THE FEDERAL COURTS LAW REVIEW

Volume 6, Issue 1 2012

COMPUTER SOFTWARE-RELATED LITIGATION: DISCOVERY AND THE OVERLY-PROTECTIVE ORDER

© Lydia Pallas Loren[+] & Andy Johnson-Laird [++]

ABSTRACT

Litigation involving allegations of intellectual property infringement concerning computer software is some of the most complex, time consuming, and expensive litigation in which private parties engage. Certain practices in discovery, including, most significantly, the use of poorly drafted discovery agreements that also include "overly protective" orders, increase that expense dramatically. Regardless of whether the allegation is patent infringement, copyright infringement, or trade secret misappropriation, prosecuting and defending the assertions in the case require a probing analysis of the computer source code. In these types of cases, both parties will engage forensic software analysts to assist the lawyers in preparation for trial and to provide expert witness testimony for the court. The forensic software analysts will dissect the computer source code, often examining the source code of both parties, looking for signs of infringement or misappropriation as well as for technical explanations of similarities in the way the code is written or structured. But first, the computer source code must be disclosed to the opposing party. Such disclosure is almost always done pursuant to a protective order, typically stipulated to by the attorneys. Lawyers often agree to protective orders that significantly and unnecessarily increase the costs of discovery.

Attorneys should pay careful attention to the provisions addressing the requirements of production and analysis. Additionally, attorneys must understand the consequences of the clauses contained in protective orders in

---

[+] Professor Lydia Pallas Loren is the Kay Kitagawa and Andy Johnson-Laird Intellectual Property Faculty Scholar at Lewis & Clark School of Law in Portland, Oregon.

[++] Andy Johnson-Laird has 38 years of experience in the computer software industry, the last 24 of which have been as a forensic software analyst. He is the President of Johnson-Laird Inc. in Portland, Oregon. He recently served as Special Master to the Hon. M. J. Garbis, District of Maryland, to resolve several of the discovery issues described in this paper.

these types of litigation. As described in this article, it is possible to provide robust protection for disclosed source code while at the same time not unnecessarily and dramatically increasing the cost of discovery by weaponizing the protective order.

The goals of this article are three-fold. First, we seek to help lawyers understand the process of forensic software analysis. Second, we provide a set of model clauses aimed at avoiding pitfalls in the design of the discovery process, including model clauses for a protective order of appropriate scope and with appropriate protections from further disclosure of the source code that is produced. Third, for judges who are asked to intervene in discovery battles, including fights over the proper scope of a protective order, this article is meant to assist in evaluating the parties' arguments.

TABLE OF CONTENTS

I. INTRODUCTION

Litigation involving allegations of intellectual property infringement concerning computer software is some of the most complex, time consuming, and expensive litigation in which private parties engage.[1] Certain practices in discovery, including, most significantly, the use of poorly drafted discovery agreements that also include "overly-protective" orders, increase that expense dramatically. Regardless of whether the allegation is patent infringement, copyright infringement, or trade secret misappropriation, prosecuting and defending the assertions in the case require a probing analysis of the computer source code.[2] In these types of cases, both parties will engage forensic software analysts to assist the lawyers in preparation for trial and to provide expert witness testimony for the court. The forensic software analysts will dissect the computer source code, often examining the source code of both parties, looking for signs of infringement or misappropriation, as well as for technical explanations for similarities in the way the code is written or structured. But first, the computer source code must be disclosed to the opposing party. Such disclosure during discovery is almost always done pursuant to a protective order, typically stipulated to by the attorneys. Unfortunately the lawyers often agree[3] to these protective orders prior to the engagement of their

---

[1] Intellectual Property cases have litigation costs that are almost 62% higher than other types of cases. EMERY G. LEE III & THOMAS E. WILLGING, FED. JUDICIAL CTR., LITIGATION COSTS IN CIVIL CASES: MULTIVARIATE ANALYSIS (2010). *See* AM. INTELLECTUAL PROP. LAW ASS'N, REPORT OF THE ECONOMIC SURVEY 2011 (2011). In 2010, the median for the total cost of patent infringement litigation was $5 million for cases in which more than $25 million was perceived to be at stake and $2.5 million for cases in which $1 to $25 million was perceived to be at stake. *Id.* at 36, apps. 1–154. These numbers include all types of patent infringement litigation. The high discovery expenses in patent cases have been reported previously. THOMAS E. WILLGING ET AL., FED. JUDICIAL CTR., DISCOVERY AND DISCLOSURE PRACTICE, PROBLEMS, AND PROPOSALS FOR CHANGE: A CASE BASED NATIONAL SURVEY OF COUNSEL IN CLOSED FEDERAL CIVIL CASES 38–39 (1997). For comparison purposes, a recent survey of more than two thousand attorneys of record in federal civil cases terminated in the last quarter of 2008 found that the median cost of litigation was $15,000 for plaintiffs and $20,000 for defendants. Emery G. Lee III & Thomas E. Willging, *Defining the Problem of Cost in Federal Civil Litigation*, 60 DUKE L.J. 765, 769–70 (2010).

