SYNOPSYS®

Silicon to Software™

# Remediating Open Source Software Issues in M&A

Some recommendation how to remediate open source license compliance issues

Jari Koivisto
Phil Odence

# Speakers

**Jari Koivisto**
Open Source Due
Diligence Consultant

**Phil Odence**
General Manager, Black
Duck Audits, Synopsys

# Disclaimer

- I am not a lawyer
  - If in any doubt, consult your legal department or seek help from outside legal counsels
- These are just recommendations
  - For most of the issues there can be several ways to remediate

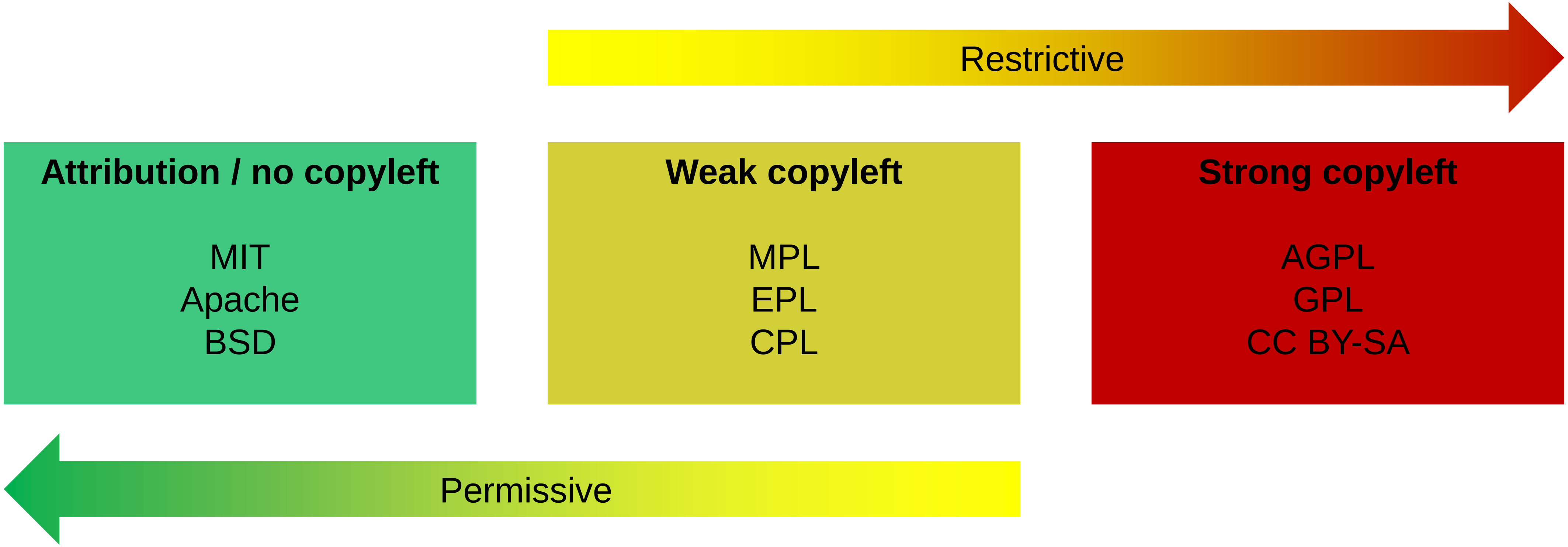
# Assumptions

- Everyone on this call knows the basics of open source licensing
  - Copyright, strong / weak copyleft, etc.
- Everyone knows basics of software products, different deployment models, etc.
  - Shipping product vs. SaaS
- The coming examples are mainly for products that are shipped
  - SaaS products are easier and AGPL and SSPL licensed components are typically the ones one needs to be more careful, also non-commercial and components without any license need attention

