Washington Journal of Law, Technology & Arts

Volume 7 | Issue 3 Article 5

1-1-2012

Loaded Question: Examining Loadable Kernel Modules under the General Public License v2

Curt Blake

Joseph Probst

Follow this and additional works at: https://digitalcommons.law.uw.edu/wjlta

Part of the Intellectual Property Law Commons

Recommended Citation

Curt Blake & Joseph Probst, *Loaded Question: Examining Loadable Kernel Modules under the General Public License v2*, 7 WASH. J. L. TECH. & ARTS 265 (2012).
Available at: https://digitalcommons.law.uw.edu/wjlta/vol7/iss3/5

This Article is brought to you for free and open access by the Law Reviews and Journals at UW Law Digital Commons. It has been accepted for inclusion in Washington Journal of Law, Technology & Arts by an authorized editor of UW Law Digital Commons. For more information, please contact lawref@uw.edu.

WASHINGTON JOURNAL OF LAW, TECHNOLOGY & ARTS
VOLUME 7, ISSUE 3 WINTER 2012

LOADED QUESTION: EXAMINING LOADABLE KERNEL MODULES UNDER THE GENERAL PUBLIC LICENSE V2

Curt Blake and Joseph Probst[*]
© Curt Blake and Joseph Probst

Cite as: 7 Wash J.L. Tech. & Arts 265 (2012)
http://digital.law.washington.edu/dspace-law/handle/1773.1/1115

ABSTRACT

This Article examines the intersection of Linux loadable kernel modules and the license under which Linux is distributed, the General Public License (GPL) Version 2. Section I of this Article discusses ambiguous terms contained within the GPL and various interpretations of these ambiguities. Next, Section II analyzes the changing scope of legal protection for computer software, particularly as it pertains to derivative works and as applied to loadable kernel modules. Section III highlights provisions contained within the GPL that may attempt to reach beyond a traditional works analysis and examines these provisions in light of recent developments at the intersection of contract law and intellectual property licensing.

---

[*] Curt Blake, University of Washington School of Law, Class of 1983. Thanks to Joseph Probst for collaborating with me on this paper, to Robert Gomulkiewicz for helping get a fellow old guy published, and to my wife Kelli and my children Gavin and Anna for keeping things fun.

Joseph Probst, University of Washington School of Law, Class of 2012. Thank you to Curt Blake for the collaborative efforts leading to this paper, to Professor Robert Gomulkiewicz for providing me with the inspiration to further analyze the legal protection of computer software, and to my family for their enduring love and support.

TABLE OF CONTENTS

Introduction 267
I. Obligations Under the GPL 269
II. Applicability of the GPL to Loadable Kernel Modules under a Derivative Works Analysis 272
A. The Evolution of the Derivative Works Test Applied to Software 272
B. Applicability of the Modern Derivative Works Test to Loadable Kernel Modules 276
III. Applicability of the GPL Beyond a Derivative Works Analysis 279
A. Alternative Interpretations of the GPL 279
B. Recent Decisions on the Intersection of Copyright Law and Contract Law 282
C. General Public License or General Public Contract? 285
D. Alternative Interpretations of the GPL Applied in Light of MDY 287
1. Distribution of a Loadable Kernel Module Standing Alone 287
2. Distribution of a Loadable Kernel Module in Conjunction with an Unmodified Linux Kernel 289
3. Distribution of a Loadable Kernel Module in Conjunction with a Modified Linux Kernel 290
E. MDY's Effect on Availability of Remedies for Non-Compliance with the GPL 291
Conclusion 293

INTRODUCTION

As manufacturers increasingly rely on embedded devices[1] to incorporate greater levels of intelligence into embedded systems,[2] demand has grown for the software required to operate these embedded devices. Embedded devices are used in common products like cellular phones, digital cameras, automobiles, and medical instruments. Demand for inexpensive, small operating systems to run these devices has grown as the price of memory and microprocessors has fallen, and the desire for "smart" functions in a variety of devices has risen. The Linux operating system caters to this demand, boasting a smaller footprint than Windows and, due to its open source heritage, a very attractive price tag. The increasing popularity of Linux has generated a need for software that facilitates interaction between the Linux operating system kernel and the specific hardware of the embedded device. Often, the solution takes the form of loadable kernel modules, such as device drivers, which communicate between a piece of hardware and the underlying Linux kernel.[3]

---

[1] *See, e.g.*, *Embedded System*, NETRINO: EMBEDDED SYSTEMS GLOSSARY, http://www.netrino.com/Embedded-Systems/Glossary-E#embedded_system (last visited January 17, 2012). An embedded system is a computer system designed to do one or a few dedicated and/or specific functions, often with real-time computing constraints. It is *embedded* as part of a complete device, which often includes hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

[2] *See, e.g.*, *id.* Embedded systems span all aspects of modern life and there are many examples of their use. Telecommunications systems employ numerous embedded systems from telephone switches for the network to mobile phones at the end-user. Computer networking uses dedicated routers and network bridges to route data. Consumer electronics include personal digital assistants (PDAs), mp3 players, mobile phones, videogame consoles, digital cameras, DVD players, GPS receivers, and printers. Many household appliances, such as microwave ovens, washing machines and dishwashers, are including embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

[3] ALESSANDRO RUBINI & JONATHAN CORBERT, LINUX DEVICE DRIVERS (2d ed. 2001), *available at* http://www.xml.com/ldd/chapter/book/index.html.

Unfortunately for many developers, the legal consequences of linking to Linux kernels are unsettled. Uncertainty in this area exerts a chilling effect on the development of embedded devices. Developers who have created proven functions for embedded devices running on non-Linux operating systems want to port those functions to embedded devices running on the Linux operating system. At the same time, manufacturers of embedded devices want to use their trusted software partners as they develop their next generation of products in a Linux-centric world. Uncertainty regarding the legal consequences of linking proprietary software to a device running the Linux kernel makes it difficult for developers of proprietary software and embedded devices to reach agreement.

The reason for this uncertainty is the General Public License (GPL),[4] the license to which those using, modifying, or distributing Linux are bound.[5] Several of the key terms used throughout the GPL are poorly defined or used inconsistently.[6] When the ambiguity in these terms is combined with the evolving scope of protection afforded to computer programs by judicial interpretations of the Copyright Act, module developers are unable to properly ascertain the extent of their rights and restrictions.[7]

This Article first analyzes the special case of loadable kernel modules under a narrow interpretation of Section 2 of the GPL, under which the GPL's "copyleft" requirements only apply to works which would be derivative works under the Copyright Act. Next, the Article examines alternate interpretations of Section 2(b) and other provisions contained throughout the GPL that may attempt to extend the copyleft restrictions beyond the scope of a traditional derivative works analysis. Finally, the Article considers these provisions in light of recent Ninth Circuit cases examining the intersection of contract law and intellectual property licensing. The Article concludes that

---

[4] All references to the GPL are to GPL version 2, because Linux is licensed under this version, and is therefore the most popular version of the license.