[2] Computer source code is the human-readable textual form of computer software.

[3] The rule governing protective orders encourages such agreement by requiring any party that seeks a protective order from the court to certify that they have "in good faith conferred or attempted to confer with other affected parties in an effort to resolve the dispute without court action." FED. R. CIV. P. 26(c). Additionally, because often these cases require both parties to disclose source code, it is in both parties' interests to stipulate to terms for disclosure that are beneficial. This dynamic, however, does not explain why so many protective orders are overly-protective.

experts, leading to protective orders that significantly and unnecessarily increase the costs of discovery.[4]

Protective orders in litigation requiring analysis of computer source code typically serve two primary purposes. First, these orders often contain an agreement concerning the types of files that will be disclosed, as well as the manner of their disclosure. Second, these protective orders contain many provisions that, in theory, have been designed to reduce the risk of disclosure of valuable source code. Typically the parties are extremely concerned with protecting their source code from disclosure as leaked source code can have devastating effects on a company. Protective orders used today, however, are infected with a kind of "paper thinking" that does not match the technical reality in which the forensic computer expert works. These overly protective orders contain clauses that, in a paper world, may have made some sense in achieving the underlying goal of reducing the risk of disclosure. In the paperless reality of today, however, these clauses serve no purpose except to increase the cost of litigation, a purpose that the Federal Rules of Civil Procedure expressly prohibit.[5]

We[6] hypothesize that the overly protective order is the product of either practitioners who (innocently) do not understand forensic software analysis and therefore do not appreciate what they have wrought, or practitioners who (less innocently) understand full well the nature of computer software and have devised restrictions specifically intended to be maximally inconvenient, impractical, and expensive to the opposing party. Purposefully engaging in conduct merely to increase the cost and inconvenience of discovery is expressly prohibited by the Federal Rules of Civil Procedure[7] and should not be tolerated. The lawyers involved in these cases should not be permitted to turn the discovery process itself into a tactical weapon.[8]

---

[4] The difficulty of gathering data concerning the costs of discovery in federal court litigation has been noted before. *See* LEE & WILLGING, *supra* note 1, at 770. *See also* Judith A. McKenna & Elizabeth C. Wiggins, *Empirical Research on Civil Discovery*, 39 B.C. L. REV. 785, 796–97 (1998) (discussing the methodological difficulties in studying discovery).

[5] Rule 26 requires that attorneys certify that "to the best of the person's knowledge, information, and belief formed after a reasonable inquiry" any discovery "request, response, or objection" is "not interposed for any improper purpose, such as to harass, cause unnecessary delay, or needlessly increase the cost of litigation . . . ." FED. R. CIV. P. 26(g)(1)(B)(ii).

[6] As a linguistic convenience, when we write "we" or "our" we mean either one or both of the authors.

[7] *See supra* note 5.

[8] *See* United States Court of Appeals for the Federal Circuit, *An E-Discovery Model Order* 2 (2011), *available at*

The goals of this article are three-fold. First, we seek to help lawyers understand the process of forensic software analysis. Second, we provide the tools for avoiding pitfalls in the design of the discovery process, including drafting a protective order of appropriate scope and with appropriate protections from further disclosure of the source code that is produced. Third, for judges who are asked to intervene in discovery battles, including fights over the proper scope of a protective order, this article is meant to assist in evaluating the parties' arguments.

II. GOALS AND METHODS OF FORENSIC SOFTWARE ANALYSIS

Cases alleging copyright infringement, patent infringement, or trade secret misappropriation concerning computer software require forensic software analysis. Before one can understand the constraints imposed by poorly drafted or ill-considered discovery stipulations, including protective orders, one must have a fundamental understanding of the nature and goals of forensic software analysis for each of these causes of action. Because the goals differ based on the cause of action asserted, we address each type of cause of action separately.

A. Goals of Forensic Software Analysis

1. Copyright Infringement

In the computer software context, copyright infringement demands the analysis and comparison of two bodies of source code: one from Plaintiff and one from Defendant.[9] The goal of this analysis is to determine whether there are any elements of protectable source code that are identical or substantially similar, a key requirement for any finding of infringement.[10]

---

http://www.cafc.uscourts.gov/images/stories/announcements/Ediscovery_Model_Order.pdf (last visited Dec. 7, 2011).

