Understanding and Auditing the Licensing of Open Source Software Distributions

Daniel M. German†, Massimiliano Di Penta‡, Julius Davies†
† Dept. of Computer Science, University of Victoria, Canada
‡ Dept of Engineering, University of Sannio, Italy
dmg@uvic.ca, dipenta@unisannio.it, juliusd@uvic.ca

***Abstract*—Free and open source software (FOSS) is often distributed in binary packages, sometimes part of GNU/Linux operating system distributions, or part of products distributed/sold to users.**

FOSS creates great opportunities for users, developers and integrators, however it is important for them to understand the licensing requirements of any package they use. Determining the license of a package and assessing whether it depends on other software with incompatible licenses is not trivial. Although this task has been done in a labor intensive manner by software distributions, automatic tools to perform this analysis are highly desired.

This paper proposes a method to understand licensing compatibility issues in software packages, and reports an empirical study aimed at auditing licensing issues in binary packages of the Fedora-12 GNU/Linux distribution. The objective of this study is (i) to understand how the license declared in packages is consistent with those of source code files, and (ii) to audit the licensing information of Fedora-12, highlighting cases of incompatibilities between dependent packages.

The obtained results—supported by feedback received from Fedora contributors—show that there exist many nuances in determining the license of a binary package from its source code, as well as cases of license incompatibility issues due to package dependencies.

Keywords: Software licenses, evolution, mining software repositories, open source systems, empirical study.

I. Introduction

The advent of free and open source software (FOSS) has created a large ecosystem of applications, libraries and components that are readily available for download and usage. Often these applications/libraries/components are distributed together with the operating system, and this happens for the various distributions of the GNU/Linux operating system. The intellectual property of FOSS is protected by licensing mechanisms and copyright notices that determine how an open source can be used, even impacting the architecture of a system [1].

Dealing with issues related to open source licensing is not trivial, as at the time of writing there are 65 open source licenses approved by the Open Source Initiative, and many more in use; each of them imposing particular constraints concerning how one can use and/or change a software. License auditing is usually a manual analysis (assisted by custom scripts) performed by GNU/Linux distributions (such as Debian, OpenSuse, RedHat, and Ubuntu).

When one installs a new application/library in a Unix system, this is often done from what is known as a binary package (such as *RPM* packages in Fedora/Redhat-like distributions or *.deb* packages in Debian-like distributions). Other than the various artifacts composing the application/library, the package also contains metadata describing, among other things, (i) under what open source license the package is distributed (which we call the *declared license of the binary package)*, and (ii) the list of other packages required in order to successfully install and use the current package (its *required packages*) [2].

From a legal point of view, modifying and redistributing a FOSS package poses two important issues:

1) *Can we trust the declared license of the package? i.e.*, is that license consistent with those of the files the package contains?
2) *Do the dependency requirements of a binary package create potential legal concerns?* Software with different licenses can be combined to create larger systems, but such combinations can increase the chance for license incompatibilities.

In this paper we describe a method to help understand licensing issues that can arise from changing, combining, and re-distributing packages in open source distributions. First, we identify the license of all files contained in the source code from which the binary package is created (the source package). This gives us a detailed overview of licenses present in the source code of a package, and allows us to identify possible inconsistencies with the declared license of the binary package. Second, we combine the dependency graph of a binary package with the declared licenses of its dependencies, trying to identify any license inconsistencies.

We have carried out a large empirical study aimed at analyzing licensing issues in the entire Linux-based Fedora-12 operating system. Results indicated the presence of some inconsistencies between the declared licenses and the source code ones, as well as problems arising because of dependencies between different packages. For many of these potential problems, we contacted Fedora people or the package maintainers, obtaining clarifications and feedback about the problems we pointed out, as reported in the paper. In many cases, either Fedora or the package maintainers

made changes to address such issues.

The main contribution of this paper is a better understanding of how auditing of licenses happens in a large ecology of packages that use many different licenses. This information is valuable towards the creation of computer support to assist legal auditors in their day-to-day work.

The paper is organized as follows. After a discussion of related work in Section II, Section III details some of the problems that can arise when combining open source software distributed under different licenses. Section IV describes the proposed method to audit open source distributions for license compatibility issues. Section V describes the empirical study we carried out on Fedora-12. Results are presented and discussed in Section VI, while Section VII discusses the threats to validity. Finally, Section VIII concludes the paper and outlines directions for future work.

II. Related Work

This section describes related work concerning (i) the analysis and understanding of legal issues in software systems, and (ii) the analysis of software distributions.

In recent years legal issues and problems concerned with intellectual property have triggered a series of research works. Licenses impose constraints and thus can be defined as logical formulae constraining what can and cannot be done with a system. Software licensing patterns have been recently studied by German *et al.* [1] using such a formalization of licenses. They introduced several legal patterns, along with examples of occurrences of these patterns. As German *et al.* [1] did, we also consider constraints imposed by open source licenses and, in particular, we rely on these constraints to mine inconsistencies (i) between licenses declared in the packages and source code licenses, and (ii) incompatibilities due to dependencies between packages and libraries having different licenses.

German *et al.* [3], presented a study of the influence of software licenses on code migration between the FreeBSD, Linux, and OpenBSD kernels. Their findings support the hypothesis of a preferential code flow induced by permissive licenses from FreeBSD and OpenBSD towards Linux. Hindle *et al.* [4] discovered that many of the largest commits correspond to changes to the licenses or copyright owners of files. We share with these previous works the heuristics used to identify comments and to compare them using information retrieval methods, although we specifically focus on comments related to licenses, *i.e.*, on licensing evolution and changes to copyright years.