[5] Sapna Kumar, *Enforcing the GNU GPL*, 2006 U. ILL. J.L. TECH. & POL'Y 1, 10 (2006).

[6] *See generally* Robert W. Gomulkiewicz, *De-Bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75, 83-92 (2002).

[7] *See, e.g.*, Jeremy Andrews, *Linux: The GPL and Binary Modules*, KERNEL TRAP, (Dec. 5, 2003, 7:14 AM), http://kerneltrap.org/node/1735.

under recent precedent, software modules linked to the Linux kernel are freely licensable because there is no remedy for a licensee's failure to follow the GPL's terms.

I. OBLIGATIONS UNDER THE GPL

The GPL is commonly known as a "strong copyleft" license—meaning that any derivative work created from a GPL-licensed code, no matter how insignificant the contribution, must also be licensed under the same terms of the GPL license.[8] However, ambiguities in the language of GPL Section 2 give rise to multiple possible interpretations of how far this copyleft provision reaches.

Under a "copyleft" license, "downstream licensees, no matter how far removed from the original licensor, are [] bound by the key GPL terms,"[9] including the requirement to license any derivative work at no charge to third parties. Section 2 of the GPL is one of the key provisions implementing these copyleft requirements. Section 2 states:

> You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions . . . b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. . . .[10]

It is uncertain how far the obligations of this provision actually reach. The first sentence of Section 2, quoted above, allows for

---

[8] John Tsai, *For Better or Worse: Introducing the GNU General Public License Version 3*, 23 BERKELEY TECH. L.J. 547, 551 (2008).

[9] *Id.*

[10] Free Software Foundation, *GNU General Public License Version* 2, GNU OPERATING SYSTEM, http://www.gnu.org/licenses/gpl-2.0.html [hereinafter GPL v2].Section 3 further requires that the licensee provide access to the source code of the distributed program. GPL v2, Section 3.

modifications that would form a "work based on the Program." However, Section 2(b) requires that the GPL be extended to "any [distributed] work . . . that in whole or in part contains or is derived from the Program or any part thereof." Furthermore, the subsequent sentences of Section 2 use various other terms to describe the result of modifications, including "modified files," "modified program," and "modified work."[11] This use of disparate terms clouds the true effect of the provision.

The first step in untangling Section 2 of the GPL is to understand the scope of a "work based on the Program." Section 0 of the GPL states that "a 'work based on the Program' means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."[12] A common interpretation of this language is that the term "work based on the Program" is directly linked to the concept of derivative works under copyright law and therefore equivalent in scope.[13] Advocates of this interpretation point to the first clause of the sentence, which limits the definition to "the Program or any derivative work under copyright law."[14] The second clause, which is arguably broader, would then simply be an "interpretive explanation . . . [which] gives an indication of what the GPL drafters thought, hoped, or may argue in a dispute is the meaning of the term 'derivative works' under copyright law."[15] Thus, because the definition directly incorporates and hinges upon an existing, statutorily-defined legal concept, any subsequent elaboration is not sufficient to alter this concept and can be viewed as a statement

---

[11] GPL v2, Section 2. *See also* Lothar Determann, *Dangerous Liasons – Software Combinations as Derivative Works? Distribution, Installation, and Execution of Linked Programs under Copyright Law, Commercial Licenses, and the GPL*, 21 BERKELEY TECH. L.J. 1421, 1487-88 (2006).

[12] GPL v2, Section 0.

[13] Gomulkiewicz, *supra* note 6, at 89. *See also* Michael F. Morgan, *The Cathedral and the Bizarre: An Examination of the "Viral" Aspects of the GPL*, 27 J. MARSHALL J. COMPUTER & INFO. L. 349, 390 (2010) (stating that "the GPL.v3 seems to make it clear that certain terminology used in the GPL is intended to have the same scope as the term 'derivative work' under copyright law" and pointing to the GPL.v3 definition of the term "modify").

[14] GPL v2, Section 0.

[15] Determann, *supra* note 11, at 1487.

of opinion.[16] Therefore, a "work based on the Program" would mean a derivative work as defined by the Copyright Act.

The second step in understanding Section 2 is to note that the conditions set forth in 2(a), (b), and (c) ("the lettered conditions") "apply to the modified work as a whole."[17] Here, the "modified work as a whole" appears to refer to the "work based on the Program" authorized by the first sentence of Section 2.[18] If this phrase is read to be limiting, then references within the lettered conditions to a "modified file," "modified program," or "work that in whole or in part contains or is derived from the Program" can be interpreted as equivalent in scope to the defined term "work based on the Program."[19] Under this interpretation, the copyleft requirements of the lettered conditions would only extend to works that qualify as derivative works under the Copyright Act.

An alternative interpretation of Section 2 of the GPL elevates the importance of the plain meaning of the provisions. For example, the reference in Section 2(b) to a "work that in whole or in part contains . . . the Program," could be construed as including any work that incorporates code from the Program, no matter how insignificant and with no regard to whether the included code would be protectable under the Copyright Act.[20] Thus, "Section 2(b)'s license condition may apply to programs derived from minuscule amounts of code or non-copyrightable code that would not otherwise make the host program a derivative work according to copyright law."[21]

---

[16] *See, e.g.*, Morgan, *supra* note 13, at 394 ("Accordingly, to the extent that the GPL v2 suggests that the copying of any subject matter from 'the Program' necessarily makes a subsequent work a derivative work, that statement is incorrect.")

[17] GPL v2, Section 2.

[18] The second full paragraph of Section 2 later uses the term "work based on the Program" as a direct substitute for the original "modified work as a whole." This direct substitution lends credence to the theory that the requirements of Sections 2(a), (b), and (c) apply to any "work based on the Program."

[19] This implication is fair if the first sentence of the second full paragraph of Section 2, "These requirements . . .," is read as limiting the scope of the lettered conditions only to "modified works." An alternative interpretation of this sentence is that it is non-limiting in the sense that it is simply identifying one category of application out of several.

[20] GPL v2, Section 2; Gomulkiewicz, *supra* note 6, at 90.

[21] Gomulkiewicz, *supra* note 6, at 90. Further textual argument for this

Section II of this Article considers the applicability of the GPL to loadable kernel modules under the first, narrower interpretation discussed above. Section III discusses the second, broader interpretation of the GPL and the effect of recent developments on its possible application to loadable kernel modules.

