Software Freedom Law Center

1995 Broadway, 17th Floor
New York, NY 10023–5882
tel +1–212–580–0800
fax +1–212–580–0898
www.softwarefreedom.org

Shareware Redistribution of FOSS Software

James Vasile

27 March 2008

1 Introduction

The Software Freedom Law Center has prepared this document for free and open source ("FOSS") software developers concerned about redistribution of their software in violation of the free software licenses under which they are distributed. Specifically, this paper discusses FOSS packages compiled into binary form and redistributed as shareware software that is available without monetary charge but is accompanied by a request or demand for payment to the author (typically for the Microsoft Windows operating system). This document discusses how and when such redistribution violates common FOSS licenses, and describes steps that can be taken during the development process to make enforcement easier in these cases.[1]

2 FOSS as Shareware

There is nothing inherently problematic about distributing or redistributing FOSS software as shareware under a variety of FOSS licenses so long as such distribution complies with the licenses of the underlying FOSS works. Unfortunately, shareware redistributors often fail to meet these obligations, particularly with respect to clauses concerning copyleft, source code distribution, attribution, and license fees.

2.1 Copyleft

"Copyleft" is a play on the word "copyright." Whereas copyright law has traditionally been used to withhold permission to copy, modify or distribute software, copyleft licenses instead use copyright law to *require* that such permissions be granted. Some free software licenses have stronger copyleft clauses than others, but they all share a common effect: creation of a large pool of software that can be combined and built upon to create new works. Copyleft licenses require that those who take material from the common pool give something back as well.

[1] The techniques presented in this document are applicable to *any* binary redistribution of FOSS software. They are presented here in the context of shareware to draw attention to shareware infringement of FOSS licenses.

Many shareware redistributors of FOSS software purport to license their shareware under terms that restrict their users' rights to copy, modify and distribute the shareware. In almost every instance, such restrictive licensing violates any copyleft clauses in licenses of the underlying free software.

2.2 Source Distribution

In order to exercise the right to redistribute modified versions of FOSS shareware, one needs access to the source code. For this reason, many free software licenses, notably the GNU General Public License (the "GPL"), require that source code be delivered or made available to recipients of the software.[2] Nonetheless, shareware redistributors of software often distribute binary packages without making any provision for providing source code to users. Such distribution can violate free software licenses such as the GPL.

2.3 Attribution

Most FOSS licenses, including permissive ones such as BSD-style or ISC licenses, require redistributors to preserve copyright notices or otherwise give attribution to the original developers of a software project. Many shareware redistributors, though, fail to preserve such notices. In almost every case, such actions violate any underlying FOSS licenses.

This is not to say that FOSS licenses protect *all* attributive elements. Some protect more than others, and in some instances, trademark concerns sometimes require shareware redistributors to refrain from redistributing certain attributive elements, such as logos. Often, license protection of attributive elements is limited to copyright notices, and developers should review the specific requirements of their licenses to fully understand their protections and obligations in this area.

2.4 License Fees

Shareware redistributors expect users to pay them money in return for providing software. This is not, in itself, a problem under any commonly-used FOSS licenses. Neither the permissive, non-copyleft licenses (BSD-style, ISC, etc.) nor the copyleft licenses (i.e. the GPL family of licenses) prohibit charging a fee for a copy of the software.[3]

Copyleft licenses do, however, require that once software is delivered, users be given certain freedoms. Using copyright law to condition those freedoms on monetary payment (or anything else) violates copyleft clauses.[4] As such, shareware that purports to legally require payment in return for continued use of the software violates any copyleft provisions in the underlying free software license.[5]

[2] *See* GPL ver. 3 §6;GPL ver. 2 §2.

[3] *See, e.g.,* GPL ver. 3 at §4 ("You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee."); GPL ver. 2 at §1 ("You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.")

[4] Note that using *technical* means to require payment might not violate the letter of copyleft clauses as long as the user has the source code and the right to remove such technical measures.

[5] *See, e.g.,* GPL ver. 3 at §10 ("you may not impose a license fee, royalty, or other charge for exercise of rights granted under this license.")

3 Enforcement

Most of the time, the license violations described above can be seen and demonstrated just by looking at the shareware packages offered for download. As such, proving non-compliance with FOSS licenses is relatively easy. Often, the difficult part is demonstrating that the shareware package includes FOSS software and that therefore license compliance is required.

FOSS developers can only require compliance with their licenses to the extent that shareware packages are derivative works of their FOSS software. Demonstrating that one binary package derives from another can be challenging when shareware redistributors obscure the source of their software. Such obfuscation can take several forms, including modifying interfaces, stripping documentation, removing or changing attribution, and refusing to distribute source code.