Recently, Di Penta *et al.* [5] investigated how licensing statements in source code files change to address the evolution of software licenses—eg licenses such as GPL evolve from version 2 towards version 3—or to cope with the need for re-distributing the software in a certain way (eg the Java Development Kit changed its license to allow its redistribution together with GNU/Linux). We share with that work the finding about licensing change, in fact this is one of the reasons for misalignment between licenses declared in the packages and licenses in source code files.

There have been some attempts towards the creation of automatic environments for the verification of software licenses [6], [7]. In both cases, they take a simple approach: packages are licensed under a single, well defined license, and the dependency data is used to identify potential violations. Alspaugh *et al.* [7] propose a requirement-based approach: the objective is to determine any restrictions in the way components are integrated into a system before this is built. Instead, Tuunanen *et al.* [6] propose a more comprehensive approach: their environment identifies licenses from source code, uses compiling information to determine if two components are connected to find potential violations. In both cases the approach has been evaluated against small applications composed of few components, and they do not deal with the complexities of a large Linux distribution containing more than 10,000 binary applications and hundreds of thousands of source code files (we found 327,286 source code files in Fedora-12).

Although specific on issues related to software licensing, this is not the first study aimed at analyzing entire GNU/Linux distributions. Robles *et al.* [8] and then González-Barahona *et al.* [9] related the evolution of software distributions with the evolution of single applications, finding that the former influences the latter. German *et al.* [2] proposed a method to analyze build dependencies in software distributions, and used it to analyze the Debian GNU/Linux distribution. They showed how the retrieved inter-dependencies helped to understand how packages are used and variants in which the package can be installed. Similarly to this work, they used package descriptions to analyze dependencies—although their study was done on Debian packages, while our study focuses on RPM packages. Stemming from that idea, we propose to integrate package dependency information with license information to identify potential cases of re-distribution with license incompatibilities.

III. Challenges of combining open source components

This section briefly summarizes the main licensing incompatibilities that can arise when integrating and redistributing open source packages. The GNU Public License (GPL) is a very common one for open source packages, and poses strict reuse constraints. For these reasons, and due to lack of space, we focus our attention on incompatibility issues involving the GPL license.

The GPL is a reciprocal license: one of its most onerous constraints is that any software that reuses code licensed under the GPL should also be licensed under the same version of the GPL. Code under some licenses—such as the Eclipse Public License (EPL)—cannot be reused in GPL licensed

packages. See http://fedoraproject.org/wiki/Licensing for a compatibility list of licenses in Fedora.

In GNU/Linux distributions most binary packages have many dependencies. This results in many different combinations of licenses. According to the interpretation of the Free Software Foundation—the author of the GPL—when a program includes source code licensed under the GPL, the resulting program should also be licensed under the same version of the GPL. Similarly, the GPL imposes strong conditions on how a GPL package—in the following referred as *callee*—can be reused by its *caller*. If the caller uses the callee via fork/exec then the caller can have any license. However, if the caller uses the callee via linking (static or dynamic), then the caller has to be licensed under the same version of the GPL as the callee. For further details on this problem see related papers [1] and [10].

Given the above stated constraints imposed by the GPL, it is worthwhile to identify:

1) packages with a declared license that is not the GPL, but that contain files distributed under a GPL license. For example, the declared license of a package is the BSD license, but it contains files under a GPL license;
2) similarly to (1), packages distributed under a license that is not the GPL, but depend on libraries or components distributed under the GPL;
3) packages that have as a declared license a version of the GPL different from the one declared by some (or all) of its files. For example, the license of the binary package is the GPLv2, but some of its files are distributed under the GPLv3. It is common to license files under the GPL version 'X' or *any newer version*; for example, GPL version 3 or any newer version, which we abbreviate GPLv3+. For this reason, if a file is licensed with the option of a newer version, then we only consider it an issue if the declared license of the binary package is an older version than the one found in the source code.
4) Similarly to (3), packages declaring a given version of GPL that depend on libraries/components having a different GPL version.

IV. Auditing license issues in open source projects

This section describes the proposed approach for license auditing. It consists of three steps, aimed at (i) extracting declared licenses and dependencies from the distribution packages, (ii) extracting and classifying actual licenses from the source code files, and (iii) using the information extracted in steps (i) and (ii) to detect licensing incompatibility issues.

Step I: Extracting information from software package metadata: in this first step we extract information from the package management system of a GNU/Linux distribution. In the context of this paper, this was done for RPM (binary) packages of the Fedora-12 GNU/Linux Distribution[1]. To extract this information, we used the RedHat's library *rpm-build*, which provides parsing of Fedora *.spec* files (including expansion of macros). The *.spec* file of each source code package was split into each of its component fields (a pair key, value). A similar process can be followed for other package management systems, such as Debian's *dpkg*.

We downloaded both the Fedora-12 source DVD, comprising 1,475 source code packages, and the Fedora-12 binary (i386) DVD, comprising 2,399 binary packages (one source package can generate one or more binary packages). For each source package we extracted and parsed its corresponding *.spec* file. For each binary package we executed the *rpmquery* command and obtained:

- general information, such as a brief description of the package, the package version and author name;
- the declared licenses(s) of that package, as indicated by the package's maintainer in the metadata;
- the list of resources the package *provides*, in terms of components and libraries (static binary libraries, shared objects, Java packages, Perl and TCL packages);
- the list of resources the applications the applications in package *require*.