[9] There may be multiple versions of each party's software products, but nevertheless the forensic analysis process is one of comparison.

[10] Copyright protection affords the copyright owner, *inter alia*, the exclusive right "to reproduce the copyrighted work in copies. . . ." 17 U.S.C. § 106(1). Courts routinely interpret the word "copies" to encompass substantially similar copies in addition to identical copies. *See, e.g.*, Computer Assoc. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992). Additionally, the copyright owner is granted the exclusive right over the preparation of "derivative works." 17 U.S.C. § 106(2). Whether another person's work is a derivative work also can involve the question of substantial similarity. Castle Rock Entm't, Inc. v. Carol Publ'g Group, Inc., 150 F.3d 132 (2d Cir. 1998). The justifications for the protection more expansive than just literal copying range from not allowing the plagiarist to escape liability, to providing robust incentives for creation. *See, e.g.*, Nichols v. Universal Pictures Co., 45 F.2d 119, 121 (2d Cir. 1930) (L. Hand, J.) (explaining "[i]t is of course essential to any protection of literary property . . . that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations."), cert denied 282 U.S. 902

Copyright considers the protectable elements to include not only the literal lines of source code, but also the structure, sequence, and organization of that code.[11] Importantly, copyright law does not protect elements of a computer program that are the product of external constraints.[12] External constraints include such things as the computer hardware, the host operating system under which the program is operating,[13] and the computer-generated source code generated by using such products as Microsoft Visual Studio.[14] This is not an exhaustive list, but the forensic software analyst investigates all instances of identicality or similarity discovered in the comparison of the two bodies of source code, seeking to determine if there is any reason related to unprotected elements that exonerates what might otherwise be evidence of infringement.

The search for identical and substantially similar source code or structure of the program (and the exoneration of constrained source code or structure) requires the use of specialized computer software because the

---

(1931). When dealing with cases involving non-literal infringement, courts employ a test for "substantial similarity." Whether computer software should be treated identically to literary works has generated much scholarly commentary. *See, e.g.*, Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994). Courts have acknowledged the need for protection of computer software beyond the literal lines of code in order to provide appropriate protection for the creators of such works and to prohibit copyists from escaping liability through minor variations. However, because of both the technical nature of computer software and the large quantity of public domain material the code is bound to employ, some courts have insisted upon a showing of "near identity" to find infringement of the non-literal elements of computer software. *See, e.g.*, Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1445-46 (9th Cir. 1994).

[11] This type of "non-literal" protection stems, in part, from computer software's classification as a literary work. Literary works, such as novels, are protected against not only literal copying, but also from paraphrasing and imitation of other expressive elements, such as plot and even specific characters. *See, e.g.*, *Nichols*, 45 F.2d at 121; Metro-Goldwyn-Mayer, Inc. v. Am. Honda Motor Co., 900 F. Supp. 1287 (C.D. Cal. 1995) (noting sufficient authority for the proposition that a plaintiff who holds copyrights in a film series acquires copyright protection as well for the expression of any significant characters portrayed therein). For software, the parallel to the "plot" of a literary work is the structure, sequence, and organization of the computer code. *See* Pamela Samuelson, *The Uneasy Case for Software Copyrights*, 79 GEO. WASH. L. Rev. 1746, 1765-71 (2011). *See also Altai*, 982 F.2d 693; Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222 (3d Cir. 1986).

[12] *Altai*, 982 F.2d 693.

[13] *Id.*

[14] Programs such as Microsoft Visual Studio contain source code templates for much of the routine source code needed for a program to run under Microsoft Windows. Such machine generated source code saves significant amounts of programming time but, as a consequence, produces source code that will be similar and/or identical to all programs created using Microsoft Visual Studio. *See Introducing Visual Studio*, MSDN http://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.80).aspx (last visited July 10, 2012).

programs at issue contain so much source code that it would otherwise defy analysis within the time available.[15]

2. Patent Infringement

In a case involving an allegation of patent infringement, the forensic goal is one of reverse engineering specific parts of the alleged infringer's computer source code to determine whether, when that source code is translated into an executing program, the asserted patent claims are infringed by the program as it executes.[16] This, of course, presumes a detailed understanding of the functionality embraced by the claimed invention and the specific claim limitations, and also requires a probing and thorough analysis of the accused source code.

Forensic software analysis in the context of patent infringement demands the ability to approach a vast (usually jumbled or curiously organized) collection of computer software and to follow the programmatic logic down its various pathways. During this descent, the forensic expert must reverse engineer the source code and create an understanding of what the different elements of the source code will do when they are translated into object code[17] and run on a computer.

