



Automated License Compliance

1	Contents	
2	Scanners	3
3	Tooling	3
4	CI Pipeline integration	4
5	Binary to source file mapping	5
6	Tooling	5
7	CI Pipeline integration	6
8	Binary Licensing Reporting	7
9	Tooling	7
10	CI Pipeline integration	7
11	Step-by-step process	7
12	During package source build on Gitlab CI pipelines	8
13	During package build on OBS	8
14	During image generation on Gitlab CI pipelines	9
15	A Linux system such as those assembled by Apertis contain components licensed	
16	under many different licenses. These various licenses impose different conditions	
17	and it is important to understand to a good degree of fidelity the terms under	
18	which each component is provided. We are proposing to implement an auto-	
19	mated process to generate software Bills Of Materials (BOMs) which detail	
20	both the components used in Apertis and the licensing that applies to them.	
21	Licensing isn't static, nor is it always as simple as all the components from a	
22	given source package deriving the same license. Packages have been known to	
23	change licenses and/or provide various existing or new components under dif-	
24	ferent terms. Either now or at some point in the future, the licenses of some of	
25	the components in Apertis may start to be provided under terms that Apertis	
26	may wish to avoid ¹ . For example, by default Apertis is careful not to include	
27	components to be used in the target system that are licensed under the GPL	
28	version 3, the licensing terms wouldn't be acceptable in Apertis'target markets.	
29	In order to take advantage of new functionality and support being developed in	
30	the software community, Apertis needs to incorporate newer versions of exist-	
31	ing software packages and replace some with alternatives when better or more	
32	suitable components are created. To ensure that the licensing conditions remain	
33	favorable for the use cases targeted by Apertis, it is important to continually	
34	validate that the licensing terms under which these components are provided.	
35	These licensing terms should be documented in a way that is accessible to Aper-	
36	tis'users.	
37	Debian packages by default track licensing on a per source package level. The	
38	suitability of a package is decided at that level before it is included in Debian,	

¹<https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/>

39 which meets the projects [licensing goals](#)². Apertis will continue to evaluate
40 licensing before the inclusion of source packages in the distribution, but also
41 wishes to take a more nuanced approach, tracking licensing for each file in each
42 of its binary packages. By tracking licensing to this degree we can look to
43 exclude components with unsatisfactory licensing from the packages intended
44 for distributed target systems, whilst still packaging them separately so they
45 may be utilized during development. A good example of this situation is the
46 `gcc` source package and the `libgcc1` binary package produced by it. Unlike the
47 other artifacts produced by the GCC source package, the `libgcc1` binary package
48 is not licensed under the stock GPLv3 license, a [run time exception](#)³ is provided
49 and it is thus fine to ship it on target devices. The level of tracking we are
50 providing will detect such situations and will offer a straight forward way to
51 resolve them, maintaining compliance with the licensing requirements.

52 To achieve this 2 main steps need to be taken:

- 53 • Record the licensing of the project source code, per file
- 54 • Determine the mapping between source code files and the binary/data
55 files in each binary package

56 These steps have been integrated into our CI pipelines to provide early detection
57 of any change to the licensing status of each package. Extending our CI pipelines
58 also enables developers to learn about new issues and to solve them during the
59 merge request development flow.

60 Scanners

61 Tooling

62 The current tool used to record the license of each package is the command
63 line tool `scan-copyrights` from `libconfig-model-dpkg-perl`⁴ which is a standard
64 Debian tool. It parses the output from `licensecheck`⁵ to generate a `DEP5`⁶. More
65 information about it can be found in [license scanning](#)⁷.

66 Based on the challenges in detecting the right licenses for source codes, other
67 tools are being evaluated, with FOSSology being one of the most interesting
68 ones.

69 FOSSology is an Open Source server based tool which provides a web front-end
70 that is able to scan through source code (and to a degree binaries) provided to
71 it, finding license statements and texts. To achieve this FOSSology employs a

²https://www.debian.org/social_contract.html#guidelines

³<https://apertis-website-0b3586.pages.apertis.org/policies/license-exceptions/#gcc8>

⁴<https://gitlab.apertis.org/pkg/libconfig-model-dpkg-perl>

⁵<https://gitlab.apertis.org/pkg/licensecheck>

⁶<https://dep-team.pages.debian.net/deps/dep5/>

⁷<https://apertis-website-0b3586.pages.apertis.org/architecture/application/license-scanning/>

number of different scanning techniques to identify potential licenses, including using matching to known license texts and keywords. The scanning process errs on the side of caution, generating false positives over missing potential licensing information, as a result it will be necessary to “clear” the licenses that are found, deciding whether the matches are valid or not. The scanning and clear process during the first time is more time consuming and requires special attention, however, subsequent runs should be much faster as FOSSology is able to use previous decisions to find the license information. Once completed, FOSSology records the licensing decisions and can apply this information to updated scans of the source. It is anticipated that, after an initial round of verification, FOSSology will only require additional clearing of license information should the scan detect new sources of potential licensing information in an updated projects source or when new packages are added to Apertis. It is possible to export and import reports which contain the licensing decisions that have previously been made, if a trusted source of reports can be found then these could also be imported, potentially reducing the work required.