It is important to emphasize that the dependency graph extracted from the source package might not be identical to the one extracted using *rpmquery*. The declared dependencies (those in the *.spec* file) are described in terms of names of binary packages; the ones found using *rpmquery* are names of files (executables and shared libraries) needed by a binary package to be installed and to properly function. We therefore see the information from the *.spec* file and from *rpmquery* as complementary.

Sometimes, a package requires one among several possible packages; for example, a package requiring a DBMS might use either MySQL or PostgreSQL, but not both. In such a case, the multiple possible dependencies are prioritized using heuristics such as:

- between two (or more) packages providing the same resource, the one having the same license (or a similar or compatible one) with respect to the requiring package is chosen;
- between two packages offering the same resource but with different versions of the license, we choose the one offering a newer version. For example, if one package requires *qt*, and two other different packages provides *qt 3* and *qt 4.5*, the latter is chosen. Similarly, if a package requires the *jre* (Java Runtime Environment), provided by *java-1.5.0-gcj* and *java-1.6.0-openjdk*, then the latter is chosen.

In summary, we determine (i) the license of each file of the source code of a package; (ii) what binary packages are

[1] http://fedoraproject.org/

created from a source code package and their corresponding license; and (iii) what other binary packages need to be installed before a given binary package.

Step II: Classifying source code file licenses: this step aims at identifying the actual licenses for the source files contained in each package. We first extract the files from the source package, and then we use the *Ninka* license identification tool [11] to classify their licenses. *Ninka* uses a sentence-based approach (based on a set of regular expressions) to detect the presence and identify open source licenses in the header comments of source code files. *Ninka* is capable of identifying more than 110 licenses, and has been designed to identify licenses in source code files. For each file *Ninka* analyzes, it outputs the name of any license it detects in it, or that *no license* was found. If it detects the existence of a license, but it does not know it, then it reports such file as having an *unknown* license. An evaluation of *Ninka* found that it was not capable of finding the license of 10% of the files in the evaluation sample, but when it found a license, it has an accuracy of 93% [11].

It is worthwhile to note that a single source package often corresponds to multiple binary packages. Therefore, it is necessary to map the source code files—for which we classified the license using *Ninka*—to binary packages. There is no simple way to do that; in this work, where this happens, we mainly use information related to the source package directory structure: for example some source packages put files used to produce different binary packages in different sub-directories, others put files for *contrib* or *plugin* in specific directories.

Step III: Mining inconsistencies: the information extracted following the steps described above is stored in a relational database. We query this database to identify incompatibilities according to constraints described in Section III and, specifically: (i) cases where the file license is different from the one declared in its package; and (ii) cases for which the presence of dependencies creates license incompatibilities.

V. Empirical Study

This section describes the study we carried out to show the relevance of the problem discussed in this paper and to show the applicability of the proposed approach. The *goal* of this study is to analyze open source packages from software distributions, with the *purpose* of understanding the kinds of licenses being used by different packages and the presence of dependencies, at various levels, between packages with different licenses. The *quality focus* is the compliance of redistributions and installations of open source packages with respect to licenses, and the *perspective* is of researchers, interested to investigate the presence of possible license incompatibilities in open source software distributions, and to develop a method able to audit the distribution to mine the presence of such incompatibilities; the study can also be seen from the perspective of users, integrators and developers who acquire, modify, combine and re-distribute open source packages, and needs to be aware on the feasibility—from a legal point of view—of what they are doing. The *context* consists in 2,399 binary packages and their corresponding 1,475 source (*src*) packages from the GNU/Linux distribution Fedora-12.

The study aims at addressing the following research questions:

- **RQ1:** *To what extent do the licenses declared in the package metadata reflect the licenses in source code files contained in the packages?* This research question aims at addressing a first level of incompatibility, *i.e.*, those *within a package*. The license declared in the package metadata often summarizes what kinds of licenses can be found within the package. It can happen, however, that the package contains files with different licenses and, above all, licenses incompatible with the declared one. We divided the analysis of **RQ1** in three parts:
  1) Analysis of source code packages that contain source code files distributed under one license only.
  2) Analysis of source code packages that contain source code files distributed under two or more licenses.
  3) A deeper analysis of packages with files distributed under the GPL, which we deem to be one of the most important licenses from the perspective of a license audit. This analysis is, in turn, divided into four parts:
     - I *Source code is under the GPL, but the declared license does not mention the GPL*. Due to the reciprocity requirement of the GPL, we expect that if there is GPL source code in a package, its declared license should be the GPL too.
     - II *Mismatch of GPL version*. Any package containing a file distributed under the GPL should be licensed under the same version of the GPL.
     - III *The declared license includes the GPL, while the source code does not*. If the declared license is the GPL, we would expect that the source code will also contain files under the GPL.
     - IV *The declared license includes the GPL, however there is at least one file under an incompatible license*. A package licensed under the GPL should not include files with incompatible licenses.
- **RQ2:** *Do dependencies between packages highlight license incompatibilities?* This research question analyzes license incompatibilities *between packages*, *i.e.*, it checks whether a package depends on libraries, shared objects, or components, and whether such dependencies use incompatible licenses with respect to the one(s) of the package.

Table I
OCCURRENCES OF LICENSES FOR SOURCE PACKAGES HAVING CODE DISTRIBUTED WITH ONE LICENSE ONLY.

