Scancode evaluation

Contents

Scancode installation **3**

Scancode output format **3**

Select Apertis packages to evaluate scancode **4**

Run scan-copyrights as gold standard **4**

Run scancode **5**

Analysis time **6**
Analysis time with –processes from 1 to 8 7
Analysis time with –timeout X 8
Analysis time with –max-in-memory 0 9
Analysis time with ONLY –license 11

Reliability of detected license **11**

No license deduction for project and folder **12**
Statistic about files without detected license 13

DEP5 invalid format **14**

scancode output format **14**

Summary **15**
Required resources for analysis 15
License scan accuracy 15
Output format 15
Outdated debian/apertis/copyright.yaml 15

Proposed plan **16**

References **17**

Currently, scan-copyrights[1] (which uses licensecheck[2] under the hood) is used in Apertis to scan copyright/license notices. This tool has some downsides, thus we are evaluating to use scancode-toolkit[3] instead. A comparison of `licensecheck` vs `scancode` is available on the ScanCode's website[4], TL;DR: *scancode is more accurate but slower*.

[1] https://tracker.debian.org/pkg/libconfig-model-dpkg-perl
[2] https://tracker.debian.org/pkg/licensecheck
[3] https://github.com/nexB/scancode-toolkit
[4] https://scancode-toolkit.readthedocs.io/en/stable/misc/faq.html#how-is-scancode-different-from-debian-licensecheck

`scancode-toolkit` has an option to export results as DEP5 format[5] (see GH#472[6]) which is the format currently use by Apertis license tooling. That means, `scancode-toolkit` is potentially compatible with the rest of the Apertis licensing tooling.

Scancode installation

`scancode` is not available as Debian package (GH#1580[7] and GH#3253[8]) nor as a Docker image (GH#3026[9]), but a Dockerfile[10] is provided by upstream[11]. That means, we can create our own Docker image, or we can reuse the OSS Review Toolkit Docker image which integrates `scancode`. Since the ORT Docker image used in our pipeline is outdated, it would be easier for now to decouple scancode from the ORT docker image to avoid having to use an outdated scancode (scancode used in the ORT image is one year old[12].

Here are the steps to build a docker image:

```
git clone https://github.com/nexB/scancode-toolkit
cd scancode-toolkit
LATEST_VER=v32.0.8
git checkout $LATEST_VER
docker build --tag scancode-toolkit --tag scancode-toolkit:$LATEST_VER .
```

Scancode output format

Scancode is able to write its output in different formats:

```
docker run  scancode-toolkit --help
...
  output formats:
    --json FILE             Write scan output as compact JSON to FILE.
    --json-pp FILE          Write scan output as pretty-printed JSON to FILE.
    --json-lines FILE       Write scan output as JSON Lines to FILE.
    --yaml FILE             Write scan output as YAML to FILE.
    --csv FILE              [DEPRECATED] Write scan output as CSV to FILE. The
                            --csv option is deprecated and will be replaced by
```

[5] https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
[6] https://github.com/nexB/scancode-toolkit/issues/472
[7] https://github.com/nexB/scancode-toolkit/issues/1580
[8] https://github.com/nexB/scancode-toolkit/issues/3253
[9] https://github.com/nexB/scancode-toolkit/issues/3026
[10] https://github.com/nexB/scancode-toolkit/blob/develop/Dockerfile
[11] https://scancode-toolkit.readthedocs.io/en/latest/getting-started/install.html#installation-via-docker
[12] https://github.com/nexB/scancode-toolkit/releases/tag/v31.2.4

```
                              new CSV and tabular output formats in the next
                              ScanCode release. Visit
                            https://github.com/nexB/scancode-toolkit/issues/3043
                              to provide inputs and feedback.
      --html FILE             Write scan output as HTML to FILE.
     --custom-output FILE    Write scan output to FILE formatted with the custom
                              Jinja template file.
      --debian FILE           Write scan output in machine-readable Debian
                              copyright format to FILE.
     --custom-template FILE  Use this Jinja template FILE as a custom template.
     --cyclonedx FILE        Write scan output in CycloneDX JSON format to FILE.
     --cyclonedx-xml FILE    Write scan output in CycloneDX XML format to FILE.
      --spdx-rdf FILE         Write scan output as SPDX RDF to FILE.
      --spdx-tv FILE          Write scan output as SPDX Tag/Value to FILE.
...
```

These formats include:

- Debian DEP5 format, the one already in use with Apertis licensing tooling.
- YAML, widely used in Apertis and used to teach `scan-copyrights` detection license issues (i.e. `debian/apertis/copyright.yml`).
- SPDX, open standard for communicating SBOM information.
- CycloneDX, another SBOM standard.