Much of the source code that the expert has to analyze will have no relevance to the claims of the patent asserted. Thus, the analysis consists of repeatedly reverse engineering different aspects of the source code, following the programmatic rabbit down the rabbit hole, only to then discard the particular line of enquiry when it becomes clear that the functionality that aspect provides is outside the scope of the asserted claims.

It is quite usual for a significant portion of the actual forensic software analysis to be useless—only hindsight can determine which were those parts of the source code relevant to the asserted claims of the patent, particularly once the court has construed those claims.[18]

---

[15] Modern programs can contain millions of textual lines of source code and it is not humanly possible to analyze (let alone read) that much source code within the time constraints of discovery.

[16] If the case involves defense assertions of patent invalidity or inequitable conduct then the forensic software analysis may devolve into reverse engineering other bodies of source code to determine whether they anticipate the invention or demonstrate, for example the patentee's reduction to practice. For simplicity, this paper only considers infringement.

[17] Object code is the binary zeros and ones that can be loaded into a computer memory and consists of the instructions that the computer executes. Object code controls the apparent behavior of the computer program. Object code is essentially incomprehensible to human beings but can be understood by certain highly trained computer programmers.

[18] Construing the claim language of a patent can be an extremely important part of any patent litigation. Claim construction is a task that the Supreme Court has held is reserved to the judge as a matter of law. Markman v. Westview Instruments, Inc., 517 U.S. 370, 372 (1996) (holding that

3. Trade Secret Misappropriation

While the law of trade secrets is significantly different from that of patents,[19] the forensic software analysis is very similar. The question at hand in a case involving an assertion of trade secret misappropriation is whether the alleged trade secrets are used within the source code. This can only be determined (a) with a clear statement of what exactly *are* the alleged trade secrets at a sufficient level of detail to recognize those secrets if they are used in the source code, and (b) by an iterative reverse engineering of the source code by the forensic software analyst to determine if those alleged trade secrets are, indeed, used. This analysis is similar to that performed in the context of cases involving allegations of patent infringement; the alleged trade secrets correspond to the asserted claims of a patent.

B. Methods of Forensic Software Analysis

Given the vast quantities of computer source code that make up a modern program[20] it is neither cost-effective nor feasible, given typical litigation calendars, for a forensic software analyst to analyze the source code without the assistance of computer software. While average citizens think of computer software as one computer program, in actuality, the type of computer software that is the subject of these complex infringement and misappropriation litigations is far more complex, involving hundreds, if not thousands of separate computer source code files that are interrelated in their programming. The analyst must first assess whether all of the relevant and necessary files of source code, in fact, have been produced in discovery. Once a complete production has been verified the analyst can turn to the task of examining the software for evidence of infringement/ misappropriation, and exoneration. The analyst will need to use a variety of analytic tools, themselves computer software programs, to engage in the necessary review of the source code. Only in rare cases will printed versions of the code aid in the examination of the software.

"construction of a patent, including terms of art within its claim, is exclusively within the province of the court.").

[19] Trade secret law can protect a computer process that would also qualify for patent protection. Trade secret protection is available under most state jurisdictions for information that is "not generally known," has value in its secrecy, and is subject to "efforts that are reasonable under the circumstances" to protect its secrecy. UNIF. TRADE SECRETS ACT § 20:2, 5 CALLMAN ON UNFAIR COMP., TR., & MONO., § 1(4) (amended 1985).

[20] Modern programs typically contain hundreds of thousands of source code files and several million text lines of computer source code for just a single version of a software product.

1. Assessing the Completeness of Source Code Production

Unless the relevant source code is a well-defined fragment of an entire computer program, usually the first stage in any forensic analysis is to determine whether all of the relevant source code has been produced. Given the huge size of modern programs, the most technically viable way of determining completeness is to compile[21] the source code into a working program. This task demands all of the source code and all of the ancillary control files,[22] as well as all of the third party components[23] (be they source or object code) that are necessary for the program to function.

The completeness of the production can only be assessed by building a finished version of the program that can be tested and run. One cannot produce an error-free version of the working version of the program unless all the constituent parts are present. Compiling the program into a working program may take several hours or a few days to complete; it is unlikely that there exists any more cost-effective completeness test. And, without performing this test first, the forensic software analyst cannot know if he or she is working with the full computer software program that is the object of the litigation and thus whether the production of the computer software is complete. The producing party is the best equipped to make this determination before handing over the production; however, without specifying this as a required step for the producing party, it will fall on the shoulders of the receiving party to attempt the process first and so doing will, yet again, dramatically and, it could be argued, unfairly, increase the time and cost burden. The receiving party essentially has to piece together a jigsaw puzzle uninformed by what the final picture must look like.