Declared License	Source License	# Src Pkgs	# Bin Pkgs
gplv2+	GPLv2+	118	145
asl 2.0	Apachev2	28	48
lgplv2+	LesserGPLv2.1+	27	36
mit	MITX11noNotice	21	30
mit	MITold	18	23
lgplv2+	LibraryGPLv2+	16	23
gpl+ or artistic	SameAsPerl	14	14
gplv2	GPLv2	11	12
bsd	BSD3	11	11
gplv2	GPLv2+	10	14

Source License in all tables refers to the license as identified by *Ninka*.

VI. RESULTS

This section reports results of our package-based license verification to answer the research questions formulated in Section V. Data for replication is available on-line[2].

A. RQ1: To what extent do the licenses declared in the package metadata reflect the licenses in source code files contained in the packages?

Determining whether the licenses "declared" in a package match those "stated" in source code files is not straightforward, as a package often contains files with several different licenses. The number of licenses found in each package varied between 1 and 41, with a median of 2 licenses, a 25% percentile of 1 license, and a 75% percentile of 3 licenses. The package with the highest number of licenses is the Linux Kernel, which contains 41 different licenses (such as the LGPLv2, LGPLv2.1, GPLv2, GPLv2+, BSD3, BSD2, MIT/X11, etc), even though it is known to be licensed under the GPLv2.

In the following we report results for **RQ1**, concerning specifically (1) the analysis of source code packages containing files distributed under one license only, (2) the analysis of source code packages containing files distributed under two or more licenses, and (3) a deeper analysis of packages containing files distributed under the GPL.

1) Packages using one license only: For these packages, we expect that the declared license of each of the binary packages that it generates to be identical. There exist 429 source code packages with only one license (29% of the total). Table I lists the most frequent licenses, and the number of corresponding binary and source packages. It can be noticed that the license names differ from those reported by *Ninka*; *i.e.*, Fedora names the Apache License Version 2 as the *afl 2.0*, while *Ninka* uses the term *ApacheV2*. Also, *Ninka*'s license granularity level is higher than Fedora's; for example, the table lists *MITold* and *MITX11noNotice* in the source code, while Fedora simply declares it as *mit* license, of which they are variations; similarly, the *LesserGPLv2.1+* and the *LibraryGPLv2+* are both listed as *lgplv2+*.

[2]http://juliusdavies.ca/uvic/fedora12/

A less trivial problem is in the interpretation of the license. For example, many Perl packages are licensed under "The Same Terms as Perl" (see [1]), however Fedora lists them as *gpl+ or artistic* which is the license used by Perl.

To distinguish real cases of mismatches from the above cases of synonymy or of slight variations, we manually built an equivalence table between Fedora and *Ninka* licenses. Taking into account these equivalences, we discovered potential inconsistencies between the declared license and the source code license for 86 packages. However, many of these inconsistencies had a simple explanation. For 22 packages *Ninka* detected a license "SeeFile", which indicates the license is contained in another file, but because *Ninka* only processes one file at a time, it does not scan the contents of such files. For 6 packages, the declared license contained the license of the source code, plus the the General Free Documentation License (*gdfl*) that applies to documentation only, and thus could not be found in the source code. We therefore ignored the *gdfl* when comparing the declared and the source code license. The remaining 64 (out of 86) cases represent real mismatches between the declared and source code licenses.

We manually inspected 12 of the largest packages (from Table II), trying to understand why the declared license was different from the source code license. We classified the results into three levels of warning: (i) *OK*, where Fedora reported the correct license; (ii) *Suspicious*, where Fedora reported what appears to be an incorrect license with respect to those contained in the source code; and (iii) *Unknown*, where it was not possible to compare Fedora's declared license against the source licenses detected by *Ninka*. The *OK* category contains the following cases:

- **Incorrect license identification**. These are the cases where *Ninka* failed to identify some of the licenses stated in the source code files (these were reported by the tool as *unknown*).
- **Optional component.** Part of the source package is distributed under a license different from the one declared in the binary package; however such a part is not built into the binary package. For instance, the *libpng* package contains a "contributions" directory with files under the GPLv2, but these are omitted during compilation.
- **Package used as a component.** A source code package contains a component that uses a license different from the one declared in the binary package. However, such a license is compatible with the declared one. For instance, the *opensp* package contains a directory *intl* that contains internalization code released under the LGPLv2+ license. This directory is likely to be used as a component, and would not interfere with the license of *opensp*.
- **Inconsistent declared license.** The declared license of *automake* includes the license of the installation script (which was not part of our analysis). Similar installation scripts appear in other packages without their scripts' licenses

Table II
RESULTS FOR MANUALLY SCANNED SOURCE CODE PACKAGES WITH ONE LICENSE THAT IS INCONSISTENT WITH THE DECLARED LICENSE.

Warning Level	Issue	Source Package	Declared License	Source License
OK	Incorrect license identification	mysql-connector-java	gplv2 with exceptions	GPLv2
		glade3	gplv2+ and (gplv2+ and lgplv2+) and lgplv2	GPLv2+
		imagemagick	imagemagick	LesserGPLv2+
		gzip	gplv2 and gfdl	GPLv2+
		mpfr	lgplv2+ and gplv2+ and gfdl	LesserGPLv2.1+
		libpng	zlib	GPLv2+
OK	Optional component	libpng	zlib	GPLv2+
OK	Used as a component	opensp	mit	LibraryGPLv2+
OK	Inconsistent declared license	automake	gplv2+ and gfdl and mit	GPLv2+
Suspicious	Fedora False Positive	eclipse-cdt	epl and cpl	EPLv1
Suspicious	License change	bsf	asl 1.1	Apachev2
		mtools	gplv2+	GPLv3+
Unknown	License was not found	ortp	lgplv2+ and vsl	LesserGPLv2.1+

appearing in the declared license, so including such is not consistent across all packages. There is no legal issue with *automake*, since the *install.sh* script is distributed under an academic license (MIT).