AGPL – Affero GPL
SSPL – Server Side Public License

# Open Source licenses

## Copyleft strength spectrum



Restrictive

**Attribution / no copyleft**

MIT
Apache
BSD

**Weak copyleft**

MPL
EPL
CPL

**Strong copyleft**

AGPL
GPL
CC BY-SA

Permissive

# Agenda

1. How to get started once audit report is available?
2. Strong copyleft components
   - Embedded
   - Full components
3. Snippets under CC BY-SA license
4. Components under LGPL
5. GPLv3, LGPLv3 special considerations
6. Non-commercial use licenses
7. No license components
8. Special cases
   - Many license options
   - Different versions – different license
   - Kernel Loadable Modules
9. Tips what to do after the audit
10. Q&A

# Actions needed once code audit done

- Code audit done for internal or external (typically for M&A) codebase
- A number of issues found
  - Depending of the size of the audit the audit report may have thousands of line items
  - The biggest audit report: >2’500 items and P1+P2 items >1’000
- Analyze the audit report and decide, what are the issues that need attention
  - Questionnaires, find out the use of the components, etc.
  - Concentrate on components that are under copyleft licenses or components without any license

# Questionnaire questions 1/2

| | Column 1 | Column 2 | Column 3 | Column 4 | Column 5 |
|---|---|---|---|---|---|
| **Component name, license, etc.** | **Is this component developed by <target_name> or is this 3rd party code?** | **Is this component used / needed in the product or to build or test the product?** | **If component not used or needed, how long its removal from codebase will take?** | **Is this component distributed to customers, vendors or other 3rd parties?** | **If component is distributed, please describe how it is done?** |
| Component #1 (GPLv2) | | | | | |
| Component #2 (LGPLv2.1) | | | | | |
| Etc. | | | | | |

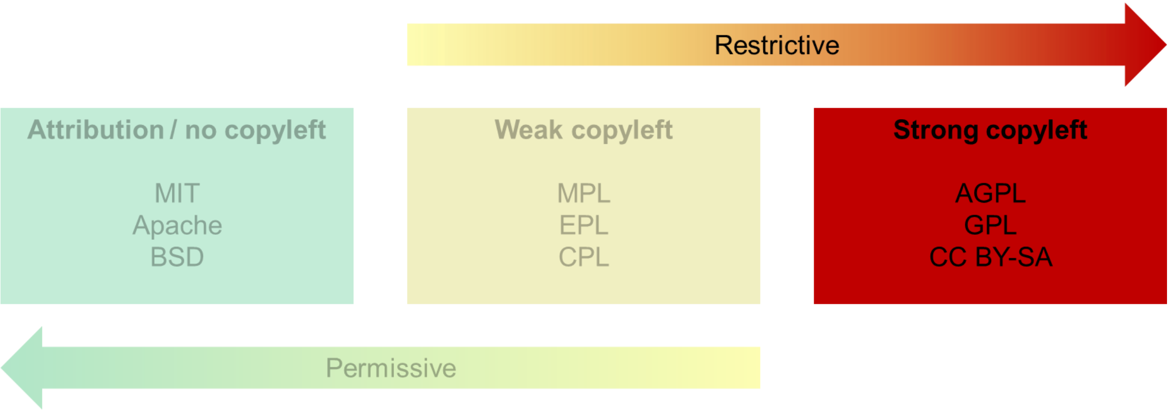
# Questionnaire questions 2/2

| Column 6 | Column 7 | Column 8 | Column 9 | Column 10 |
|---|---|---|---|---|
| **How the compliance with license requirements is ensured?** | **In which product(s) this component is used?<br>How does it interact with <target_name> product?** | **How important this component is for the product? (Very important, medium, low)** | **Are there modifications in this component?** | **If there are modifications, is there any critical IP in those modifications and is proprietary code needed to build, install or run the modification?** |
| | | | | |
| | | | | |