II. APPLICABILITY OF THE GPL TO LOADABLE KERNEL MODULES UNDER A DERIVATIVE WORKS ANALYSIS

As discussed above, the GPL attempts to restrict non-GPL software from linking to GPL-licensed programs by asserting the copyright holder's exclusive right to prepare derivative works.[22] This section first provides a brief description of courts' changing attitudes regarding the level of protection afforded to software programs under the Copyright Act, particularly emphasizing the scope of derivative work rights. Next, loadable kernel modules are introduced and analyzed to determine whether they qualify as derivative works.

A. The Evolution of the Derivative Works Test Applied to Software

Over time, the protections afforded software programs and their associated derivative work[23] rights have decreased as courts have better understood the idea-expression dichotomy as applied to software. In particular, as discussed below, courts have expressed willingness to allow copying of software interfaces for purposes of interoperability.

Early court decisions dealing with inter-changeable or inter-operable media, such as *Midway Mfg. Co. v. Artic International*,

---

interpretation contrasts the use of "*the* modified files," "*the* modified program," and "*the* modified work as a whole" with "*any* work that you distribute or publish." *See* GPL v2, Section 2 (emphasis added).

[22] GPL v2, Sections 0, 2 and 3.

[23] A derivative work is defined within the Copyright Act as "a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted." 17 U.S.C. § 101 (2006).

Inc.,[24] and *Worlds of Wonder, Inc. v. Veritel Learning Systems, Inc*,[25] "looked at the output of the combination of [the original and the follow-on] works rather than at the works themselves."[26] Under such a broad definition, "courts would struggle to find any add-on components or software that did not create an infringing derivative."[27]

The high point of copyright protection for software was the Third Circuit's decision in *Whelan Assoc., Inc. v. Jaslow Dental Lab, Inc.*[28] In *Whelan*, the plaintiff had provided defendant Jaslow Dental Laboratory with software designed to aid in management of defendant's dental office.[29] After Jaslow developed its own similar office software based upon internal knowledge of Whelan's program, Whelan alleged that the new program infringed the copyright to the original program.[30] Although there were substantial "differences in programming style, in programming structure, in algorithms and data structures," the two programs shared significant "overall structural similarities."[31] The Third Circuit looked beyond the absence of literal, verbatim copying of the source code and instead, by analogy to a literary work, relied upon substantial similarities contained within the structure, sequence, and organization – the "SSO."[32] However, the Third Circuit went even further by suggesting that "the

---

[24] Midway MFG. Co., v. Artic Int'l, Inc., 704 F.2d 1009, 1013-14 (7th Cir. 1983).

[25] Worlds of Wonder, Inc. v. Veritel Learning Sys, Inc., 658 F. Supp. 351 (N.D. Tex. 1986).

[26] Douglas A. Hass, *A Gentlemen's Agreement: Assessing the GNU General Public License and Its Adaptation to Linux*, 6 CHI.-KENT J. INTELL. PROP. 213, 257 (2007).

[27] *Id.*

[28] Whelan Assoc., Inc. v. Jaslow Dental Lab, Inc., 797 F.2d 1222 (3d Cir. 1986).

[29] *Id.* at 1225-27.

[30] *Id.*

[31] *Id.* at 1228.

[32] *Id.* at 1234. ("The copyrights of other literary works can be infringed even when there is no substantial similarity between the works' literal elements. One can violate the copyright of a play or book by copying its plot or plot devices. . . . By analogy to other literary works, it would thus appear that the copyrights of computer programs can be infringed even absent copying of the literal elements of the program.")

sole idea of a computer program is the purpose the program seeks to achieve. In *Whelan*, the purpose was "to aid in the business operations of a dental laboratory." According to the Third Circuit, anything more specific in the program would be considered protectable expression. This approach is quite sweeping in the amount of protection it grants and various commentators have criticized the decision for providing overbroad protection to software.[33]

In the early 1990s, courts began to reduce the scope of copyright protection afforded computer software. As illustrated by the Second Circuit's 1992 decision in *Computer Associates International, Inc. v. Altai, Inc.*,[34] courts began to use more sophisticated analysis, expanding on the idea-expression dichotomy. In *Altai*, the Second Circuit adopted an "abstraction, filtration, comparison" test.[35] Initially, the court divided the program into component parts based upon increasing levels of abstraction. Then, the court filtered out those portions of the software which were unprotectable at each level. In performing the filtering step, the court removed from protection those segments of the code which were a merger of expression and ideas, using "scenes a faire" analysis, as well as those segments of code which were dictated by efficiency concerns.[36] Only after this level of analysis was complete for each level of abstraction did the court compare the two works to determine if, given protectable expression, enough substantial similarity existed to warrant a finding that infringement had occurred. Thus, the *Altai* court removed a significant portion of the protection granted by *Whelen* by denying "protection to specific elements of programs [such as] purely functional features, features dictated by efficiency, and features necessary for compatibility with other programs."[37]

Building on *Altai*, the Ninth Circuit further honed the

---

[33] *See, e.g.*, Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1082-82 (1989).

[34] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, (2d Cir. 1992).

[35] *Id.* at 706

[36] *Id.* at 706-09. *See also* David C. Tunick, *How to Avoid Infringing the Copyright of a Computer Program: From the Perspective of a Computer Programmer Turned Attorney/Law Professor*, 4 J. INTELL. PROP. L. 49, 56-60 (1996).

[37] Hass, *supra* note 26, at 261.

"abstraction, filtration, comparison" test in deciding two video game cases. In *Sega Enterprises, Ltd. V. Accolade, Inc.*,[38] the court ruled that despite Accolade's reverse engineering of Sega's game console software, its use of only those portions of Sega's software necessary to make its games interoperate with the console was a fair use privileged under § 107 of the Copyright Act.[39] The court stated that "[i]n some circumstances, even the exact set of commands used by the programmer is deemed functional rather than creative for purposes of copyright."[40] Further, "when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement."[41] The court ruled that the *Altai* test, when applied to the facts before it, allowed wholesale copying (during reverse engineering) of the console software to the extent necessary to determine which elements of the code which were not protected expression.[42] Furthermore, the court explicitly noted that "the functional requirements for compatibility with the Genesis console . . . are not protected by copyright."[43]

In *Sony Computer Entertainment v. Connectix Corp.*, the Ninth Circuit again ruled reverse engineering to be a fair use.[44] In fact, the court allowed copying of Sony's code not just for the creation of games that interoperate with the plaintiff's game console, but for the creation of software which would at times replace the plaintiff's console software and enable Sony Playstation compatible games to be played on a PC.[45] Again, they ruled that wholesale copying is acceptable when necessary to locate unprotected elements of a software program.[46] In justifying this decision they found not just those elements necessary for interoperability unprotected, but all

---

[38] Sega Enters. LTD. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

[39] *Id.* at 1527.

[40] *Id.* at 1524.

[41] *Id.* at 1524 (quoting National Commission on New Technological Uses of Copyrighted Works, Final Report 1 (1979)) (internal quotations omitted).

[42] *Id.* at 1527.