The *Suspicious* category contains:

• **Fedora false positives.** Fedora lists a license that is mentioned in the documentation, but that it is not actually used to license the source code of the package. However, differently from the case of the *gdfl* mentioned above, such a license cannot be discarded a-priori as it is not just a license used for the documentation, but might be used to distribute source code. Specifically, the *Common Public License* is mentioned in the documentation files of *eclipse-cdt*, however its source code files are licensed under the EPLv1.

• **Evolution of license.** The package recently upgraded its license. We believe that Fedora performed the license analysis before these packages changed their license, and might not be aware of the change. The issue of licensing evolution is particularly frequent and relevant in open source systems, as described in [5]. The packages *bsf*, and *mtools* belong to this category, as their license changed from GPLv2+ to GPLv3+, while Fedora was still declaring them as GPLv2+. We reported these issues to Fedora, and we were informed that they were already aware of *bsf* (it is fixed in Fedora-13), while they acknowledged the problem for *mtools*, and at the time of writing were looking into it.

2) Binary packages with different licenses that originate from the same source code package: As mentioned in Section IV, a source package can generate more than one binary package. Sometimes a source package generates several binary packages with different declared licenses among them. This happens for 31 source packages. We selected and manually analyzed the 13 largest ones to understand their licensing. The results are shown in Table III and can be categorized as follows:

• **Split library from programs.** Many libraries are licensed under the LesserGPL or the LibraryGPL (LGPL). However, in some cases, their source packages might contain executables—licensed under the GPL—that use the libraries. Libraries are placed in one binary package and programs in another.

Table III
SOURCE PACKAGES THAT GENERATE BINARY PACKAGES WITH DIFFERENT LICENSES (SEPARATED BY SEMICOLON).

Warn Level	Issue	Declared License	Source Package
OK	Executables are separated from libraries	gplv2; lgplv2	cdparanoia
			cups
			e2fsprogs
		gplv2+; lgplv2+	audit
			gnome-bluetooth
			gimp
			glade3
			gnome-desktop
			gnome-doc-utils
			gnome-panel
			kdemultimedia
OK	A few files with a different license	gpl+ or artistic; (gpl+ or artistic) and (gplv2+ or artistic)	perl
OK	Each file has its own license	gplv2; gplv2+; gplv2+ w/ exceptions; gplv3+ w/ exceptions; gplv3+ w/ exceptions and gplv2+ w/ exceptions and gplv2+ and gplv2	smc-fonts

• **A small number of files having a different license.** We found one case (*Perl*), for which there was one single file with a different license than the others. The source package of Perl creates 19 binary packages, but one of them has a different license (GPLv2+). Only one source code file of the *Perl package* uses this license.

• **Each file has its own license.** The source package *smc-fonts* creates 8 different binary packages. Each binary package corresponds to one source file only, and each source file has a different license.

3) Packages with files distributed under the GPL: As described in Section III we focus on specific issues related to the GPL license. As explained in Section V, we divided the analysis into four parts, discussed below.

I) *Source code is under the GPL, but the declared license does not mention the GPL.* We scrutinized the declared licenses of binary packages that did not include any of the different versions of the GPL, trying to find possible inconsistencies. Sometimes a source package contained one or more files under the GPL with an exception that allowed their use under another license. For example, any interpreter generated by *bison* is licensed under the GPL with a special

exception that allows its use under other licenses. We therefore ignored any files with a GPL license that included an exception. After removing these cases, there were 115 source packages left that do not declare the GPL even though they include at least one file licensed under the GPL. We inspected 18 of them to understand why the GPL was not mentioned. The results of the inspection are summarized in Table IV.

Table IV
SOURCE PACKAGES WITH GPL SOURCE CODE, BUT ITS LICENSE DOES NOT INCLUDE THE GPL.

Warning Level	Issue	Source Package
OK	Split library from programs	libnice, libopenraw, libgxim, libgsf, kdelibs, gnome-python
OK	Plugin	opal
OK	Contrib or extras	spambayes, antlr, subversion, cscope, xscreensaver
OK	Tests or Demos	mesa, pcsc-lite
OK	Special permission	vim
OK	License name or interpretation	wpa_supplicant, iptools
Suspicious	Many GPL versions	kdebindings

• **Split library from program.** As mentioned above, many libraries split their packages into actual libraries (in one binary package, licensed under the LGPL), and applications using those libraries (in another, under the GPL).
• **Plugin.** The architecture of the system allowed some files to have a different license.
• **Contrib or extras.** These files were contained in the source package, but they were not used to build the binary packages.
• **Tests or demos.** These packages contained test cases or demo programs under the GPL.
• **Special permission.** *vim* contained one file under the GPL with an explicit note that stated permission to use it under a different license.
• **License name or interpretation.** In one case we could not find an explanation of what a declared license was (*iptools*), while in the other case the name of the declared license (*wpa_suplicant*) was actually a disjunctive license of the GPL and BSD.

The only suspicious package *kdebindings* contained many files under many packages. This is a very heterogeneous package, composed of many different library bindings for different programming languages, each with different licenses, and probably the reason for many licensing conflicts. We have reported this to Fedora, and at the time of writing they are looking into it.