Initially, it should be simpler to continue using the Debian DEP5 since the whole Apertis licensing tooling is using it. But for a long term plan, we may want to switch to a more widely used format like SPDX or CycloneDX which are also compatible with ORT and other tools. This should make the Apertis license/SBOM processes more flexible.

Select Apertis packages to evaluate scancode

Let's use packages that are wrongly detected by `scan-copyrights` (i.e. packages with the use of `override-license` in `debian/apertis/copyright.yml`.

Here is a small random list of packages based on a local grep of `override-license`: debianutils, libarchive, libgdata, libunistring, libusb, nss, nss-pem, openjpeg2, openssl xorg-server.

Run scan-copyrights as gold standard

First, `scan-copyrights` is run on the package in a v2025dev2 VM:

```
# Because of the use of "override-license" in debian/apertis/copyright.yml
LIST_PKGS=" debianutils libarchive libgdata libunistring libusb nss nss-pem openjpeg2 openssl xorg-ser
# Adding other well known pkgs
LIST_PKGS+=" pipewire rust-coreutils "
for PKG in $LIST_PKGS:
do
  git clone https://gitlab.apertis.org/pkg/${PKG}.git
  cd ${PKG}
  /usr/bin/time -f "%e" scan-copyrights > ../${PKG}-scan-copyright 2> ../${PKG}-time
  cd ..
done
```

Run scancode

Now, `scancode` is run by excluding `debian/copyright` and `debian/apertis/copyright` because they can easily confuse `scancode` (see GH#2885[13]).

[13] https://github.com/nexB/scancode-toolkit/issues/2885#issuecomment-1136268172

```
# Because of the use of "override-license" in debian/apertis/copyright.yml
LIST_PKGS=" debianutils libarchive libgdata libunistring libusb nss nss-pem openjpeg2 openssl xorg-ser
# Adding other well known pkgs
LIST_PKGS+=" pipewire rust-coreutils firefox-esr"
for PKG in $LIST_PKGS:
do
  git clone https://gitlab.apertis.org/pkg/${PKG}.git
  cd ${PKG}

  # DEP5 output
  docker run -v $PWD/:/project scancode-toolkit \
      --copyright --license --license-text --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/copyright \
      --ignore */debian/apertis/${PKG}-scancode-copyright \
      -n 8 --debian /project/debian/apertis/${PKG}-scancode-copyright \
      /project/.

  # YAML output
  docker run -v $PWD/:/project scancode-toolkit \
      --copyright --license --license-text --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/copyright \
      --ignore */debian/apertis/${PKG}-scancode-copyright \
      -n 8 \
      --yaml /project/debian/apertis/${PKG}-scancode-copyright-yaml \
      /project/.

  cd ..
done
```

Analysis time

This analysis was performed on a XPS13-9310 laptop with a CPU i7-1185G7 (@3.00GHz×8), 16 GB RAM and an SSD hard disk.

Package	Time scan-copyrights	Time scancode	Diff
debianutils	1.3 s	38 s	~29 times slower
libarchive	6.6 s	7 m 25 s	~67 times slower
libgdata	4.7 s	3 m 55 s	~50 times slower
libunistring	8.1 s	10 m 48 s	~80 times slower
libusb	1.2 s	54 s	~45 times slower
nss	25.8 s	28 m 47 s	~67 times slower
nss-pem	28.6 s	26 m 59 s	~56 times slower
openjpeg2	3.7 s	3 m 8 s	~50 times slower

Package	Time scan-copyrights	Time scancode	Diff
openssl	23.6 s	18 m 3 s	~46 times slower
pipewire	6.1 s	4 m 57 s	~48 times slower
rust-coreutils	4.4 s	1 m 54 s	~26 times slower
xorg-server	9.1 s	9 m 24 s	~62 times slower
firefox-esr*	XX s	OOM killed after ~ 1 d [1]	~XX times slower

- `firefox-esr` is one of the biggest packages in Apertis, but is not in the `target` repository. Thus, we wouldn't have to analyze it with `scancode`, but it is used here to evaluate scancode in the worst cases.
- [1] scancode ran on `firefox-esr` for ~ 23 hours and 30 mins before being OOM killed. It seems, the scan was over and scancode was processing data to generate its output file when it was killed. Its scanning parallel processes have stopped, only the main process was running and the used RAM was at ~ 3 GB (of 16 GB available) about 10 mins before OOM.
- While doing the analysis with 8 parallel processes, all of them were at 100% during the entire analysis time, so at least the CPU is a bottleneck.