# Strong copyleft components

# Embedded GPL and similar strong copyleft snippets

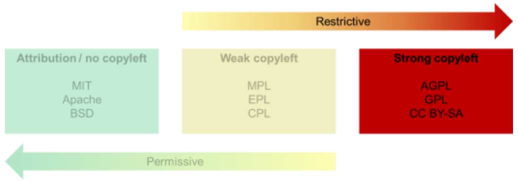
- Remediation ideas:
  - Can the component be open sourced?
    - If yes -> Package the component and make ready for release
    - If no -> Remove and rewrite (this is the typical remediation recommendation)
    - If no -> Contact the code copyright owner and ask to purchase a commercial license
      - Risk: code copyright owner will know that there is a potential GPL violation



GPL – GNU General Public License

# GPL and other strong copyleft licensed components

- Some things to find out:
  1. Is the component running in a separate process?
  2. How is the component accessed?
     - Inter-process communication (IPC), e.g. sockets, pipes typically o.k.
  3. Is intimate knowledge needed about the GPL component?
  4. Are there modifications?
- Some remediation thoughts:
  - Make sure that the GPL component runs in a separate process
  - Communication between proprietary code and GPL through sockets, pipes, etc.
  - Find out if the same functionality can be found using a component that is under a permissive license
  - If needed, consider if the proprietary code can be released
  - Remove and rewrite
  - Contact code owner and ask to purchase a commercial license (risk of revealing the potential GPL violation)



Restrictive

Attribution / no copyleft
MIT
Apache
BSD

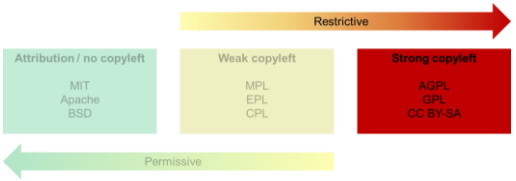
Weak copyleft
MPL
EPL
CPL

Strong copyleft
AGPL
GPL
CC BY-SA

Permissive

# GPL with Exceptions

- GPL may have Exceptions added to it
  - E.g. Classpath Exception, but there are plenty
- Remediation thoughts:
  - Does the Exception apply?
    - If yes -> normally you are good
    - If no -> normal GPL compliance



Restrictive

Attribution / no copyleft

MIT
Apache
BSD

Weak copyleft

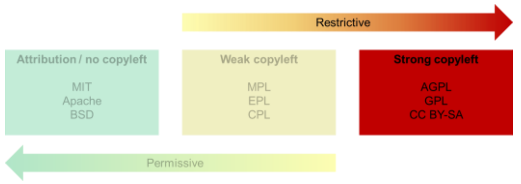
MPL
EPL
CPL

Strong copyleft

AGPL
GPL
CC BY-SA

Permissive

# CC BY-SA snippets



Restrictive

Attribution / no copyleft

MIT
Apache
BSD

Weak copyleft

MPL
EPL
CPL

Strong copyleft

AGPL
GPL
CC BY-SA

Permissive

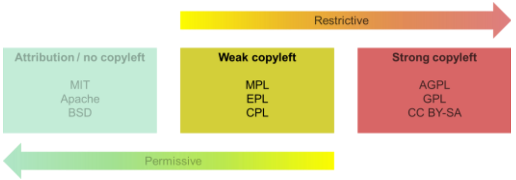
- Typical source is StackOverflow, other typical source is Wikipedia
- A couple of options for remediation:
  1. Is code copy-pasted or is the source used as an inspiration?
  2. Is the code very generic -> if yes, normally o.k.
  3. Find out if StackOverflow is **not** the original source of the snippet. If snippet is originally coming from a source that has a permissive license -> typically o.k.
  4. Is the component something that can be open sourced?
     - If yes -> package the component and make ready for release
     - If no -> Remove and rewrite (common remediation action)

CC BY-SA – Creative Commons Attribution-ShareAlike license
**ShareAlike** – *If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.*

# LGPL

## Lesser General Public License

- Weaker copyleft than GPL
- Questions to ask:
  - Are there any modifications in the component?
  - How is the component linked?
    - Static / dynamic linking
- If there are modifications, one needs to package the modified LGPL code and make it available under LGPL if requested
- If static linking, can dynamic linking be used?