II) *Mismatch of GPL version*. Table V reports the cases in which a declared license of the package was different than the license of at least one file contained in the source package (excluding again files with a GPL-with-exception license). The first 18 cases (marked as **Ok**) do not represent real problems: if the source code permits to license it under a newer version of the stated license (the "+" suffix in our abbreviations), then it can be embedded in a binary package with a newer license. The other 25 cases (marked as **Suspicious**) require a more in-depth analysis. We inspected 10 of these suspicious packages, and reported the results of such an inspection in Table VI. As it can be seen, the main issues we found are:

Table V
PACKAGES DECLARING A SPECIFIC GPL LICENSE VERSION AND CONTAINING FILES WITH A DIFFERENT VERSION OF GPL (ONLY FIRST 10 DUE TO SPACE RESTRICTIONS).

Warning Level	# Binary Packages	Declared License	Source License
Ok	15	gplv3+	GPLv2+
	2	gplv3	GPLv2+
	1	gplv3+ with exceptions	GPLv2+
Suspicious	13	gplv2+	GPLv3+
	5	gplv2	GPLv3+
	2	gplv2+ and gfdl	GPLv3+
	2	bsd and gplv2 and ijg and mit and public domain	GPLv3+
	1	gplv3+ and redistributable, no modification permitted	GPLv2+
	1	gplv3+	GPLv2
	1	gplv2 and gplv2+ and bsd with advertising and public domain	GPLv3+

Table VI
ANALYSIS OF SOME SOURCE PACKAGES THAT CONTAIN FILES DISTRIBUTED WITH DIFFERENT GPL VERSIONS.

Warning Level	Issue	Package	Declared License	Source License
Suspicious	License Evolution	fetchmail	gplv1+	GPLv2+
		iptables	gplv1+	GPLv2+
		cvs	gplv1+	GPLv2+
		bash	gplv2+	GPLv3+
		bison,	gplv2+	GPLv3+
	Some inconsistent files	mtools	gplv2+	GPLv3+
		vinagre	gplv2+	GPLv3+
	Contradictory documentation	vinagre	gplv2+	GPLv3+

• **License evolution.** As previously reported, some packages have changed license, but the declared license is obsolete. Fedora has acknowledged this problem and it will be resolved in the next version.
• **Inconsistent files.** In two packages (*mtools* and *vinagre*) there were some files with an inconsistent license (GPLv3). Fedora has been notified and is looking into these packages.
• **Contradictory documentation.** In one case (*vinagre*) the package documentation states GPLv2+, while some files were GPLv3+.

III) *The declared license includes the GPL, while the source code does not.* This situation arose in 46 different packages. We looked at 8 of them, and the results of the inspection are summarized in Table VII. In particular, we found the following situations:
• **See File.** The license is stated in a file separated from the source code;
• **Same as Perl.** As described before, the package has the same license as Perl (gplv1+ or artistic).
• **License not classified.** *Ninka* could not classify the license

Table VII
ANALYSIS OF SOME SOURCE PACKAGES THAT ARE LISTED AS GPL BUT DO NOT CONTAIN ANY FILE UNDER THAT LICENSE.

Warning Level	Issue	Packages
Ok	See File	cups
	Same as Perl	perl-dbi, perl-libwww-perl
	Files not classified	less, procps
	License in documentation	libofa
Suspicious	Incorrect declared license	rpcbind, nc

Table VIII
PACKAGES UNDER THE GPL WITH CODE UNDER THE BSD-4

Warning Level	Issue	Packages
Ok	Copyright by UofC	ftp, guile, kernel, nmap, rpm, squid
	Copyright by NetBSD	exiv2, rpcbind
	Sample code	bash
Suspicious	Files using BSD-4	cups, isdn4k-utils, xen

of any of the GPL files of the package.

• **License in documentation**. In *libofa* the license was found in the README file. Its source code had no licensing information.

• **Incorrect declared license.** Fedora had incorrectly labeled three packages (*nc*, *rpcbind*, and *libofa*) as GPL. For *nc*, the word GPL was found in the source code documentation "No GPLs, Berkeley copyrights or any of that nonsense.". Thus, very likely, Fedora developers used a simple text matching approach (in this case) to mine licenses in packages to be re-distributed. This problem has been fixed in Fedora-13. For *rpcbind* and *libofa* the installation scripts contained the GPL.

IV) *The declared license includes the GPL, however there is at least one file under an incompatible license*

We focused our attention to GPL packages with at least one BSD-4 file. We analyzed all cases (12 packages), and the results are summarized in Table VIII.

• **Copyright owner clarification.** Most of these cases were resolved because the original copyright owner of the BSD license, the The Regents of the University of California, indicated that the "advertising" clause can be removed from any source code under the license. Hence, any BSD-4 license file from them is effectively under the BSD-3. A similar situation arises when the copyright owner is the NetBSD Foundation.

• **Tests or Demos.** *bash* contained BSD-4 code but only in test cases.

Only three cases (*isdn4k-utils*, *xen* and *cups*) contained suspicious files, which we reported to Fedora and/or upstream.

B. RQ2 Do dependencies between packages highlight license incompatibilities?

There exist 1,516 different combinations of *caller license*/*callee license*, with a median frequency of 3, a minimum of 1, and a maximum of 2,424. Not surprisingly, the most common callee licenses allow reuse with few conditions (*i.e.*, Apache, BSD, MIT, zlib, lgpl, artistic). We pruned the list of 1,516 license combinations, with the aim of keeping only the ones highlighting potential incompatibility issues. Specifically, we pruned out:

- Cases where the caller and the callee have the same license.
- Cases where the license of the caller is one of the licenses of the callee.
- Combinations that do not cause license incompatibility problems. For example, when the callee uses a license (such as LGPL, MIT, BSD) that allows the caller to use any license.