[43] *Id.* at 1522.

[44] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000).

[45] *Id.* at 608.

[46] *Id.*

"functional elements."[47]

Thus, as courts have become increasingly familiar with computer software, their unmistakable trend is to reduce the scope of protection granted to program code. In particular, as evidenced by the *Altai* filtration step and specific language from both the *Sega* and *Connectix* opinions, code specifically required for interoperability between programs has been explicitly identified as unprotectable, functional code necessitated by efficiency. Thus, courts "have become increasingly solicitous of parties who copy only interfaces of copyrighted software, where the purpose of doing so is to achieve interoperability."[48]

B. Applicability of the Modern Derivative Works Test to Loadable Kernel Modules

To better understand the application of the GPL to loadable kernel modules, a cursory knowledge of the purpose and structure of loadable kernel modules is necessary. The Linux kernel is the core section of Linux code: it is the heart of the operating system and is responsible for allocating system resources such as power, memory, or network connectivity.[49] Loadable kernel modules, on the other hand, are independently developed pieces of code that can be "loaded" into the kernel at runtime (a process also known as "dynamic linking")[50] and that often add new functionality or capabilities.[51] A common example of a loadable kernel module is a device driver, which allows for communication between the kernel

---

[47] *Id.* at 599.

[48] Sean Hogle, *Unauthorized Derivative Source Code*, 18.5 COMPUTER & INTERNET LAW. 1, 6 (2001).

[49] RUBINI, *supra* note 3, at Chapter 1.

[50] This paper deals almost exclusively with the case of dynamically linked kernel modules. For more in depth analysis of the legal ramifications of dynamic vs. static linking of modules see Mitchell Stoltz, *The Penguin Paradox: How the Scope of Derivative Works in Copyright Affects the Effectiveness of the GNU GPL*, 85 B.U. L. REV. 1439 (2005); Tsai, *supra* note 8; and Morgan, *supra* note 13. These references conclude that static linking of a module into the kernel code almost certainly creates a derivative work. They offer differing conclusions with regard to dynamic linking.

[51] RUBINI, *supra* note 3, at Chapter 1.

and a specific piece of hardware.[52] Dynamic linking of kernel modules allows "the original program and the module [to] occupy two separate object code files that can be sold and distributed separately."[53]

Due to their ability to be dynamically linked, loadable kernel modules represent a unique class of software somewhere between the kernel itself and standalone applications. The module resembles an extension of the kernel in the sense that it performs operating system-like functions and communicates with the kernel using the kernel's own internal communication structure.[54] However, a loadable module also contains similarities to standalone applications. Module code is never actually combined with the kernel code, but instead uses a system of interfaces to allow intercommunication between the various active components.[55]

A ruling that standalone applications were derivative works would mean the demise of an entire industry: it is common practice for proprietary applications to run on many different operating systems, including Linux.[56] Fortuitously for application developers, current (though perhaps not pre-*Altai*) decisions have clearly held that the use of software elements necessary for interoperability are unprotected expression.[57] Assuming these elements are the only ones

---

[52] *Id.*

[53] Stoltz, *supra* note 50, at 1449. Later, using a specified module interface, the module can be inserted by reference into the kernel proper and await later invocation of the module's functions. It is important to note, however, that the module code is not literally inserted into the kernel code; Instead, a reference to the module's location within the computer's memory is inserted into the kernel code. Then, when the module functionality is required, the kernel will communicate with the module at the referenced location. After the module functionality is no longer desirable, the module can be unloaded and any references to the module contained within the kernel are eliminated.

[54] RUBINI, *supra* note 3, at Chapter 2.

[55] Hass, *supra* note 26, at 254-55.

[56] *Id.* at 251.

[57] See discussion of *Altai*, *Sega*, and *Connectix* in Section II.A, *supra*; *but see* Edward J. Naughton, *Bionic Revisited: What the Summary Judgment Ruling in* Oracle v. Google *Means for Android and the GPL*, BROWN RUDNICK ALERT, 5-8 (Nov. 2011), *available at* http://www.brownrudnick.com/nr/pdf/alerts/Brown_Rudnick_Bionic_Revisited_Naughton_11-11.pdf (pointing to recent developments in *Oracle v. Google* and arguing that inline functions and variables

borrowed from an operating system in the creation of an application, standalone applications are not derivative works.

With this in mind, distinguishing between standalone applications and kernel modules is arguably a matter of degree and not kind. For instance, a Windows version of Adobe Photoshop cannot run on the Windows operating system without using the Windows-specific system call interface. In the same fashion, the device driver for a video card in a Windows PC cannot communicate with the operating system without using Windows-specific driver interfaces. Thus, both the standalone application and the device driver module are independent pieces of code designed to interact with a specific operating system using specified interface code.

Depending on the desired function and design of a kernel module, the interface between the module and the kernel can range from simple to highly complex and incorporate a significant amount of functional code.[58] Under the logic of *Altai*, *Sega*, and *Connectix*, however, it does not matter how much of the functional interface code a module contains because this code is inherently unprotectable under the Copyright Act. For instance, after noting that certain works more closely track the core intent of the Copyright Act than others, the *Connectix* court stated that "Sony's BIOS [software] lies at a distance from the core because it contains unprotected aspects . . . [w]e consequently accord it a 'lower degree of protection than more traditional literary works.'"[59] Further, the *Sega* court noted that "[u]nder a test that breaks down a computer program into its component subroutines and sub-subroutines and then identifies the idea or core functional element of each . . . many aspects of the program are not protected by copyright."[60] Because these courts

---

cannot be deemed *per se* uncopyrightable and instead must be subjected to a line-by-line analysis.)

[58] *See, e.g.*, Hass, *supra* note 26, at 265 (discussing Linus Torvald's comments on the stability of the Linux API and the changing scope of module functionality) and *id.*, at 255 (discussing a driver facilitating communication between the kernel and a high-speed data networking card, by "copy[ing] required data structures and other function names" from the kernel).

[59] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 603 (9th Cir. 2000) (quoting Sega Enters. LTD. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992)).

[60] *Sega*, 977 F.2d at 1525.

specifically identified functional requirements for compatibility as unprotected,[61] use of such unprotectable code should never, by itself, lead to a finding of infringement upon an exclusive right of a copyright holder.

It remains to be seen to what extent courts will be willing to allow copying for the sake of interoperability. For instance, a module that "pervasively incorporates" the underlying data structures or internal communication processes of the kernel may be found to be a derivative work, either because some of the code will be deemed protectable expression or because the module copies non-literal elements of the kernel, such as the structure, sequence, or organization, which are protected under copyright.[62] However, assuming Linux kernel modules only contain the source code or headers necessary to enable efficient interoperability of proprietary code with the Linux Kernel, the modules fall squarely within the protections elucidated by *Altai*, *Sega* and *Connectix* for successful avoidance of classification as a derivative work.

