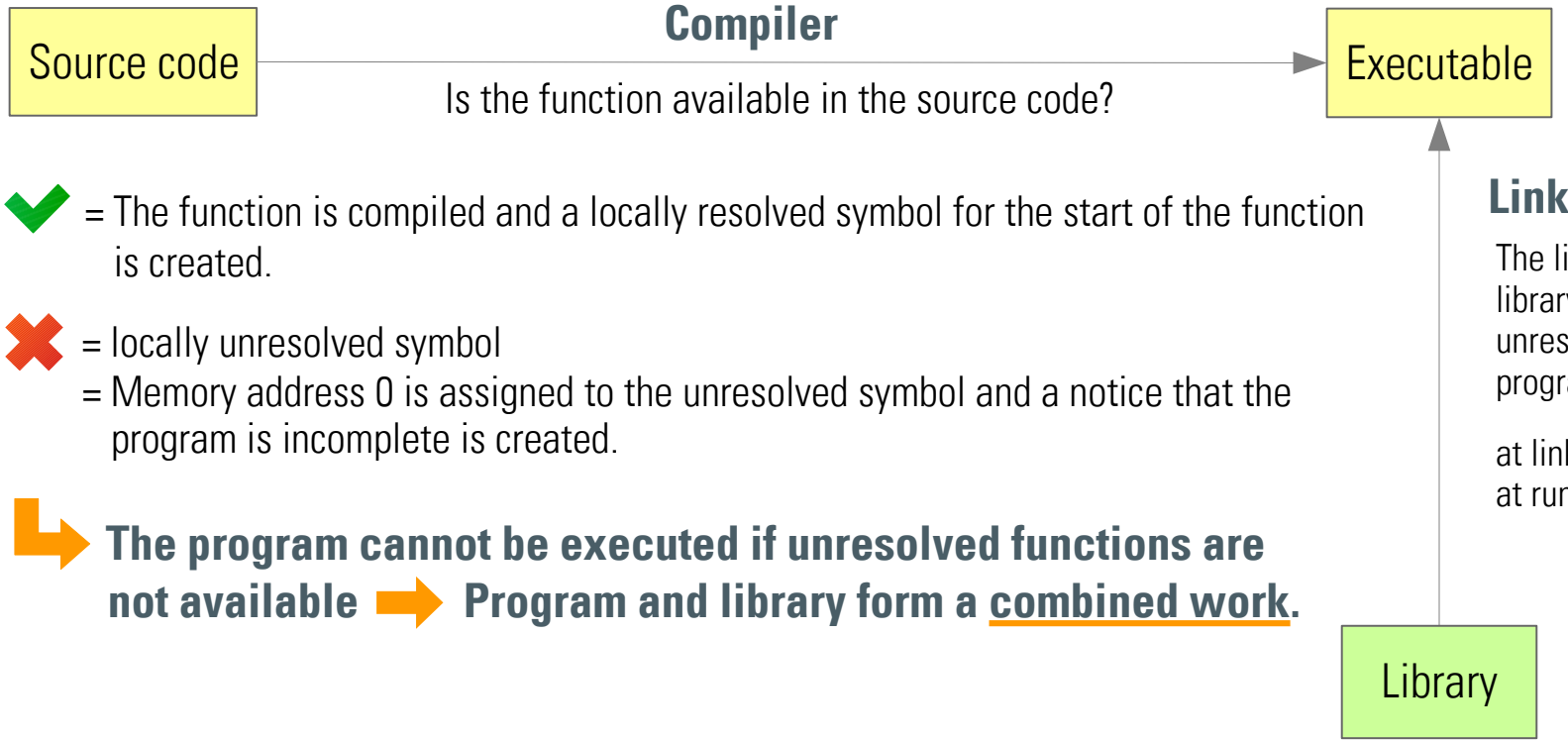
Digression: Function calls



= The function is compiled and a locally resolved symbol for the start of the function is created.

= locally unresolved symbol
= Memory address 0 is assigned to the unresolved symbol and a notice that the program is incomplete is created.