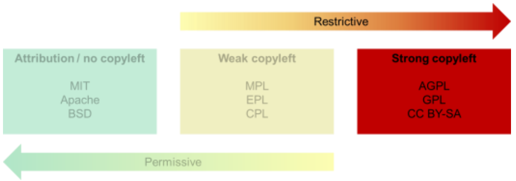
Restrictive

**Attribution / no copyleft**
MIT
Apache
BSD

**Weak copyleft**
MPL
EPL
CPL

**Strong copyleft**
AGPL
GPL
CC BY-SA

Permissive



Restrictive

Attribution / no copyleft
MIT
Apache
BSD

Weak copyleft
MPL
EPL
CPL

Strong copyleft
AGPL
GPL
CC BY-SA

Permissive

# GPLv3, LGPLv3 1/2

## Anti-tivoization

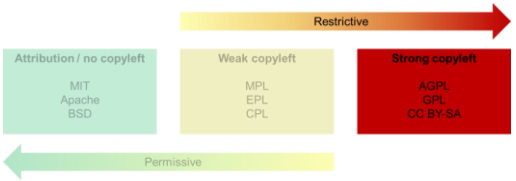
- *Tivoization is the creation of a system that incorporates software under the terms of a copyleft software license (like the GPL), but uses hardware restrictions or digital rights management to prevent users from running modified versions of the software on that hardware.*

  **Source:** https://en.wikipedia.org/wiki/Tivoization

- The goal of v3 license is to prevent Tivoization
  - The anti-tivoization clause was limited **not to apply** when the SW is distributed to a business
  - The anti-tivoization clause applies to consumer products

http://gplv3.fsf.org/dd3-faq (last modified 2007-06-26):
***Section 6: Why do distributors only have to provide Installation Information for User Products?***

> *Some **companies** effectively outsource their entire IT department to another company. Computers and applications are installed in the company's offices, but managed remotely by some service provider. In some of these situations, **the hardware is locked down; only the service provider has the key**, and the customers consider that to be a desirable security feature.*
>
> *We think it's unfortunate that people would be willing to give up their freedom like this. But they should be able to fend for themselves, and the market provides plenty of alternatives to these services that would not lock them down. As a result, we have introduced **this compromise to the draft: distributors are only required to provide Installation Information when they're distributing the software on a User Product**, where the customers' buying power is likely to be less organized.*
>
> *This is a compromise of strategy, and not our ideals. Given the environment we live in today—where Digital Restrictions Management is focused largely in consumer devices, and everyone, including large companies, is becoming increasingly worried about the effects of DRM thanks to recent developments like the release of Microsoft's Windows Vista—we think that the proposed language will still provide us with enough leverage to effectively thwart DRM. We still believe you have a fundamental right to modify the software on all the hardware you own; the preamble explains, "If such problems [as locked-down hardware] arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users."*



Restrictive

Attribution / no copyleft
MIT
Apache
BSD

Weak copyleft
MPL
EPL
CPL

Strong copyleft
AGPL
GPL
CC BY-SA

Permissive

# GPLv3, LGPLv3 2/2

## Anti-tivoization

- Remediation:
  - Consumer products (such as STB / DVR / smart watch / etc.):
    - Find out if the same functionality can be found using components that are not under v3 license
  - If a product is **not** shipped to a consumer but to another business
    - Normally v3 Tivoization prevention clause is not a problem, but normal GPL and LGPL issues need to be taken into account

# Non-commercial use license

## Shipping and SaaS products

# Components with Non-Commercial license

- E.g. CC BY-NC or sometimes someone adds non-commercial addendum to MIT, BSD, etc. licenses
- Some remediation options:
  1. Find out if there is a commercial version available
  2. Find out if there are other components with similar functionality and that can be used for commercial products
  3. Remove and write the functionality in-house