Thus, under the interpretation of Section 2(b) of the GPL set forth above in Section II of this Article, the requirements of the GPL will only extend to those loadable kernel modules that would qualify as derivative works under the Copyright Act. Modules that only incorporate unprotected, functional code necessary for interoperability do not trigger the requirements of the GPL.[63]

III. APPLICABILITY OF THE GPL BEYOND A DERIVATIVE WORKS ANALYSIS

A. Alternative Interpretations of the GPL

Unfortunately for software developers hoping to create

---

[61] *See id.* at 1522; *Connectix*, 203 F.3d at 603.

[62] Hogle, *supra* note 48, at 5. *See also* Naughton, *supra* note 57, at 8-9 (arguing that Google's attempt to "clean" the GNU C library ("glibc") of copyright protectable material when creating the Android Bionic library failed, in part, because Google did not consider the copyright covering "the overall structure of the API").

[63] This Article will further discuss the implication that this finding has upon various modes of distribution in Section III, *infra*.

proprietary modules that interact with the Linux kernel, certain provisions of the GPL might be interpreted to reach beyond a straightforward derivative works analysis. As discussed in Section II of this Article, *supra*, some commentators have pointed to the plain language of Section 2(b) of the GPL, which requires the GPL to be applied to any work "that in whole or in part contains or is derived from the Program or any part thereof."[64] Thus, reading this phrase literally, a module could be an independent, non-derivative work under copyright law, but still required to be released under the GPL by the express terms of the agreement because it contains "a part" of the Program. Furthermore, later provisions of Section 2 also purport to extend control beyond that of copyright law. In particular, after setting forth the lettered conditions, Section 2 states:

> These requirements [the lettered conditions] apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
>
> Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.[65]

At first blush this portion of the GPL ("the collective works

---

[64] GPL v2, Section 2(b); *see e.g.* Tsai, *supra* note 8, at 555-56.

[65] GPL v2, Section 2. Section 2 subsequently states: "In addition, mere aggregation of another work not based on the Program with the Program (or a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License."

provision") appears to track copyright law by applying the obligations of the GPL only to derivative works, but it introduces ambiguity by attempting to apply the GPL to non-derivative works which are "reasonably considered independent" if they are distributed "as part of a whole which is a work based on the Program."[66] As discussed above, the definition of the term "work based on the Program" is open to interpretation and as such could be equal to or broader in scope than the concept of derivative works under the Copyright Act.

The GPL further muddies the water by suggesting, through an interpretive gloss, intent to control the distribution of "collective works based on the Program."[67] Once again, the extent of the GPL's reach is unclear due to the combination of statutorily defined terminology[68] with idiosyncratically worded concepts such as a "work based on the Program." However, the net effect of the collective works provision appears to be an attempt extend the GPL's reach beyond the program and its derivative works to any "collective work based on the program" which contains a *modified* GPL-covered program in addition to any number of independent, non-derivative sections, if those independent sections can be considered part of a "modified work as a whole."[69]

In the following sections, this Article discusses the application and possible effects of these ambiguities on various factual scenarios. In particular, each of the provisions introduced above will be assessed in light of several recent Ninth Circuit cases that analyze the intersection of copyright law and contract law. This distinction between contract and copyright law is critical because, in the event the GPL is interpreted as a contract, the remedy for breach, absent a provision enabling injunctive relief, is likely limited to monetary

---

[66] *Id.*

[67] *Id.*

[68] *See* 17 U.S.C § 101 (2010) ("A 'collective work' is a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole").

[69] Section 2 of the GPL also notes that "mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License."

damages.

B. Recent Decisions on the Intersection of Copyright Law and Contract Law

Several recent Ninth Circuit cases have analyzed the license-versus-sale dichotomy and expounded upon the interaction between contract and licensing law. Although these cases are all interpretations of the first-sale doctrine, they have implications on how the Ninth Circuit will enforce software license agreements. The trio of cases—*Vernor v. Autodesk, Inc.*,[70] *UMG Recordings, Inc., v. Augusto*,[71] and *MDY Industries, LLC, v. Blizzard Entertainment, Inc.*[72]—set limits on the availability of copyright infringement actions as a remedy for non-compliance with an agreement, whether styled as a contract or a license.

In *Vernor*, an eBay vendor brought an action seeking a declaratory judgment that he had the legal right to resell copies of Autodesk's software packages.[73] Autodesk claimed that the agreement that accompanied the software (the "software license agreement" or "SLA") was, in fact, a license to use the software under specific conditions, one of which forbade the resale of the software.[74] Vernor alleged that the software had been sold to its first owner, rather than licensed, and therefore the first sale doctrine applied.[75] The *Vernor* court, in holding that Autodesk was entitled to an injunction halting re-sale of its software online, stated:

> [A] software user is a licensee rather than an owner of a copy where the copyright owner (1) specifies that the user is granted a license; (2) significantly restricts the user's ability to transfer the software; and (3) imposes notable use restrictions. Applying our holding to Autodesk's SLA, we conclude that CTA [the initial transferee of the Autodesk software] was a

---

[70] Vernor v. Autodesk, Inc., 621 F.3d 1102 (9th Cir. 2010).
[71] UMG Recordings, Inc. v. Augusto, 628 F.3d 1175 (9th Cir. 2011).
[72] MDY Indus., LLC v. Blizzard Entm't, Inc., 629 F.3d 928 (9th Cir. 2010).
[73] *Vernor*, 621 F.3d at 1104-06.
[74] *Id.*
[75] *Id.*

licensee rather than an owner. . . .[76]

In *UMG*, the court refused to find that a statement on the label of an unsolicited, promotional CD delivered via mail constituted a license.[77] Even though the statements on the CDs purported to create a license, the unsolicited nature of the mailing, coupled with the lack of any affirmative statement or actions denoting acceptance made the existence of a license problematic.[78] In the absence of a license, the distribution was ruled a "first sale" immunizing the defendant from UMG's claim of copyright infringement.[79]

In *MDY*, plaintiff MDY Industries sought a declaratory judgment that sales of its software, a type of "bot" called "Glider," did not infringe Blizzard's copyright in its popular "World of Warcraft" online multi-player game.[80] Applying *Vernor*, the *MDY* court found that the End User License Agreement ("EULA"), together with the Terms of Use ("ToU"), constituted a license because Blizzard "reserves title in the software and grants players a non-exclusive, limited license. Blizzard also imposes transfer restrictions if a player seeks to transfer the license..."[81] However, the Ninth Circuit ruled that use of the Glider software, which automated play within some levels of Blizzard's game, did not infringe the online game's copyright even though it violated the ToU of the game.[82]

In coming to this conclusion, the Ninth Circuit engaged in a more detailed and nuanced[83] analysis of the exact provisions of the ToU at

---

[76] *Id.* at 1111.