Digression: Function calls

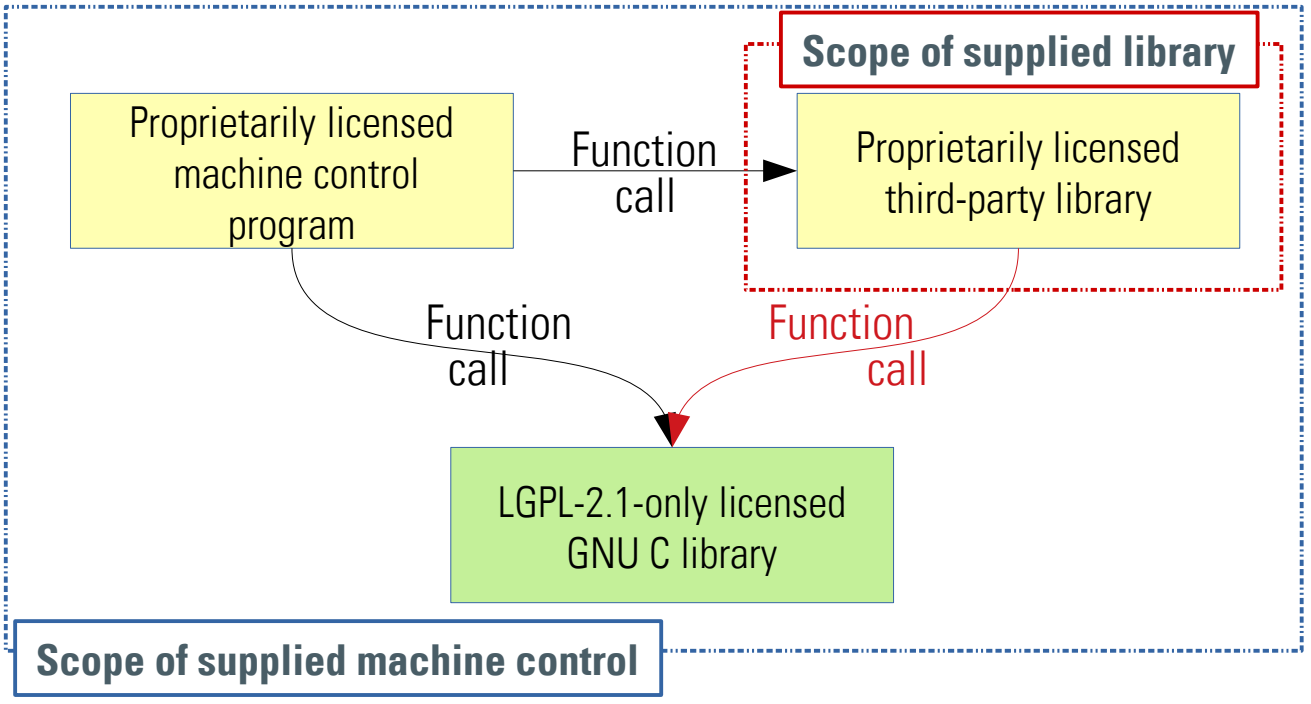


= The function is compiled and a locally resolved symbol for the start of the function is created.

= locally unresolved symbol
= Memory address 0 is assigned to the unresolved symbol and a notice that the program is incomplete is created.

The program cannot be executed if unresolved functions are not available ➡ Program and library form a combined work.

Scenario



Neither the proprietary machine control program nor the supplier's proprietary third-party library can be executed without functions provided by the dynamically linked GNU C library.

Original wording of LGPL-2.1 Article 6

As an exception [..], you may also combine or link a **"work that uses the Library"** with the Library to produce a work containing portions of the Library, and distribute that work under **terms of your choice, provided that the terms permit modification of the work** for the customer's own use **and reverse engineering** for debugging such modifications.