3.1 Creating Unique Identifiers

If shareware redistribution occurs in such a way as to obscure its FOSS origins, bare observation might not suffice to demonstrate that the shareware derives from specific FOSS packages. More in-depth analysis of the binary shareware package must be undertaken to find elements common to a developer's FOSS package and the shareware in question.

This in-depth analysis can include examining a program's behavior at run time, searching the binary package as a raw data stream or comparing media distributed with the package. In all of these instances an examiner looks for elements copied from the FOSS package into the shareware distribution. Copying can be demonstrated by the presence of elements that are (1) demonstrably authored by the FOSS developer and (2) common to both packages.

The most useful of these common elements are directly comparable, unique identifiers that are in and of themselves copyrightable and clearly originating with the FOSS developer. Such elements usually represent specific and unique parts of the FOSS package and its development team. They can be as simple as strings in the code, as clever and demonstrative as easter eggs, or as subtle as digital watermarks in graphics and videos.

Often, analysis demonstrates similarities and evidence of copying, but sometimes unique identifiers are difficult to find. FOSS authors worried about demonstrating the provenance of their code redistributed as shareware should consider including some of these unique elements in their FOSS packages.

3.1.1 Unique Strings

One of the most common ways of demonstrating that a binary is a derivative work of source code is to search that binary for unique strings from the FOSS source distribution. These strings can be attributive marks, documentation, prompts, code symbols (i.e. variable and function names) or anything else. They don't even have to be text. So long as they are unique to the FOSS source code and clearly originate with the FOSS author, they can demonstrate the copying.

Such strings do not have to be copyrightable in their own right. Instead, they act to demonstrate that copying of other copyrighted material has occurred.

3.1.2 Easter Eggs

Easter eggs are hidden features in software purposefully meant to be entertaining, humorous and difficult to access accidentally. Usually easter eggs are of little utility, although in some types of software (games, usually), they can unlock additional program functionality.

Because easter eggs are creative, unique and hidden features, they are excellent markers of authorship and copying. And because they are usually intertwined with the functionality of software, they can be difficult to remove.

Developers should note that if an easter egg might be presented to demonstrate copying, such features should produce output that is presentable to a legal adversary, the public or a court.

3.1.3 Cutting Room Scraps

Developers whose FOSS package includes digital photos, video or music, can save (and not distribute) copies of the high resolution or high bit-rate, pre-edited source material used in production. These files are demonstrative proof of authorship of such media content.

3.1.4 Digital Watermarks

Digital watermarks and fingerprints are mathematical transformations applied to digital media data. The transformation does not affect how the media is displayed or played for end users, but it provides a digital signature that can demonstrate authorship.

Many FOSS developers believe the presence of their media files in the shareware copy is sufficient to demonstrate that copying has occurred. Unfortunately, there are additional steps: FOSS authors often need to be able to demonstrate that they are the authors of the media files in question to show they have a copyright interest. Watermarks make that demonstration much easier.

Of the techniques mentioned here, watermarking will often require the most additional work. However, if a large portion of the value of your FOSS package is the accompanying media files, it might be worth the effort.

3.1.5 Bugs

Sometimes bugs produce revealing errors that are idiosyncratic enough that they can identify a piece of code. While those bugs should be fixed, they should also be noted. When other code behaves in the same buggy, idiosyncratic way, that behavior can demonstrate copying.

3.2 Using Unique Identifiers

Not all of these methods of marking software will work for every package or developer. In choosing among these techniques, some considerations to keep in mind are the number and variety of techniques used. As the number and variety of common elements increases, it becomes easier to demonstrate that the shareware redistribution is subject to the FOSS package's license.

Towards this end, FOSS developers concerned about this issue can use more identifiers and employ a variety of techniques. When inserting such elements into FOSS packages, developers should take care not to call too much attention to them. The more effort it takes to obscure or remove such marks, the more likely they will survive repackaging.

4 Conclusions

FOSS licenses allow shareware redistribution of FOSS software if the applicable FOSS licenses are followed. When they are not, developers sometimes face difficulty enforcing their FOSS licenses against binary-only shareware redistribution because it can be difficult to prove that the shareware is based on the FOSS project.

There are a variety of techniques that can be applied during the development process to ease future enforcement efforts. These techniques can be combined to demonstrate copying of a FOSS package in a shareware redistribution. Developers concerned about license-violating, binary-only redistribution (whether in the shareware context or otherwise) should familiarize themselves with these techniques.