Table IX
SUSPICIOUS COMBINATIONS BETWEEN CALLER AND CALLEE.

Caller Lic	Callee Lic	Occurrences
gplv2+	python	75
gplv2+	gplv3+	64
lgplv2+	python	50
gplv2	gplv3+	30
mit	gplv2+	27
lgplv2+	gplv3+	22
asl 2.0	gplv2+	20
gplv2	python	19
mit	python	16
mit	(gpl+ or artistic) and (gplv2+ or artistic)	15

The pruning left a total of 309 different combinations, with a median (and minimum) frequency of 1, and a maximum of 75. Table IX shows the most common combinations.

Whether or not a given combination of licenses is permitted or not also depends on the way that the caller interconnected to the callee. For example, according to the Free Software Foundation, an application distributed under the GPL cannot link to a component under the Python license (before version 2.0.1); however it can run the Python interpreter as an executable.

Through manual inspection we discovered the caller interconnected with the callee using exec/fork in the majority of the suspicious combinations. For example, in many cases where we observed a callee with a Python license, the package in question was the Python interpreter, presumably used as an executable (and not linked as a library). Similarly, when a package distributed under the GPLv2+ uses a callee distributed under the GPLv3+, it is often the case the callee is a common Unix command, such as *grep*, *gawk*, *gdb*, *m4*, *cpio*, *tar*, *wget*, *find*, etc. Careful investigation of these Unix commands found that many of the callee packages contained only executables, documentation, and configuration files, therefore precluding any chance of dynamic or static linking. In a few cases where library files were found among these commands, they were either not in a location available for dynamic-linking, or a close analysis of the combination found an overt exec/fork.

We sampled the remaining suspicious dependencies looking for potential problems. Below we describe some of the most interesting cases:

- *kdebase-workspace.* Code distributed (GPLv2) was linking to GPLv3+ code (packages *libqzion* and *libqedje*). We inquired the Fedora License maintainers about it. They discovered that these libraries were being used by programs in *kdebase-workspace* that were licensed under the GPLv2+, even if the entire package was licensed only as GPLv2. Hence, theoretically, such programs could be extracted from the package and relicensed under the GPLv3+, the license of the packages they were linking to.
- Two GPLv2 source packages (*lvm2*, *pilot-link*) were using *readline* (GPLv3+). These appeared related to the evolution and change of licenses: in previous versions of Fedora the *readline* package used the GPLv2+. Fedora is currently trying to resolve these by replacing the *readline* dynamic links in these two cases with the *libedit* library, which provides a similar function but is licensed under the BSD3.
- *php* (which uses the PHP license, considered to be incompatible with the GPL) was also dynamically linking to *readline*. This problem was created because *php's* build scripts would link to either *readline* or *libedit* if such are available in the build environment (with *php* giving *readline* priority). We contacted Fedora, and they promptly fixed their build environment to prevent *php* from using *readline*.

VII. Threats to Validity

This section discusses the main threats to validity that can affect the study we performed.

In particular, threats to *construct validity* may concern imprecision in the measurements we performed. Specifically, there could be imprecision in *Ninka* license classification, although previous studies indicate a precision of 96.6%. Also, licenses declared in the Fedora packages can be, in turn, imprecise or out-of-date. However, the latter does not constitute a threat as we are interested to point out problems due to incorrectly declared licenses.

Dependency analysis done using *rpmquery* is also a source of imprecision. Above all, *rpmquery* can only identify dependencies between an entire package and some components. This does not mean that all applications contained in the package have these dependencies. Future work will aim at performing dependency analysis at a lower level of detail, also analyzing linking dependencies of single application. To mitigate such a threat, we analyzed all cases of possible inconsistency, to check whether it was a real inconsistency or, instead, a false positive.

Threats to *internal validity* do not apply to observational studies like the one we performed. Nevertheless, as reported in our discussions, we have checked the cases where our automatic analysis reported symptoms of likely licensing problems, and we tried to understand the reasons by (i) manually inspecting the package (metadata and licenses declared in the source code) and (ii) communicating with the developers.

Table X
Results of interactions with Fedora and Upstream.

Status	Issue	Source Package
Resolved Upstream	Incorrect license in sources	enchant, kdesdk, wireshark
Resolved Independantly	Incorrect license in sources	xen
Resolved by Fedora	Incorrect declared license	abrt
	Dynamic linking with GPL	php
Acknowledged by Fedora	Dynamic linking with GPL	lvm2, pilot-link
Reported Upstream	Incorrect license in sources	cups, isdn4k-utils
Reported to Fedora	Incorrect declared license	alsa-utils, bison, eclipse-cdt, fetchmail, firstboot, iproute, iptables, kdebindings, mtools, ortp, rpcbind, vinagre, vino, yum

Threats to *external validity* concern the generalization of our results. The study concerns a very large number (2,399) of software packages belonging to the Fedora-12 GNU Linux distribution, developed by different authors, having different licenses and depending on different sets of other applications. However, further studies—*e.g.*, on other distributions like the Debian, using different package format—would be desirable.

VIII. Conclusion

FOSS systems are distributed as packages—such as the RedHat/Fedora RPM packages or the Debian *.deb* packages, generated from source code distributed under a wide variety of licenses, and requiring the installation of libraries or components that, in turn, have other licenses. The large set of existing licenses, the constraints they pose, and the complex dependency relationships among packages determine a situation in which it is difficult to understand under what conditions a package can be used and/or redistributed. Although software distributions—eg GNU Linux distributions—perform their own license auditing—this is often insufficient.