Original wording of LGPL-2.1 Article 6

As an exception [..], you may also combine or link a **"work that uses the Library"** with the Library to produce a work containing portions of the Library, and distribute that work under **terms of your choice, provided that the terms permit modification of the work** for the customer's own use **and reverse engineering** for debugging such modifications.

Is my software a “work that uses the Library”?

- ***objdump:*** A tool to disassemble and analyze binary executables.
- Locally unresolved symbols are marked as **“UND”** and the **function name** is displayed.
- Example: C program with the function

```
puts(“Hello world\n“);
```

- The function `puts` is not available within the program but requires a library (= unresolved symbol)

Example: Output of *objdump*

```
objdump -x hello
0000000000000000 F *UND* 0000000000000000 puts@@GLIBC_2.2.5
```

Example: Output of *objdump*

```
objdump -x hello
0000000000000000 F *UND* 0000000000000000 puts@@GLIBC_2.2.5
```

UNDefined: unresolved symbol

- Function name: `puts`
- Library that provides the function: GNU C Library *glibc* version 2.2.5
  - What is the license of the *glibc*?
  - What obligations does it require to fulfill?

Original wording of LGPL-2.1 Article 6

As an exception [..], you may also combine or link a **"work that uses the Library"** with the Library to produce a work containing portions of the Library, and distribute that work under **terms of your choice, provided that the terms permit modification of the work** for the customer's own use **and reverse engineering** for debugging such modifications.

Recommended addition to the company's Terms of Use to account for general Open Source licenses

If the Terms of Use contain clauses such as

```
Except, and only to the extent that may be permitted under applicable law,
you may not copy, decompile, disassemble, or reverse engineer the software
by any means whatsoever, or alter, modify, enhance, or create a derivative
work of the Software.
```

they must be modified by appending for example:

```
The above restrictions do not apply, if particular other licenses (for
example of Free and Open Source software) grant more extensive rights to
copy and distribute or explicitly permit reverse engineering under certain
conditions. In this case and in this context, the other licenses take
precedence over these Terms of Use.
```

Recommended additional licensing to account for LGPL-2.1 obligations

Required explicit licensing to disable a legal prohibition:

```
Modifications of the software for the user's own use and reverse engineering
for debugging such modifications are herewith permitted.
```

Limit the required permissions as far as possible:

```
However, forwarding the knowledge acquired during reverse engineering or
debugging to third parties is prohibited. Furthermore, it is prohibited to
distribute modified versions of the software. In any case, warranty claims
on the software will expire, as long as the customers cannot prove that the
defect would also occur without these modification.
```

Our supplier refuses to grant required permissions

The software cannot be distributed compliantly.

Question: Are there any legal means to force our supplier to grant the permissions?

Our supplier refuses to grant required permissions

The software cannot be distributed compliantly.

Question: Are there any legal means to force our supplier to grant the permissions?

Answer: **NO**

Our supplier refuses to grant required permissions

The software cannot be distributed compliantly.

Question: Are there any legal means to force our supplier to grant the permissions?

Answer: **NO**

Defect of title

- Notice of defects: Request for rectification
- Options: Withdrawing from the contract, reclaiming the purchase price

Our supplier refuses to grant required permissions (2)

Question: What can happen if we grant our customers the additional permissions without the consent of our supplier?

Our supplier refuses to grant required permissions (2)

Question: What can happen if we grant our customers the additional permissions without the consent of our supplier?

Answer: **License violation**, as you cannot grant rights that you do not possess.

Our supplier refuses to grant required permissions (2)

Question: What can happen if we grant our customers the additional permissions without the consent of our supplier?

Answer: **License violation**, as you cannot grant rights that you do not possess.

↳ Claims (of your supplier) under copyright law:
e.g. injunction, abatement and removal, right of information, damages

Practical recommendations

Purchase department (before ordering!)

Inquiry to third-party software suppliers which additional components that are not part of the delivery are required to run the software.

Receiving department (before the expiry of potential return dates!)

Check which additional components that are not part of the delivery are required to run the software, e.g. using *objdump*.