FOSSology is backed by the Linux Foundation, it appears to have an active user and developer base and a significant history and it is the de-facto Open Source Software solution for license compliance. As such, it is felt that this tool is likely to be maintained for the foreseeable future.

As this tool provides a web based UI, it presents an additional advantage, as it makes it easier for non-technical users, such as auditors or lawyers, to access and manage the reports, allowing a smooth integration in an audit process.

For all the reasons mentioned above we understand this would a good choice for improving the current Apertis workflow.

Apertis currently uses scan-copyrights as default scanner. Initial integration of FOSSology is already available but not enabled.

CI Pipeline integration

In order to avoid manual tasks, the license detection needs to be integrated into the CI process.

Currently, `scan-copyrights` is integrated in the CI script `ci-license-scan`⁸ which is automatically triggered on package upgrades. This is straight forward since `scan-copyrights` is a command line tool.

FOSSology provides a [REST API](https://www.fossology.org/get-started/basic-rest-api-calls/)⁹ to enable such integration.

FOSSology is able to consume branches of git repositories, thus allowing scanning of the given source code straight from GitLab. This process should be triggered after updating a package from external sources, as in this cases a

⁸<https://gitlab.apertis.org/infrastructure/apertis-docker-images/-/blob/apertis/v2023dev3/package-source-builder/overlay/usr/bin/ci-license-scan>

⁹<https://www.fossology.org/get-started/basic-rest-api-calls/>

license change can be introduced. A report will be generated and retrieved, using the REST API, which describes (among other things) the licensing status of each file. The report can be generated in a number of formats, including various SPDX flavors that are easily machine parsable, using [DEP5](https://dep-team.pages.debian.net/deps/dep5/)¹⁰ as the preferred option. It is suggested that each component should require a determination of the licensing to have been made for every file in the project. Due to the large volume of licensing matches that will result from the initial licensing scan, we recommend that the absence of license information initially generates a warning. In some cases, to achieve the fine grained licensing information desired, the licensing of some files may need to be clarified with the components author(s). Once an initial pass of all Aptis components had been made we would expect missing license information to result in an error, as such errors would be as a result of new matches being found, which would need to be resolved in FOSSology before CI would complete without an error. The generated report should be saved in the Debian metadata archive so that it is available for the following processing.

In a possible future integration, the adoption of FOSSology would be gradual and in parallel with the current license scanning process in order to compare the results and improve the workflow. At a later stage, once the new process is fully reviewed and tested with all the packages in the target repository, FOSSology would potentially become the default scanner.

Binary to source file mapping

Tooling

Binaries are built from many different source files, but the exact list of them depends on build options. For this reason a reliable mechanism needs to be put in place to extract this list after the build process in order to determine the license information.

Compilers store information in the binaries it outputs, that can be used by a debugger to pause execution of a process at a point corresponding to a selected line of source code. This information provides a mapping between the lines of source code and the compiled machine code operations. Executable binaries in Linux are generally stored in the [Executable and Linkable Format](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format)¹¹ (ELF), the associated [DWARF](https://en.wikipedia.org/wiki/DWARF)¹² debugging data format is generally used to store this debugging information inside the ELF in specific “debug” sections.

The tool `dwarf2sources` parses this information and extracts the name of the source files that were used to generate each binary, generating a `json` file that can easily be parsed later. Combining this with the licensing information provided

¹⁰<https://dep-team.pages.debian.net/deps/dep5/>

¹¹https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

¹²<https://en.wikipedia.org/wiki/DWARF>

146 in the licensing report, a mapping can be made between each binary and it's
147 associated licenses.

148 CI Pipeline integration

149 Apertis uses the Open Build Service (OBS) platform to build the binary pack-
150 ages in a controlled manner across several architectures and releases. OBS uti-
151 lizes `dpkg-buildpackage` behind the scenes to build each package. This utility has
152 access to the source licensing report as it is contained in the Debian metadata
153 archive. As well as the source licensing, the Debian metadata archive contains
154 configuration to help `dpkg-buildpackage` determine how to build the source. This
155 is typically done with the help of `debhelper`¹³, which provides helpers that sim-
156 plify this process.