Producing printouts on paper of the source code is wholly inadequate. Such printouts cannot be verified for completeness because the only effective means of verification is to compile the source code into object code. Such compilation requires the code be in digital textual form. Additionally, printouts of code cannot be searched cost-effectively, and navigation through the source code—jumping from function to function as required by the programming logic—cannot be done at a sensibly fast rate.

---

[21] "Compiling" the source code transforms it into executable object code.

[22] In addition to the actual source code files, there are so-called "makefiles" that are textual recipes that control the process of taking source code and creating the finished executable program, along with the requisite "header" files that contain frequently used definitions that are included by reference when required.

[23] It is quite normal for a computer program to include source code and object code that has been licensed from third parties to provide specific specialized functionality. This is sometimes referred to as "bought in" or "off-the-shelf" code.

2. Forensic Analysis Tools

Once assured of the completeness of the produced files, the forensic analyst will typically turn to the task of examining the produced source code on a computer screen.[24] The analyst will employ specially designed programs[25] that allow high-speed navigation through the tangled logic path of the source code. This logic path can flow into one source code file and then another as each software function call[26] is encountered.

Analytical computer programs are the most cost-effective means for performing the necessary analysis by enabling high-speed navigation within the source code. These tools also facilitate searching vast amounts of source code by looking either for a particular function, data variable, or source code file in the hundreds of thousands of files, and potentially millions of textual lines of source code. Such initial analyses are more often hindered by producing results that contain too much data of the wrong kind. Thus, it is not unusual that the results of the initial analysis using third party software tools will require the development of some *ad hoc* software tools (so-called scripts) to refine the analysis and provide more relevant results. What those scripts might need to do cannot be easily predicted, as the needs are dependent upon the initial results. The analyst writes those scripts or mini-programs on the fly depending on the task to be performed.

a. Tools Specific for Copyright Infringement

For copyright infringement actions, specialized computer software makes it feasible to review and compare millions of lines of source code produced in discovery by both parties and to find those places in the source code where there are several adjacent lines of source code that are either identical or substantially similar.[27] Experience has shown that looking for

---

[24]An analyst is likely to use two large screens so as to be able to compare different aspects of the programs side-by-side. Looking at source code through one keyhole is bad enough. Two screens at least increases the size of the keyhole!

[25] One such program is Understand, which is a very sophisticated tool for ingesting large amounts of source code, analyzing it, creating logic flow paths through that source code, and then allowing the static analysis of that code, navigating through the code as though it were being executed, or searching for relevant parts of the source code. *See* UNDERSTAND: SOURCE CODE AND METRIC ANALYSIS, http://www.scitools.com/ (last visited Oct. 22, 2011).

[26] A "function call" is where one statement in the source code transfers control over to another part of the source code to effect some specific "function" (in the sense of purpose). For example: printf ("Goodbye world") would transfer control to the printf (print formatted) function that transmits the text "Goodbye world" to a display device such as a screen or printer.

[27] *See supra* text accompanying note 10.

just single lines of code that are identical produces an excess of false positives that merely serve to obfuscate the analysis.[28]

Another source of false positives is the hundreds or thousands of lines of code that may have been machine generated or dictated by the use of third party software.[29] These lines of code are either not eligible for copyright protection because they are necessitated by external constraints imposed on the software (such as needing to run under Microsoft Windows or running on an iPad), or because the copyright in those lines of code are owned by the third party, not by the plaintiff asserting infringement.

There are no standard computer programs that perform the filtering of constrained code and juxtaposing instances where several lines of source code are similar or identical. This process demands custom software created by a forensic software analyst. These specialized software tools are created to order and almost always require tailoring to the specific source code produced by the parties in order to minimize false positives and false negatives (those instances where similar code is just dissimilar enough not to be detected).

b. Tools Specific for Patent Infringement

Patent infringement analyses rarely require the comparison of two bodies of source code. Instead the relevant analysis in patent infringement cases requires substantial examination of the source code from the accused infringing device or program. In this analysis the goal is to create a road map of the source code and then locate the relevant areas of functionality in the program. Once identified, these areas of functionality can be studied by the forensic software analyst to determine if they practice an element of the patented invention.[30] Such study can be aided greatly by being able to quickly navigate around the source code following the programmatic logic

---

[28] While one line of code is not protectable under copyright law, exactly how many lines of code must be reproduced to constitute infringement is unclear and will vary with the circumstances. *See* Justin Hughes, *Size Matters (Or Should) in Copyright Law*, 74 FORDHAM L. REV. 575 (2005). The Sixth Circuit has held that a program consisting of eight lines of code is just too short to be copyrightable. Lexmark Int'l, Inc. v. Static Control Components, Inc., 387 F.3d 522, 542-43 (6th Cir. 2004).