[77] UMG Recordings, Inc. v. Augusto, 628 F.3d 1175, 1180 (9th Cir. 2011).

[78] *Id.* ("Our conclusion that the recipients acquired ownership of the CDs is based largely on the nature of UMG's distribution. First, the promotional CDs are dispatched to the recipients without any prior arrangement as to those particular copies. The CDs are not numbered, and no attempt is made to keep track of where particular copies are or what use is made of them. As explained in greater detail below, although UMG places written restrictions in the labels of the CDs, it has not established that the restrictions on the CDs create a license agreement.").

[79] *Id.* at 1180-81.

[80] MDY Indus., LLC v. Blizzard Entm't, Inc., 629 F.3d 928, 934-35 (9th Cir. 2010).

[81] *Id.* at 938.

[82] *Id.* at 941-42.

[83] *See* Nancy S. Kim, *The Software Licensing Dilemma*, 2008 B.Y.U. L. REV. 1103, 1003-04 (2008) ("[S]oftware transactions are not a binary proposition. While some transactions can clearly be identified as either licensing or sales deals, most

issue to determine whether they constituted conditions on the copyright license or were purely contractual in nature. In this context, the court stated that "contractual terms that limit a license's scope [are] 'conditions,' the breach of which constitute copyright infringement."[84] The court referred "to all other license terms as 'covenants,' the breach of which is actionable only under contract law."[85] Applying this distinction between conditions and covenants to the provisions at issue, the court determined that the prohibition on the use of automated "bots" was a covenant, rather than a condition.[86] Therefore, the use of bots in violation of the ToU was simply a breach of contract and did not rise to the level of copyright infringement.

As justification for this conclusion, the court provided the following policy views:

> Were we to hold otherwise, Blizzard—or any software copyright holder—could designate any disfavored conduct during software use as copyright infringement, by purporting to condition the license on the player's abstention from the disfavored conduct. . . . This would allow software copyright owners far greater rights than Congress has generally conferred on copyright owners.[87]

In concluding its analysis of conditions and covenants, the Ninth Circuit held that "for a licensee's violation of a contract to constitute copyright infringement, there must be a *nexus* between the condition and the licensor's exclusive rights of copyright."[88] In other words, "the potential for infringement exists only where the licensee's action (1) exceeds the license's scope (2) in a manner that implicates one of the licensor's exclusive statutory rights."[89]

---

entail both.")

[84] *MDY*, 629 F.3d at 939.

[85] *Id*.

[86] *Id.* at 939-40.

[87] *Id.* at 941.

[88] *Id.* (emphasis added).

[89] *Id.* at 940. The *MDY* court also used the phrasing "the copyright owner's complaint must be grounded in an exclusive right of copyright" in place of "in a manner that implicates one of the licensor's exclusive statutory rights." *Id.*

Together, *Vernor, UMG,* and *MDY* show an increasing focus on the proper balance between copyright law and contract law. In particular, the policy discussion in *MDY* exhibits a firm recognition that licensing agreements present an opportunity for the misuse and unsanctioned extension of copyright rights.[90] As such, the *Vernor* court refined the legal test for distinguishing between a license and a contract for sale. Further, even after determining that the agreement contained a copyright license, the *MDY* court created an explicit test for determining whether a particular provision in a license agreement exceeds the scope of rights that Congress sought to confer upon copyright owners and therefore should be regarded solely as a contractual covenant.

The remainder of this Article applies these new legal tests to the GPL. In particular, several of the GPL's most debated terms are applied to different scenarios and interpreted in light of the *MDY* trio of cases. Under these cases, it is possible that either: 1) the GPL may be found to be a contract and not a license, or 2) the provisions of the GPL which necessitate the disclosure of non-derivative source code to downstream recipients may be construed as contractual covenants. If true, in neither case would a remedy of copyright infringement be forthcoming.

C. General Public License or General Public Contract?

Arguably, the GPL is not a license agreement at all, despite its internal protestations to the contrary.[91] In *Vernor,* to qualify an agreement as a license, the court required that the copyright owner (1) specify that a user is granted a license, (2) include significant restrictions on the transfer of the software, and (3) include notable use restrictions.[92] The GPL refers to itself as a license several times and states clearly that the recipient of the software is only being granted a

---

[90] *See* Justin Van Etten, *Copyright Enforcement of Non-Copyright Terms:* MDY v. Blizzard; Krause v. Titleserv, 2011 DUKE L. & TECH. REV. 7, 42 (2011) ("The Ninth Circuit, in *MDY*, has explicitly created a rule against rightsholders using copyright to enforce non-copyright terms, and has based this rule in the unequivocal policy arguments that copyright should not be expanded by contract.").

[91] *See, e.g.*, GPL v2, Section 4.

[92] Vernor v. Autodesk, Inc., 621 F.3d 1102, 1110-11 (9th Cir. 2010).

license. Section 1 allows for copying and distribution of "verbatim copies" of the Program under several minor conditions: namely, the software must include a copyright notice, keep intact all notices that refer to the GPL, and provide any recipients of the software with a copy of the GPL.[93] Section 3 provides additional terms required in order to distribute the Program in object code. One could also view the "copyleft" requirements of Section 2 as a form of restriction on the transfer of the software in the sense that distribution of a work based on the Program is only permitted if it is also licensed under the terms of the GPL. The GPL appears to satisfy at least the first two requirements of the three-part *Vernor* test. [94]

However, the GPL does not appear to include any notable use restrictions.[95] Use of software licensed under the GPL is not restricted to the extent it does not involve distribution: "The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program)."[96] In addition, neither copying nor modification carries any additional restrictions absent subsequent distribution. Thus, while restrictions are placed upon the creation of derivative works and the distribution of copies of the work, there do not appear to be any restrictions regarding the actual "use" of the program. Absent provisions restricting use, a strict literal reading of the *Vernor* decision implies the GPL is not a license.

However, given the unique "copyleft" requirements of the GPL

---

[93] These conditions may not rise to the level of "significant" restrictions, however.

[94] One can argue, however, that copying and distribution of "verbatim *copies*" under Section 1 or 3 is not the same, in a strict legal sense, as transferring *the* licensed copy of the program. In this sense, *Vernor*'s requirement of "significant restrictions on the transfer of the software" could be interpreted to require restrictions on the particular, primary copy that is the original subject of the license. 621 F.3d at 1110-11. Furthermore, the copyleft provisions are referring to distribution of a modified work rather than transfer of the particular copy that is the primary subject of the license.

[95] *See, e.g.*, Eben Moglen, *Enforcing the GNU GPL*, FREE SOFTWARE FOUNDATION, Sept. 10, 2001, http://www.gnu.org/philosophy/enforcing-gpl.html ("The license does not require anyone to accept it in order to acquire, install, use, inspect, or even experimentally modify GPL'd software.").