157 Apertis extended `debhelper` by including a new command `dh_setup_copyright` to
158 perform the source file name extraction using `dwarf2sources` as described above,
159 as well as copy in any copyright reports coming from source files that are part
160 of external packages. Typically the binaries are stripped (using a `debhelper`
161 command called `dh_strip`) prior to packaging, removing the debug symbols from
162 the binary and reducing its size. For this reason `dh_setup_copyright` is placed
163 before this step in the `dh` sequence. Whilst the debug symbols are kept, packaged
164 separately in the `dbgsym` package, it's easier to perform the mapping before this
165 is done. All the results from this command are stored in the binary package
166 under `/usr/share/doc/<package>/`.

167 Following this same idea, Apertis also extends `debhelper` the command
168 `dh_installdocs` to install the license report generated by the scanner under
169 `/usr/share/doc/<package>/copyright_report`.

170 Despite that, this solution should work for most packages. Some packages might
171 instead need special handling, for instance because they are not using `debhelper`.
172 An example of that is the `linux` kernel package. These special cases will be
173 covered with further improvements.

174 As these reports are provided by each binary package, the reports from installed
175 packages can be accessed at image build time and amalgamated into an image
176 wide report at that point should it be required. As a binary can be built from
177 multiple sources, each with differing licenses, it is necessary for the report to
178 detail each file that is used to create each binary and the licensing under which
179 it is provided. In some circumstances dual licensed source code may allow for
180 a binary to be effectively licensed under the terms of a single license, that is
181 the user has the option to pick a license that results in the whole binary being
182 able to be provided under the terms of a single license. Where dual licensed
183 source code isn't used, the terms of all applicable licenses should be declared.
184 The terms of the various licenses may be considered `compatible`¹⁴, allowing the

¹³<https://manpages.debian.org/jessie/debhelper/debhelper.7.en.html>

¹⁴https://en.wikipedia.org/wiki/License_compatibility

185 binary to effectively be managed under the terms of the more restrictive license.
186 For example, a binary derived from source code licensed with the GPLv2 license
187 and other source code licensed with the MIT license, the terms of both apply to
188 the binary, though as the terms of the MIT license will be met if the binary is
189 used in accordance with the terms of the GPLv2, then handling the binary as
190 though it was licensed under the GPLv2 will ensure the terms of both are met.
191 Not all possible combinations of licenses work out this way and thus why it is
192 important to ensure that licensing is properly tracked.

193 Binary Licensing Reporting

194 Tooling

195 The approach each project using Apertis takes with regards to the reporting of
196 licensing information should be driven by how this information is to be utilized,
197 i.e. some projects may wish to parse the license information and present it in a
198 single BOM file in HTML, XML or human readable text.

199 For the images provided by the Apertis project, the script `generate_bom.py` com-
200 bines the reports saved in `/usr/share/doc/<package>/`, using the binary-to-sources
201 JSON mappings and the external package copyright information, into a single
202 `json` file which is provided with the image. This file can be generated with dif-
203 ferent levels of verbosity allowing to list licenses per image, package, binary or
204 source file.

205 This same scripts also issues a warning in case a problematic license is found.

206 CI Pipeline integration

207 Apertis utilizes [Debos](https://github.com/go-debos/debos)¹⁵ in its image generation pipeline, which provides a very
208 versatile way of customizing them. During the final stage of the image cre-
209 ation, the script `generate_bom.py` is used to build the BOM file with the license
210 information of the image and export it as an additional artifact. Finally as
211 both `fixedfunction` and `hmi` images should not ship extra data, the contents of
212 `/usr/share/doc/` are dropped from the image.

213 Step-by-step process

214 This is a description of the steps in the process as currently implemented:

215 The following step-by-step process is followed for all the packages, however it
216 is only valid for packages that use standard `dh` rules and build binaries. Other
217 packages only provide copyright information which currently is not included in
218 BOM file.