[29] For example, Microsoft Visual Studio generates large quantities of "boilerplate" source code that must be filtered from comparison as it will either appear identically in both bodies of source code or will appear substantially similar. Some programs created by Visual Studio may have 80% to 90% of the textual lines of source code be generated by Visual Studio. *See Introducing Visual Studio*, *supra* note 14.

[30] Patent lawyers often refer to the different "elements" of a claim in a patent as "limitations." LYDIA PALLAS LOREN & JOSEPH S. MILLER, INTELLECTUAL PROPERTY: CASES AND MATERIALS 117 (Ver. 2.2, Semaphore Press 2011). In this paper, we use the term elements to help a reader not familiar with patent jargon.

flow from one source code file to another. It is not unusual that several hundred source code files will be implicated in the analysis of a single element of the asserted patent claim, and patent claims typically involve multiple elements.

Fortunately, this type of source code analysis parallels the process of software maintenance—that is, the correction of mistakes in software and/or the addition of new functionality. Therefore the forensic software analyst can use such maintenance programs as *Understand*[31] which ingests large volumes of source code and maps out how the individual source code files and programmatic functions are related, building "family trees" of logic flow as it goes. *Understand* can then be used to navigate rapidly around the source code, effectively ignoring the fact that the source code is distributed across hundreds or thousands of files. This gives the forensic software analyst (or maintenance programmer) a "computer's eye view" of the source code sufficiently rapidly that the forensic software analyst can create a mental model of the structure and purpose of the source code's programmatic logic.

c. Tools Specific for Trade Secret Misappropriation

As alluded to previously, trade secret misappropriation demands a forensic software analysis that is more akin to that performed for patent infringement. An asserted trade secret can be viewed as a claim in a patent and the analysis devolves into answering the question whether the source code performs the functionality that makes up the asserted trade secret. The forensic software analysis is effectively the same as that for patent infringement with the same tools providing extremely important and necessary assistance to the analyst.

III. APPROPRIATE DISCOVERY AND PRODUCTION OF SOURCE CODE FOR FORENSIC ANALYSIS

In federal court litigation,[32] rules 25 through 37 of the Federal Rules of Civil Procedure govern the discovery phase of litigation. The scope of

---

[31] *See supra* note 25 and accompanying text.

[32] While much of the analysis provided in this article is equally applicable to state law litigation involving computer based evidence, our focus in this article is on intellectual property litigation involving either copyrights or patents (or both). Subject matter jurisdiction for those cases rests exclusively with the federal courts. 28 U.S.C. §1338 (2012). Trade secret protection is, for the most part, derived from state law, but often trade secret misappropriation claims are litigated in federal court either because of diversity jurisdiction under 28 U.S.C. §1331, or as a result of being part of a case in which federal copyright or patent infringement is asserted. 28 U.S.C. § 1367 (2012) (defining supplement jurisdiction).

discoverable information is extremely broad. Rule 26 permits discovery of "any nonprivileged matter that is relevant to any party's claim or defense."[33] Additionally, a "court may order discovery of any matter relevant to the subject matter involved in the action" even if that information would not be admissible at trial, so long as it "appears reasonably calculated to lead to the discovery of admissible evidence."[34] This section discusses what is required for complete production of computer source code and recurring problems encountered. This section also discusses appropriate security measures when disclosing source code.

A. The Least Cost Production and Analysis of Source Code, Documentation, and Other Computer-Based Evidence

While the cost increase brought on by an overly protective order is the most significant problem in modern-day litigation involving computer software, ineffective methods of producing software also contribute to the expense of these types of litigation. Thus, before addressing problematic restrictions that routinely appear in overly restrictive protective orders, we first discuss the most cost-effective method of producing source code.

1. Source Code Production

The most cost-effective method of producing the source code for a party is to produce the entire source code tree as it has been maintained under the revision control system. The producing party can verify the completeness of the source code production prior to providing the files to the receiving party by loading the production onto a suitably configured computer system and rebuilding an executable version (or versions) of the program. If a functioning executable version of the program can be created using the files, the source code production is, by definition, complete. If it cannot be done, then the process will reveal the missing components.

It is not unusual for large programs to require several terabytes[35] of disk storage and counsel must anticipate that the task of just making a verbatim copy of the encrypted source code on disk may take several *days*

---

[33] FED. R. CIV. P. 26(b)(1).