# No license components

## Shipping and SaaS products

# Example 1/2

| | Column 1 | Column 2 | Column 3 | Column 4 | Column 5 |
|---|---|---|---|---|---|
| **Component name, license, etc.** | **Is this component developed by <target_name> or is this 3rd party code?** | **Is this component used / needed in the product or to build or test the product?** | **If component not used or needed, how long its removal from codebase will take?** | **Is this component distributed to customers, vendors or other 3rd parties?** | **If component is distributed, please describe how it is done?** |
| Component #1 (Unknown) | 3rd party | Yes, part of the product | N/A | Distributed | Embedded in a device |
| Component #2 (Unknown) | 3rd party | Yes, part of the product | N/A | Distributed | Embedded in a device |

# Example 2/2

| Column 6 | Column 7 | Column 8 | Column 9 | Column 10 |
|---|---|---|---|---|
| **How the compliance with license requirements is ensured?** | **In which product(s) this component is used?<br>How does it interact with <target_name> product?** | **How important this component is for the product? (Very important, medium, low)** | **Are there modifications in this component?** | **If there are modifications, is there any critical IP in those modifications and is proprietary code needed to build, install or run the modification?** |
| **We did not find any license, so we assumed public domain** | Product xyz<br>Embedded in device | Medium | Yes | No |
| **We did not find any license, so we assumed public domain** | Product zyz<br>Linked library | Very important | No | No |

# Components without a license

## No license -> no right to use

- Typical sources: blog posts, GitHub
- Some remediation options:

  1. Ask the copyright owner if she/he can add a license
     - Often times the idea was that the code is under some no copyleft license or public domain, but the one who wrote the code forgot the license or statement that it is public domain
  2. Ask copyright owner if it is o.k. to use the code in commercial product
     - Get this in written (email normally o.k.)
  3. Remove and rewrite

No license ≠ public domain

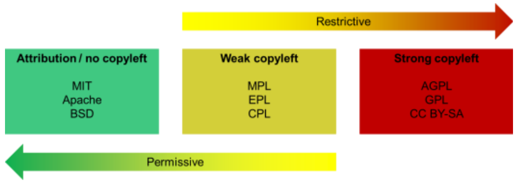
# Some special cases

# Many license options

- If a component has several (2 or more) license options:
  - Typically one selects the more permissive license, but not always, e.g. because of license compatibility

# Different versions have different licenses

- Same component, different versions have different licenses
  - Newer version may have more or less permissive license
    - Newer version **more** permissive license (e.g. old GPL, new MIT) -> update the component to the latest version (MIT)
    - Newer version **less** permissive license: need to be careful with component updates



Restrictive

**Attribution / no copyleft**

MIT
Apache
BSD

**Weak copyleft**

MPL
EPL
CPL

**Strong copyleft**

AGPL
GPL
CC BY-SA

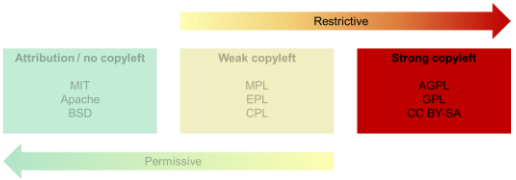
Permissive

# Linux Kernel Loadable Modules (KLMs)

## a.k.a. Loadable Kernel Modules (LKMs)

- KLM is a module that can be linked to the Kernel while it is running
- KLMs are used to implement:
  - Drivers
    - Device, filesystem or network
  - System calls

- KLM:
  - Are they GPLv2 or not?
  - Safe bet is to say that KLMs are under GPLv2
    - afaik there is no case law on this and therefore it is still a gray area



# Next steps after audit and remediation plan

## Acquisition case

1. Wait for the Legal Close
   - Typically weeks or months
2. Integration phase:
   - Good practice: OSDD Lead participates at least for the first couple of weeks
   - More knowledge about product and codebase -> alternative (better) remediation solutions may come up
   - Once all remediation steps done -> another scan & audit recommended

# Contacts

https://www.linkedin.com/in/jarikoivisto/
jari.p.koivisto@iki.fi
+41 78 7479791 (Central Europe Time zone)

# Q&A

Silicon to Software™

# Thank You