¹⁵<https://github.com/go-debos/debos>

219 During package source build on Gitlab CI pipelines

- 220 1. when a package is imported from Debian to Apertis the `scan-license` job in
221 the packaging pipeline¹⁶ will call `ci-license-scan`¹⁷ to submit the sources
222 to the scanner, be it `scan-copyrights`, FOSSology or any other tool
- 223 2. metadata in `debian/apertis/copyright.yml`¹⁸ can be used to override things
224 where the scanner gives the wrong results, which would no longer be
225 needed if using FOSSology for example, where the correct licensing in-
226 formation would be stored in its database
- 227 3. the output is committed in the `debian/apertis/copyright` file in the
228 sources¹⁹
- 229 4. if some files have problematic licenses but they do not really affect us for
230 any reason, the reason is documented in `debian/apertis/copyright.whitelist`²⁰
- 231 5. for packages meant to be installed on production devices, the packaging
232 pipeline will fail if problematic licenses are detected and the affected files
233 are not whitelisted

234 During package build on OBS

- 235 1. when the sources are submitted to OBS, during the build the
236 `dh_setup_copyright` subcommand for `Debhelper`²¹ calls the `dwarf2sources`
237 tool²² to generate a mapping from binaries to the source files used to
238 build them and determine if any of those source files came from external
239 packages
- 240 2. the output is included in the same `.deb` file as the processed li-
241 brary/executable:
 - 242 • `/usr/share/doc/$packagename/$packagename_bin2sources_$packagearch.json`,
243 containing the mapping from binaries to source files
- 244 • `/usr/share/doc/$packagename/external_copyrights/`, a directory con-
245 taining all the copyrights of packages whose source files were directly246 embedded into this package's binaries- 247 • `/usr/share/doc/$packagename/$packagename_metadata_$packagearch.json`,
248 containing any other metadata related to copyrights (at the moment,249 this maps source files from external packages to the package names250 that provided them)

¹⁶<https://gitlab.apertis.org/infrastructure/ci-package-builder/-/blob/master/ci-package-builder.yml>

¹⁷<https://gitlab.apertis.org/infrastructure/apertis-docker-images/-/blob/apertis/v2023dev2/package-source-builder/overlay/usr/bin/ci-license-scan>

¹⁸<https://gitlab.apertis.org/pkg/gnutls28/-/blob/apertis/v2023dev2/debian/apertis/copyright.yml>

¹⁹<https://gitlab.apertis.org/pkg/gnutls28/-/blob/apertis/v2023dev2/debian/apertis/copyright>

²⁰<https://gitlab.apertis.org/pkg/gnutls28/-/blob/apertis/v2023dev2/debian/apertis/copyright.whitelist>

²¹https://gitlab.apertis.org/pkg/debhelper/-/blob/apertis/v2023dev2/dh_setup_copyright

²²<https://gitlab.apertis.org/pkg/dwarf2sources/>

- 251 3. during the same build on OBS, a custom hook in the `dh_installdocs`
252 step stores the `debian/apertis/copyright` sourcefile-to-licenses mapping as
253 `/usr/share/doc/$packagename/copyright_report.gz` in the binary `.deb` pack-
254 ages, to make it available when the packages get installed
- 255 4. for each installed `.deb` package, `/usr/share/doc/$packagename/$packagename_bin2sources_$packagearch.json`,
256 `/usr/share/doc/$packagename/$packagename_metadata_$packagearch.json`,
257 `/usr/share/doc/$packagename/external_copyrights/`, and `/usr/share/doc/$packagename/copyright_report.gz`
258 get unpacked during image generation

259 During image generation on Gitlab CI pipelines

- 260 1. the `generate_bom.py` script²³ is invoked at the end of each image recipe²⁴,
261 loading all the `/usr/share/doc/$packagename/$packagename_bin2sources_$packagearch.json`
262 binary-to-sourcefiles mappings, `/usr/share/doc/$packagename/copyright_report.gz`
263 sourcefile-to-licenses mappings, and `/usr/share/doc/$packagename/external_copyrights`
264 external package copyrights to combine them and produce a JSON `.licenses`
265 `report`²⁵ with the binary-to-licenses mapping to match each library
266 and executable shipped in the image to the licenses of the sources used to
267 build them
- 268 2. the `check_bom.py` script²⁶ is invoked afterwards to ensure that the license
269 files conform to the `Apertis license expectations`²⁷
- 270 3. human-readable reports in any format can be generated by the JSON data
271 describing the licenses that apply to the libraries and executables shipped
272 in the image itself

²³https://gitlab.apertis.org/infrastructure/apertis-image-recipes/-/blob/apertis/v2023dev2/scripts/generate_bom.py

²⁴<https://gitlab.apertis.org/infrastructure/apertis-image-recipes/-/blob/apertis/v2023dev2/image-uboot.yaml>

²⁵https://images.apertis.org/release/v2023dev2/v2023dev2.0/arm64/fixedfunction/apertis_v2023dev2-fixedfunction-arm64-rpi64_v2023dev2.0.img.licenses.gz

²⁶https://gitlab.apertis.org/infrastructure/apertis-image-recipes/-/blob/apertis/v2023dev2/scripts/check_bom.py

²⁷<https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/>