[34] *Id.*

[35] A terabyte is 1,000,000,000,000 (a million million) characters of information. Currently 3TB hard disks cost approximately $139 on Amazon. AMAZON, http://www.amazon.com (last visited Jan. 24, 2011).

of continuous computer time. Backing up to media other than disk is not feasible as the storage capacity of other media is insufficient to the task.[36]

2. Other Files Types Required

Several other types of files also must be produced in addition to those files containing the actual source code. Without these files, a functioning executable version of the program cannot be created, and thus the necessary production would be incomplete. Some additional files, such as program documentation, are also necessary to assist in the full analysis of the program.

a. Header Files

Rather than mindlessly restating certain groups of text lines (with the concomitant risk of misstating), it is normal for programmers to insert statements in the source code[37] that include other source code files by reference. These are called header files because such inclusion by reference normally occurs at the start of the source code file in question. Therefore, to make sense of any given source code file, the header files to which it refers must also be produced.

b. Makefiles

The human-readable source code for a program must be translated into object code before that program can control the behavior of a computer. This involves feeding each appropriate source code file into a special computer program called a compiler and then "linking" the output from the compiler together to form the executable object code file. A modern computer program may involve hundreds of thousands of source code files and this compiling and linking process is controlled by a textual file called a "makefile." Makefiles are thus the "recipe" for controlling the process and also serve to explain to a forensic software analyst how the finished program is created.

---

[36] For example, a hard disk may contain 3TB (3,000 gigabytes). A standard DVD can hold 4.7 gigabytes. It would require over 600 DVDs to back up a full 3TB hard disk.

[37] *See* Andrew Johnson-Laird, *Software Reverse Engineering in the Real World*, 19 U. DAYTON L. REV. 843, 856–58 (1994).

c. Revision Control Systems

Where the source code is managed by a revision control program,[38] the order should require the production of the entire source code tree. The entire source code tree should be a required disclosure because (a) this is the least burdensome on the producing party, (b) it contains the complete history of the source code, and (c) it is the electronic form "in which it is ordinarily maintained or in a reasonably useable form or forms."[39]

d. Required Documentation

The production almost always also includes supporting design documentation, user manuals, software development documents (such as project planning, budgeting, testing, bug reporting, etc.). These documents act as a "road map" without which the vast numbers of source code files are even harder to understand.

Usually all but the largest software developers do not bother to prepare any "road map" documentation offering insights as to how the software works and how the software's functionality is packaged within the source code. Thus, finding a particular piece of functionality is akin to finding one's way in a city where there are no street name signs, using a map drawn on a napkin by someone you met in a bar. It is not easy.

The original source code authors do not remember accurately how all of the source code they wrote works. This appears to be because software is now sufficiently complex that it cannot be retained in the minds of the original programmers, especially when months or years have elapsed since it was written. Thus 30(b)(6) witnesses[40] may not be relied upon to explain where certain functionality can be found in the source code and the forensic software analyst must construct the missing road map to navigate around the source code.

Additional "road map" documentation rarely manifests after the initial delivery. If it does, it is usually in response to a request from the

---

[38] A revision control program is akin to the document management system one finds in a modern law office. It tracks all documents and keeps a revision history showing all changes made to a given document.

[39] FED. R. CIV. P. 34(b)(2)(E)(ii).

[40] Federal Rule of Civil Procedure 30(b)(6) permits a party to depose an entity, such as a partnership or corporation so long as the party "describe[s] with reasonable particularity the matters for examination." FED. R. CIV. P. 30(b)(6). In turn, "[t]he named organization must then designate one or more officers, directors, or managing agents, or designate other persons who consent to testify on its behalf; and it may set out the matters on which each person designated will testify." *Id.* When such a designated person is deposed, they "must testify about information known or reasonably available to the organization." *Id.*

forensic software analyst and has been prepared specifically for the litigation; it may therefore have been prepared in haste and of dubious completeness and accuracy.

B. Fundamental Problems in the Production of Source Code During Discovery

It is easy to fall into the trap of believing that the source code that is produced in litigation represents an ordered delivery of all relevant versions of the relevant computer source code and documentation and, thus, it will be a relatively easy task to analyze source code appropriately. In fact, recurring problems fall into three categories, each discussed in turn below.

1. Obtaining All Relevant Source Code Files in Appropriate Digital Format

As discussed above, ensuring complete production of the program at issue must be a top priority. Often, significant amounts of the forensically relevant source code will not be produced in the initial production, if at all. Computer programs are so complex that often companies cannot keep track of their source code. At the same time, significant amounts of forensically *irrelevant* source code will be produced, but often in the wrong form.

The most usable, most cost-effective form of production is to produce the entire source code "tree" as created and maintained under the aegis of a revision control system.[41] Nevertheless, source code production often appears as printed versions of source code, or the even more bizarre form of Adobe Acrobat Portable Document Format (PDF) files created by scanning in printed versions of source code—which are thus graphic images and not searchable! All formats other than the original source code tree in digital format are maximally inconvenient and significantly increase the cost of the analysis. Furthermore, there seems to be no legal or technical basis for not producing the source code in the form in which it is created and maintained.[42] Computer programmers rarely print out source code, as it is too hard to navigate and slow navigation obscures understanding. Instead they will use the source code tree in digital form and use specialized

---

[41] *See supra* Part II.A.1.