Analysis time with –processes from 1 to 8

From the scancode options[14]:

```
-n, --processes INTEGER

    Scan <input> using n parallel processes. [Default: 1]
```

This option allows to use several processes for scanning files.

```
PKG="debianutils"
git clone https://gitlab.apertis.org/pkg/${PKG}.git
cd ${PKG}

for N in {1..8}
do
  # YAML output
  docker run -v $PWD/:/project scancode-toolkit \
      --copyright --license --license-text --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/* \
      -n ${N} \
      --yaml /project/debian/apertis/${PKG}-${N}-scancode-copyright-yaml \
      /project/.
done
```

[14] https://scancode-toolkit.readthedocs.io/en/stable/cli-reference/core-options.html

N processes	Time scancode —debianutils
1	1 m 34.5 s
2	58.2 s
3	45.4 s
4	39.0 s
5	38.1 s
6	37.2 s
7	36.7 s
8	36.0 s

Adding more parallel processes improve the scanning time, but it seems we are reaching a threshold at ~ 4 parallel processes where adding more processes only slightly improves the scanning time. This may be due to the fact that the tested package is *small*. For a bigger package like `firefox-esr`, this threshold may be higher and it could be beneficial to have more parallel processes.

Analysis time with –timeout X

From the scancode options[15]:

```
--timeout FLOAT

  Stop scanning a file if scanning takes longer than a timeout in seconds. [Default: 120]
```

This option allows to avoid getting stuck on a file for too long.

```
PKG="debianutils"
git clone https://gitlab.apertis.org/pkg/${PKG}.git
cd ${PKG}

for N in 120 110 100 90 80 70 60 30 10
do
  # YAML output
  docker run -v $PWD/:/project scancode-toolkit \
      --copyright --license --license-text --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/* \
      --timeout ${N} -n 8 \
      --yaml /project/debian/apertis/${PKG}-${N}-scancode-copyright-yaml \
      /project/.
done
```

[15] https://scancode-toolkit.readthedocs.io/en/stable/cli-reference/core-options.html

Timeout	Time scancode -n 8 –debianutils
120 (default)	38.5 s
110	43.6 s
100	44.1 s
90	39.9 s
80	40.3 s
70	38.3 s
60	37.8 s
30	37.9 s
10	29.6 s

Decreasing the timeout per file seems to be quite efficient to reduce the scanning time, but since some files are no longer fully scanned, a more comprehensive comparison of detected licenses should be done to ensure we are not losing too much data.

```
PKG="firefox-esr"
git clone https://gitlab.apertis.org/pkg/${PKG}.git
cd ${PKG}

date
docker run -v $PWD/:/project scancode-toolkit \
   --copyright --license --license-text --strip-root \
   --ignore */debian/copyright --ignore */debian/apertis/* \
   --timeout 10 -n 8 \
   --yaml /project/debian/apertis/${PKG}-scancode-timeout-10-copyright-yaml \
   /project/.
date
```

Interestingly, the `--timeout 10` option on the `firefox-esr` scan avoids triggering the OOM kill issue, probably because more files are no longer scanned, thus `scancode` has to manage a smaller dataset and therefore a smaller memory footprint. Unfortunately, this option is not enough to make scanning time acceptable since it took 35 hours to complete the full scan of `firefox-esr`.

Analysis time with –max-in-memory 0

From the scancode options[16]:

```
--max-in-memory INTEGER

   Maximum number of files and directories scan details kept in memory during a
    scan. Additional files and directories scan details above this number are
```

[16] https://scancode-toolkit.readthedocs.io/en/stable/cli-reference/core-options.html

```
    cached on-disk rather than in memory. Use 0 to use unlimited memory and
  disable on-disk caching. Use -1 to use only on-disk caching. [Default: 10000]
```

Based on an upstream issue (see GH#1014[17]), the disk cache seems to be really slow.

```
# Because of the use of "override-license" in debian/apertis/copyright.yml
LIST_PKGS=" debianutils libarchive libgdata libunistring libusb nss nss-pem openjpeg2 openssl xorg-ser
# Adding other well known pkgs
LIST_PKGS+=" pipewire rust-coreutils firefox-esr"
for PKG in $LIST_PKGS:
do
  git clone https://gitlab.apertis.org/pkg/${PKG}.git
  cd ${PKG}
  # YAML output
  docker run -v $PWD/:/project scancode-toolkit \
      --copyright --license --license-text --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/* \
      --max-in-memory 0 -n 8 \
      --yaml /project/debian/apertis/${PKG}-scancode-copyright-yaml \
      /project/.
  cd ..
done
```