[96] GPL v2, Section 0.

and the ambiguous, non-statutory nature of the term "use," a court may construe other GPL provisions as providing restrictions on use. In fact, the *Vernor* court itself noted that "the SLA [Software License Agreement] also imposed use restrictions against . . . modifying, translating, or . . . removing any proprietary marks from the software or documentation."[97] While a prohibition against modification or translation is better classified as a restriction on derivative work rights, the *Vernor* court explicitly recognized these as restrictions on "use" in fulfillment of the third requirement.[98] With this in mind, it appears likely that the GPL would be interpreted as a license under a *Vernor*-styled framework due to the restrictions discussed above.[99]

Even if the GPL is found to be a license, however, certain provisions, when scrutinized under the *MDY* test, are likely to be found to be contractual covenants rather than conditions upon the license grant.

D. Alternative Interpretations of the GPL Applied in Light of MDY

1. Distribution of a Loadable Kernel Module Standing Alone

The implications of the *MDY* case for GPL's applicability to loadable kernel modules are profound. As stated by the Ninth Circuit in *MDY*, a finding of copyright infringement will only follow if there is a nexus between the provision contained within the agreement and the licensor's exclusive rights of copyright.[100] Assuming loadable

---

[97] *Vernor*, 621 F.3d at 1111.

[98] In addition, Autodesk's restriction on "removing any proprietary marks from the software," *id.*, may be analogous to the GPL's requirement to maintain copyright notices and provide a copy of the license to any downstream licensee.

[99] It is also possible that a court would interpret the GPL as ineffective in light of *UMG*. Similar to the "license" denied in *UMG*, the GPL does not provide for any affirmative interaction between the putative licensor and licensee. For more discussion of whether the GPL's notice and language are sufficient to form a binding contract under traditional "offer and acceptance" doctrine, *see* Kumar, *supra* note 5, at 16-19; Margaret Jane Radin, *Humans, Computers, and Binding Commitment*, 75 IND. L.J. 1125, 1132-33 (2000); Christian H. Nadan, *Open Source Licensing: Virus or Virtue?*, 10 TEX. INTELL. PROP. L.J. 349, 362-63 (2002).

[100] MDY Indus., LLC v. Blizzard Entm't, Inc., 629 F.3d 928, 941 (9th Cir.

kernel modules are not derivative works of the Linux kernel,[101] distribution of such modules under a license other than the GPL or without the corresponding source code is at worst breach of a covenant within the GPL, not meriting a finding of copyright infringement.

Under a narrow interpretation of the GPL, discussed in Section II, *supra*, a "work based upon the Program" is equivalent in scope to the concept of derivative works under the Copyright Act. If this is the case, then the requirements of the lettered conditions of Section 2 and of the collective works provision only apply to modified works that would qualify as derivative works. Assuming that loadable kernel modules containing only unprotected, functional code are not derivative works of the Linux kernel, the requirements of the GPL do not extend to these modules in any fashion. Thus, any entity that holds a copyright on the kernel would not have any claim, grounded in either contract or copyright law, that the terms of the GPL were breached or the copyright infringed.

Under the broader interpretation of the GPL discussed in Section III.A, *supra*, the term "work based on the Program" incorporates any work that contains any portion of the Program, no matter how insignificant. In this case, the lettered conditions of Section 2 and the collective works provision reach beyond a derivative works analysis and require compliance with the GPL for programs containing any portion of the code.[102] However, non-compliance with these conditions would not trigger a finding of copyright infringement because distribution of a *non-derivative* work, standing alone, does not implicate, or have a nexus with, any of the copyright holder's exclusive rights any more than distribution of a completely unrelated work does.[103] Thus, under an *MDY* analysis, even the broader reading of the GPL results only in a breach of contract action for distribution

---

2010).

[101] *See supra* Section II.B.

[102] This is assuming that the plaintiff copyright holder would be able to satisfy a preliminary showing that a contract was, in fact, formed under Section 5 of the GPL.

[103] *See, e.g.*, Nadan, *supra* note 99, at 369 ("The copyleft provision [of the GPL] purports to infect independent, separate works that are not derivative [works] . . . . Attempting to extract such rights exceeds the scope of the copyright.").

of a loadable kernel module that is not a derivative work. The implications of this finding regarding availability of remedies are discussed in Section III.E, *infra*.

2. Distribution of a Loadable Kernel Module in Conjunction with an Unmodified Linux Kernel

Adding an unmodified Linux kernel into the distribution package should not change the legal conclusions reached above. Under the narrower interpretation of the GPL, the loadable kernel module is not a derivative work for the reasons set forth in Section II.B, *supra*, and the requirements of the GPL only extend to the unmodified kernel. Thus, the developer may distribute the unmodified kernel in full compliance with Section 1 of the GPL and distribute the module as he sees fit. Because neither the unmodified kernel nor the module is a derivative work, the terms of Section 2 are never triggered or implicated. Thus, under an interpretation of the GPL that hinges upon a derivative works analysis, any entity that holds a copyright on the kernel would not have any claim, grounded in either contract or copyright law, that the terms of the GPL were breached or the copyright infringed by the distribution of a loadable kernel module and an unmodified kernel.

Under the broader interpretation of the GPL, in which a "work based on the Program" includes any program containing any portion of the kernel code, the module is subject to the requirements of the lettered conditions of Section 2 and the collective work provision. However, non-compliance with these requirements still amounts to only a breach of contact claim. This is because the breach of contract claim could only be based upon failure to distribute the module in compliance with the requirements of Section 2; the unmodified kernel is in complete compliance with the distribution requirements of Section 1.[104] As discussed above in Section III.D.1, if the module is not a derivative work but still subject to the terms of the GPL, then non-compliance with the GPL will only lead to a breach of contract claim: Distribution of an independent, non-derivative work does not

[104] This analysis does not apply however, if the unmodified kernel and the module are, together, considered a single "modified work as a whole" or a collective work. *See* Section III.D.3, *infra*.

implicate, or have a nexus to, any of the copyright holder's exclusive rights. Thus, under the broader interpretation of the GPL, distribution of a loadable kernel module with an unmodified version of the Linux kernel still only exposes the distributor to a possible breach of contract claim.

3. Distribution of a Loadable Kernel Module in Conjunction with a Modified Linux Kernel

The legal results reached in the two preceding scenarios are drastically altered if the distributor chooses to also distribute a *modified* version of the Linux kernel. Under the narrower interpretation of the GPL, although the module would not qualify as either a derivative work or a "work based on the Program," the modified kernel would qualify as both. With this in mind, the requirements of the lettered conditions of Section 2 and the collective works provision would apply to the modified kernel. The legal conclusion to this scenario depends upon a reading of the collective works provision. In particular, whether the loadable kernel module and the modified Linux kernel, when distributed together, constitute a "modified work as a whole" or collective work will determine whether the result is a copyright infringement or simply a breach of contract claim.[105]

If the two programs, when distributed together, are ruled to constitute a "modified work as a whole," then the copyright has been infringed because the modified kernel is unquestionably a derivative work of the original kernel. Therefore, a provision that restricts how this derivative work may be distributed would have nexus to the exclusive distribution right of the copyright holder. As such, non-compliance with this provision would lead to a finding of copyright infringement.