This paper described a semi-automatic method to audit licensing compatibility issues in software distributions, and reports a study in which we audited Fedora-12 packages. The study allowed us to show several licensing incompatibility issues, concerning both the mismatch between declared licenses and those stated in the source code, and dependencies between packages and libraries with different licenses. In many cases we contacted the package developers or Fedora people for clarification or to highlight potential problems—which in several cases were real ones. Overall, from the study we conducted, it can be learned that auditing licensing issues is quite complex due to several reasons:

- *Automatic auditing of source code is, in general, difficult*. The solution is likely to be tools that assist the experts who do it. Such tools guarantee that the

person doing the audit spends her time looking at the most challenging issues first, without wasting time to scrutinize cases where it is evident there cannot be licensing problems.

- *A correct license identification is crucial for the analysis*. Even though *Ninka* was capable of identifying in 93% of the source code files, there are still cases where automatic license identification is challenging.
- *Many packages contain source code under different licenses.* This makes determining the license of the entire package difficult. Package maintainers could help simplify this problem, by splitting a source package into multiple packages, each one distributed under a different license, and making sure that the license of a binary package is always the same as the one of its source package.
- *Licenses evolve over time*. We found many cases in which the license of a package changed, and this created problems, *e.g.*, the package still declared the old license, or one of its callers was using an older license, making the package use potentially incompatible.
- *Packages interact in various ways*, and it is not trivial to determine if they violate the license of its dependencies (callees).

It is important to emphasize that, even though our method discovered many issues, we cannot claim it can find them all. As described before, whether files (or packages) with two licenses can coexist together depends if and how they interconnect together. This is an area that requires ample further work. Nonetheless, our study proved effective, as we discovered several suspicious inconsistencies that we reported to Fedora and to the *upstream* developers (the term upstream refers to the developer community that develops and maintains the product in the source package). The results are summarized in Table X. Some of these issues have been already resolved, while are others are still being evaluated.

In summary, the empirical study showed that a deeper understanding of licensing issues requires human expertise and might not be fully automatic. Nevertheless, future work should be devoted to incorporate in auditing methods and tools additional heuristics—many of them documented herein—to reduce the amount of manual inspection needed.

Also, we aim at extending the proposed approach by (i) using finer-grained dependency information between single applications/components rather than at package level; and (ii) keeping into account license changes as they can be captured from versioning systems [5]. Last, but not least, it is desirable to replicate the study on other GNU/Linux distributions.

We also acknowledge the prompt responses from Fedora's licensing team and other individuals in the *fedora-legal-list*[3] and from the package maintainers. They are truly committed to making sure any license inconsistencies are resolved.

[3]https://www.redhat.com/mailman/listinfo/fedora-legal-list

ACKNOWLEDGEMENTS

We thank Tom "Spot" Callaway and Richard Fontana from Red Hat's Legal team for their invaluable help in the preparation of this manuscript. The work of D.M.German has been funded by Hewlett-Packard to support the FOSSology Project. The work of J. Davies has been funded by a University of Victoria Undergraduate Research Scholarship.

REFERENCES

[1] D. M. Germán and A. E. Hassan, "License integration patterns: Addressing license mismatches in component-based development," in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*. IEEE, 2009, pp. 188–198.

[2] D. M. Germán, J. M. González-Barahona, and G. Robles, "A model to understand the building and running inter-dependencies of software," in *14th Working Conference on Reverse Engineering (WCRE 2007), 28-31 October 2007, Vancouver, BC, Canada*, 2007, pp. 140–149.

[3] D. M. German, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "Code siblings: Technical and legal implications," in *Proc. of the 2009 Working Conference on Mining Software Repositories, MSR 2009*, 2009, pp. 81–90.

[4] A. Hindle, D. M. German, and R. Holt, "What do large commits tell us? a taxonomical study of large commits," in *MSR '08: Proc. of the 2008 international working conference on Mining software repositories*, May 2008, pp. 99–108.

[5] M. Di Penta, D. M. German, Y.-G. Gueheneuc, and G. Antoniol, "An exploratory study of the evolution of software licensing," in *Proceedings of the ACM/IEEE 32rd International Conference on Software Engineering (ICSE 1010) 2-8 May 2010, Cape Town, South Africa*, 2010.

[6] T. Tuunanen, J. Koskinen, and T. Kärkkäinen, "Automated software license analysis," *Automated Software Eng.*, vol. 16, no. 3-4, pp. 455–490, 2009.

[7] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi, "Intellectual property rights requirements for heterogeneously-licensed systems," *Requirements Engineering, IEEE International Conference on*, vol. 0, pp. 24–33, 2009.

[8] G. Robles, J. M. González-Barahona, M. Michlmayr, and J. J. Amor, "Mining large software compilations over time: another perspective of software evolution," in *Proc. of the 2006 International Workshop on Mining Software Repositories, MSR 2006, May 22-23, 2006*. ACM, 2006, pp. 3–9.

[9] J. M. González-Barahona, G. Robles, M. Michlmayr, J. J. Amor, and D. M. Germán, "Macro-level software evolution: a case study of a large software compilation," *Empirical Software Engineering*, vol. 14, no. 3, pp. 262–285, 2009.

[10] L. Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law*. Prentice Hall, 2004.

[11] D. M. German, Y. Manabe, and K. Inoue, "A sentence-matching method for automatic license identification of source code files," Under review, available at http://turingmachine/~dmg/papers/, 2009.