Package	Time scan-copyrights	Time scancode	Diff
debianutils	1.3 s	34 s	~26 times slower
libarchive	6.6 s	6 m 41 s	~60 times slower
libgdata	4.7 s	3 m 35 s	~46 times slower
libunistring	8.1 s	11 m 12 s	~83 times slower
libusb	1.2 s	54 s	~45 times slower
nss	25.8 s	27 m 47 s	~66 times slower
nss-pem	28.6 s	26 m 49 s	~57 times slower
openjpeg2	3.7 s	3 m 14 s	~52 times slower
openssl	23.6 s	18 m	~46 times slower
pipewire	6.1 s	5 m 29 s	~54 times slower
rust-coreutils	4.4 s	2 m 3 s	~28 times slower
xorg-server	9.1 s	10 m 9 s	~67 times slower
firefox-esr*	XXX s	OOM killed after ~ 1 d [1]	~XX times slower

Passing `--max-in-memory 0` to `scancode` doesn't improve scanning time since these results are of the same order of magnitude (+/- random fluctuation) to the ones without this option.

[17] https://github.com/nexB/scancode-toolkit/issues/1014

Analysis time with ONLY –license

i.e. without `--copyright --license-text`

```
PKG="nss"
git clone https://gitlab.apertis.org/pkg/${PKG}.git
cd ${PKG}

# YAML output
docker run -v $PWD/:/project scancode-toolkit \
   --license --strip-root \
   --ignore */debian/copyright --ignore */debian/apertis/* \
   -n 8 \
   --yaml /project/debian/apertis/${PKG}-scancode-ONLYlicense-copyright-yaml \
   /project/.
```

Package	Time scan-copyrights	Time scancode with copyright	Time scancode without copyright	Impro
nss	25.8 s	27 m 47 s	21 m 42 s	22%

Avoiding the scan for copyrights and only checking licenses slightly decreases the scanning time but it remains in the same order of magnitude without significantly improving the comparison with `scan-copyright`.

Reliability of detected license

Some of debian/apertis/copyright.yaml files used are no longer required since the Bookworm rebase, so all packages analyzed don't have a problematic file which can be used to compare `scan-copyrights` and `scancode`.

Package	File	Actual license
libarchive	shar.1[18]	BSD-4-Clause-UC
libgdata	README[19]	LGPL-2.1-or-later
libunistring	version.c[20]	LGPL-3.0-or-later OR GPL-2.0-or-later
libusb	06_bsd.diff[21]	BSD-2-Clause [1]
nss	derdump.1[22]	MPL-2.0 [2]
nss-pem	doc/rst/legacy/*[23]	MPL-1.1 OR GPL-2.0-only OR LGPL-2.1-only
openjpeg2	opj_getopt.c[24]	BSD-3-Clause
openssl	cmll-x86*.pl[25]	Apache-2.0 OR GPL-2.0-or-later OR LGPL-2.1-or-later OR MPL-
xorg-server	hw/xwin/winprefsyacc.*[26]	GPL-3.0-or-later WITH Bison-exception-2.2 AND LicenseRef-scan

[18] https://gitlab.apertis.org/pkg/libarchive/-/blob/13653b9a562c658133b37e5859693898a7b929e8/contrib/shar/shar.1

- [0] scan-copyrights has improved in Bookworm.
- [1] Retrospective change: https://www.netbsd.org/about/redistribution.html#why2clause
- [1.1] BSD-2-Clause-NetBSD and/or BSD-2-clause and/or BSD-3-clause and/or FSFUL and/or FSFULLR and/or GPL-2 and/or LGPL-2 and/or X11
- [2] Simplified upstream with bookworm, debian/apertis/copyright.yaml outdated.
- [3] A second license is later in the code

`scancode` is better to deal with complex licenses combinations, especially because it scans the whole file and not only the first lines. Moreover, it reports all licenses detected with a matching score (available in the YAML output).

No license deduction for project and folder

While `scan-copyrights` is able to perform some deduction of license/copyright for a project and/or folder, `scancode` only performs scanning at file level.