If the modified kernel and the loadable module are *not* found to be a single "modified work as a whole," then the GPL has been complied with if distribution of the modified kernel accords with all

---

[105] For the Free Software Foundation's interpretation of the phrase "modified work as a whole," *see Frequently Asked Questions About Version 2 of the GNU GPL*, FREE SOFTWARE FOUNDATION, http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MereAggregation (last updated Jan. 8, 2012).

requirements of Section 2, regardless of whether the source code of the module is opened. Under the narrower interpretation of the GPL, a loadable kernel module that is not a derivative work does not implicate any provisions of the GPL because it is not a "work based on the Program." As long as the modified kernel is distributed in accordance with the GPL, the module is not implicated and there is no claim grounded in either contract or copyright law.

Analysis similar to the above also applies under the broader interpretation of the GPL, in which the copyleft requirements apply to both the module and the kernel. If the module and the kernel are ruled to be a single "modified work as a whole," then failure to provide source code for the module would likely be ruled a copyright infringement because the modified kernel is a derivative work. If the module is considered to be part of a single work with the kernel, then it too would be a derivative work and a provision regarding distribution of this derivative work would have a nexus to the copyright holder's exclusive distribution rights.

However, if the loadable module is *not* considered part of the "modified work as a whole" then failure to provide source code for the module would only give rise to a breach of contract claim. Operating under the broader interpretation of the GPL, all copyleft provisions apply to the module, regardless of whether it is a derivative work. However, because it is *not* a derivative work, distribution of this non-derivative work would not implicate, or have a nexus to, any of the copyright holder's exclusive rights, and thus noncompliance would amount solely to a breach of contract.

E. MDY*'s Effect on Availability of Remedies for Non-Compliance with the GPL*

Absent a finding of copyright infringement, the plaintiff's remedies must flow from a breach of contract claim. As the court in *MDY* implied, breach of contract remedies are generally confined to damages, and those damages are "generally limited to the value of the actual loss caused by the breach."[106] On the other hand, the remedies

---

[106] MDY Indus., LLC v. Blizzard Entm't, Inc., 629 F.3d 928, 941 n.3 (9th Cir. 2010). *See also*, Sean Hogle, *Conditions vs. Covenants: California Rulings Threaten the Practical Enforceability of Open Source Licenses*, 25.9 COMPUTER &

available to a copyright holder following a successful infringement claim are much more favorable and include lost profits or a reasonable royalty, statutory damages of as much as $150,000 (regardless of actual damages), and attorney's fees in exceptional cases.[107]

In a case where the GPL itself mandates that source code be provided for free,[108] damages for breach will be very difficult, if not impossible to ascertain, and arguably zero.[109] Unlike the situation in which a case is brought under copyright law, injunctive relief in a breach of contract case is rarely awarded, especially when it is not stipulated to within the agreement itself.[110] The GPL contains no such reference to injunctive relief as a remedy for violation of its provisions. Thus, "in the open source context, where software is licensed without charge, establishing economic loss could prove daunting if not impossible."[111] As such, even if a plaintiff is victorious on the merits of a breach of contract claim based upon non-compliance with the GPL, he may be unable to fashion any practical remedy.

INTERNET LAW. 1, 2 (2008) [hereinafter Hogle, *Conditions*]; *but see* Jose J. Gonzalez de Alaiza Cardona, *Open Source, Free Software, and Contractual Issues*, 15 TEX. INTELL. PROP. L.J. 157, 187 (2007) ("If [the GPL] is a contract, it seems that a person who refuses to comply with the terms of the GNU GPL could be forced to release the source code of his derivative work."). Dr. Gonzalez is presumably arguing that this forced "opening" could be reached under a specific performance doctrine.

[107] Hogle, *supra* note 106, *Conditions* at 2; Van Etten, *supra* note 90, at 11 (2011).

[108] GPL v2, Section 1.

[109] Kumar, *supra* note 5, at 15 ("If [contractual] consideration for the author is the release of changes back to the community, how would a court financially compensate the author under contract law? Money damages would not be an appropriate remedy . . . .").

[110] Hogle, *supra* note 106, *Conditions* at 2 ("[I]njunctive relief is facilitated by the irreparable harm presumption that applies if the plaintiff is likely to succeed on the merits of the copyright infringement claim. . . . [while i]njunctive relief is typically not available for breach of contract claim."

[111] *Id.*

CONCLUSION

Linux is a very popular operating system that is increasingly used in embedded devices. Uncertainty regarding the legal consequences of modifying proprietary software for, or simply linking proprietary software to, a device running the Linux kernel makes it difficult for developers of the proprietary software and embedded devices to reach agreement.

The reason for uncertainty is the GPL, the license to which all those using, modifying, or distributing Linux are bound. The GPL purports to require that any software derived from or linked to software licensed under it be distributed for free, with all source code included. Requiring developers to distribute proprietary software for free removes their ability to be compensated for providing their code to a third party in object form. Requiring distribution of the corresponding source code is even more damaging, because publication of the source releases trade secrets to not only the version of the code distributed under the GPL (say for a Linux version), but for the same version released under a proprietary license (e.g., for any other operating system).

Assuming Linux kernel modules only contain the source code or headers necessary to enable efficient interoperability of proprietary code with the Linux kernel, the modules fall squarely within the analysis articulated by *Altai*, *Accolade*, and *Connectix* for successful avoidance of classification as a derivative work. For the GPL to reach beyond derivative works of the Linux kernel and effectively require any software linked to it be distributed for free and with software's source code: (1) the GPL must be interpreted as a license rather than a contract, and (2) the provisions of the license which are breached must be of a type such that their breach merits a finding of copyright infringement.

According to the recent Ninth Circuit *MDY* holding, a finding of copyright infringement will only follow if there is a nexus between the conditions (of the license) and the licensor's exclusive rights of copyright. Unless the "licensee's" software is determined to be a derivative work of software licensed under the GPL, none of the exclusive rights of copyright are implicated, because though reproduction and distribution occur, they are not reproduction or distribution of anything in which the copyright holder has a legal

interest. In essence, the recent *MDY* ruling, together with conventional derivative works analysis, makes software modules linked to the Linux kernel freely licensable without regard to release of those modules' source code because there is no practical remedy for a licensee's failure to follow the terms of the GPL.