For instance, `scan-copyrights` gives the following result for `openssl`:

```
Files: *
Copyright: 1998-2023, The OpenSSL Project
 1995-1998, Eric A. Young, Tim J. Hudson
License: Apache-2.0

...

Files: crypto/ec/asm/*
Copyright: 1998-2023, The OpenSSL Project Authors.
License: Apache-2.0 and/or OpenSSL
```

This result give us the information that the project is under the license `Apache-2.0` and files in `crypto/ec/asm/*` are under `Apache-2.0 and/or OpenSSL` licenses.

[19] https://gitlab.apertis.org/pkg/libgdata/-/blob/6e88c1a214f86c3025b0124a97b557c3710df339/README

[20] https://gitlab.apertis.org/pkg/libunistring/-/blob/d4765b859b6ca17f0ea885d3ec9adea691dc1133/lib/version.c

[21] https://gitlab.apertis.org/pkg/libusb/-/blob/5890eb277e80734cea681e58c52712a06516d85b/debian/patches/06_bsd.diff

[22] https://gitlab.apertis.org/pkg/nss/-/blob/4a93da238116e085ad1d949b650e9ec32d98038a/nss/doc/nroff/derdump.1

[23] https://gitlab.apertis.org/pkg/nss-pem/-/blob/65192a7e89364a15fadb7b2c832a72a0efa16e63/nss/nss/doc/rst/legacy/nss_3.12.2_release_notes.html/index.rst

[24] https://gitlab.apertis.org/pkg/openjpeg2/-/blob/a1d3412b2721a1d7dbd7cd3d7f13bd6647bda7f5/src/bin/common/opj_getopt.c

[25] https://gitlab.apertis.org/pkg/openssl/-/blob/f603f6ae103b26ef89089fc63945e833c0401839/crypto/camellia/asm/cmll-x86.pl

[26] https://gitlab.apertis.org/pkg/xorg-server/-/blob/2d6db144d7c3393d2f3f113168b4f421192c90c8/hw/xwin/winprefsyacc.c

This behavior allows to assign a license to files by inheriting it from the license of the project (or from the higher level folder's license).

Some projects don't add license/copyright information in all of their files, which could be annoying for `scancode` as it won't be able to detect the right license. We would need to add this logic in `scancode` or in another Apertis script (like ci-license-scan[27]?).

Some related upstream issues:

- Proposal: Scan deduction and summarization[28]
- Primary license detections not shown properly in debian_copyright[29]

Statistic about files without detected license

The yaml file generated by scancode is sometimes malformed due to `--license-text` (see #GH3219[30], but seems not enough).

```
# Because of the use of "override-license" in debian/apertis/copyright.yml
LIST_PKGS=" debianutils libarchive libgdata libunistring libusb nss nss-pem openjpeg2 openssl xorg-ser
# Adding other well known pkgs
LIST_PKGS+=" pipewire rust-coreutils firefox-esr"
for PKG in $LIST_PKGS:
do
  git clone https://gitlab.apertis.org/pkg/${PKG}.git
  cd ${PKG}

  # YAML output
  docker run -v $PWD/:/project scancode-toolkit \
      --license --strip-root \
      --ignore */debian/copyright --ignore */debian/apertis/copyright \
      --ignore */debian/apertis/${PKG}-scancode-copyright \
      -n 8 \
      --yaml /project/debian/apertis/${PKG}-scancode-copyright-yaml \
      /project/.

  cd ..
done
```

[27] https://gitlab.apertis.org/infrastructure/apertis-docker-images/-/blob/apertis/v2025dev2/package-source-builder/overlay/usr/bin/ci-license-scan
[28] https://github.com/nexB/scancode-toolkit/issues/377
[29] https://github.com/nexB/scancode-toolkit/issues/3424
[30] https://github.com/nexB/scancode-toolkit/issues/3219

Package	Files number	Files number with license	Detected license %
debianutils	134	44	32.8 %
libarchive	1420	716	51.9 %
libgdata	705	326	46.2 %
libunistring	2118	1961	92.5 %
libusb	96	30	31.2 %
nss	4531	2393	52.8 %
nss-pem	4574	2423	52.9 %
openjpeg2	478	334	69.8 %
openssl	4655	3349	72 %
pipewire	1251	952	76 %
rust-coreutils	1311	326	24.8 %
xorg-server	1791	1227	68.5 %
firefox-esr*	XXX	XXX	XXX

DEP5 invalid format

`scancode` generates malformed `files stanza`. As defined in the debian/copyright specification[31], each `files stanza` is composed by mandatory fields (i.e. `Files`, `Copyright` and `License`) and one optional field (i.e. `Comment`). For instance:

```
Files: Xext/sleepuntil.h
Copyright: 1993-2003, The XFree86 Project, Inc.
License: Expat
```

When `scancode` is not able to define a copyright or a license for a file, then it creates a stanza with only the `Files` field whereas `scan-copyrights` fills the missing field with `UNKNOWN`. Here is an example of malformed stanza by `scancode`:

```
Files: CODE_OF_CONDUCT.md
```

Here is another example from `scan-copyrights` where a missing field is filled:

```
Files: xkb/Makefile.in
Copyright: 1994-2021, Free Software Foundation, Inc.
License: UNKNOWN
```

This issue should easy be fixable in `scancode`, by filing missing field with an `UNKNOWN` value.

scancode output format

Support of `DEP5` format is incomplete in `scancode`, but bigger issues can probably be easily fixed.

[31] https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#files-stanza

YAML format gives way more information like: lines of the detected licenses, the pattern, a score of matching, several identifiers of the licenses detected, etc. Having all of these information may be useful for future enhancement of Apertis license tooling.

Summary

Their different approaches in file analysis explains why scancode is slower, but is able to detect way more licenses than scan-copyrights (see upstream comparison[32]):

- licensecheck is "a Perl script using hand-crafted regex patterns to find typical copyright statements and about 50 common licenses";
- scancode's detection "is based on a (large) number of license full texts (~2100) and license notices, mentions and variants (~32,000) and is data-driven as opposed to regex-driven. It detects and reports exactly where license text is found in a file. Just throw in more license texts to improve the detection."

Required resources for analysis

scancode is ~50 times slower than scan-copyrights. scancode requires much more RAM than scan-copyrights.

License scan accuracy

- scan-copyrights has improved between Bullseye and Bookworm.
- scancode has a better detection for complex cases.

Output format

- DEP5 incomplete support, but already used by apertis license tooling.
- YAML seems a sensible alternative since it provides many more information, but would require to adapt apertis license tooling to this new format.

Outdated debian/apertis/copyright.yaml

This file would need to be refreshed in Apertis packages since the Bookworm rebase. scan-copyrights is smarter and some packages have fixed their licensing issues.

[32] https://scancode-toolkit.readthedocs.io/en/stable/misc/faq.html#how-is-scancode-different-from-debian-licensecheck

Proposed plan

Some general guidelines:

- We need to come with a progressive approach, this is not something that will happen from one day to the other
- Most of the packages are small and should take a reasonable amount of time to scan
- We should be able to selectively disable scancode when necessary
- We can add additional logic to only scan the files that have changed since last scan

Proposed plan to use scancode instead of scan-copyrights:

1. Update the docker image used to generate ORT reports in order to reuse it for scancode. Apertis uses a handcrafted docker image to generate ORT reports, since this image already contains scancode, it's possible to reuse it to run scancode. The first step is to switch to an up-to-date image provided by ORT[33]. This step will requite to adjust some scripts use by Apertis including the template used to generate ORT reports.
2. Fix the DEP5 format created by scancode by adding missing mandatory fields. Scancode generates report in the DEP5 format[34], unfortunately, mandatory fields are missing when the copyright/license is not detected (see GH#3714[35]). Instead, scancode should fill missing field with `UNKNOWN` or `no-info-found` as done by `scan-copyrights`.
3. Add support of "license deduction for project and folder"to scancode. scancode is unable to deduce a license for a whole project/folder based on the license of other files. Without this feature, ~ 50% files will have a missing license which is a regression compared to scan-copyrights. This task consists in adding a logic to scancode to deduce a license for a folder and/or project.
4. Add a new job running `scancode` to the `ci-package-builder` pipeline in parallel to the current `scan-licenses` job using `scan-copyrights`.
5. Generate a new `scancode` report for all packages in `target` using the job added in the previous step.
6. In the SBOM logic, add preference to use the `scancode` report if available otherwise use the one from `scan-copyrights`.

Some other tasks can be done in parallel:

- Investigate how to use caching to avoid scanning files already scanned in a previous run.
- Investigate how to improve performance of scancode (speed and RAM usage).

[33] https://github.com/oss-review-toolkit/ort/pkgs/container/ort
[34] https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
[35] https://github.com/nexB/scancode-toolkit/issues/3714

References

- scancode-toolkit[36]

[36] https://github.com/nexB/scancode-toolkit