[42] Indeed, the Federal Rules of Civil Procedure require parties to "produce documents as they are kept in the usual course of business or must organize and label them to correspond to the categories in the request." FED. R. CIV. P. 34(b)(2)(E)(i). The rule also requires that if "a request does not specify a form for producing electronically stored information, a party must produce it in a form or forms in which it is ordinarily maintained or in a reasonably usable form or forms." *Id.* 34(b)(2)(E)(ii).

computer programs ("development environments") to navigate quickly around the source code tree.

Not all relevant versions of the forensically significant source code will be produced. It is not unusual for smaller companies not to use revision control systems at all, or to change the revision control system between versions as a product evolves. Additional deliveries of missing source code may take place over weeks and/or months either as it is "discovered" by the producing party or, more likely, when the forensic software analyst demonstrates to the parties and/or the court that the production is incomplete. Such staggered productions will, more likely than not, require the computer-aided forensic analysis to be redone to create a complete picture of the entire source code production to date. Each such late partial delivery can increase the time and cost of performing the forensic software analysis by an order of magnitude.

While purposefully engaging in conduct merely to increase the cost and inconvenience of discovery is expressly prohibited by the Federal Rules of Civil Procedure,[43] computer source code productions in any form other than the original source code tree does just that. Similarly, partial productions and incremental productions, both of which require re-analysis, also serve to increase the cost and inconvenience of the receiving party. The discovery order agreed to by the parties should not tolerate inconvenient formats for production and should be designed to ensure complete production from the beginning.

2. Obtaining Necessary and Relevant Information Beyond the Source Code Files

As described above, in addition to the actual source code of the program, several other types of files also must be produced in order to obtain a functioning executable version of the program.[44] The documentation contained within the source code, by and large, will be acceptable, but much of what should have been written will not have been, or having been written will not have been updated to correspond to the current version of the source code, thus rendering it obsolete and misleading in places. Programmers in general appear to hate preparing documentation—perhaps because good documentation is really hard work to prepare—so much so that technical writers are employed to write the documentation and often they are not programmers, nor do they have access to the most current version of the software.

---

[43] *See supra* note 5.

[44] *See supra* Part III.A.2.

Reverse engineering source code is much harder than writing it in the first place. The original programmer has the benefit of understanding his or her intent. The reverse engineer must glean that intent by constructing mental models of the source code "as is," without the benefit of the original intent. Unless suitable accurate comments are interwoven in the source code, the source code will reveal what is being done, but not why. The understanding of why has to be re-synthesized by the forensic software analyst as they create a mental model of what the source code would do when compiled and executed on a computer. To the extent that additional documentation exists, it should be promptly produced as part of the discovery process.

All third party source code products used to augment the source code should also be part of the initial production. A party may assert that such production would be a breach of the license agreement for that software or that such production would constitute copyright infringement. To the authors' knowledge there has never been a breach of license or copyright action filed when third party source code and/or software has been produced in the context of litigation and subject to a protective order. However, the issue of production of relevant third party software that is a component of the program at issue must be addressed in the discovery order agreed to by the parties.

3. Shifting the Cost of Discovery for Incomplete Production

One possibility for encouraging full and complete production initially is for the shifting of costs associated with incomplete production.[45] Rule 37 permits a court to award "reasonable expenses, including attorney's fees, caused by" a party's failure to comply with a discovery order, "unless the failure was substantially justified or other circumstances make an award of expenses unjust."[46] That same rule provides that "an evasive or incomplete disclosure, answer, or response must be treated as a failure to disclose, answer, or respond."[47]

The stipulated order may contain a provision concerning the payment of expenses associated with disclosure. While the default assumption is that each party will pay its own expenses, any clause addressing the payment of expenses should address whether a court could

---

[45] *See* Martin H. Redish, *Electronic Discovery and the Litigation Matrix*, 51 DUKE L.J. 561, 608 (2001) (advocating cost-shifting for requests involving electronic information that is stored in a format not reasonably accessible by the producing party).

46 FED. R. CIV. P. 37(d)(3).

[47] *Id.* 37(a)(4).

order the payment of expenses despite the agreement. Without addressing such a scenario, the silence of the agreement may be difficult for the court to interpret.[48]

C. Important and Appropriate Security measures

Parties disclosing their software are understandably concerned about the risk of disclosure beyond the expert(s) hired by the opposing party. Sometimes they fear disclosure to the opposing party as well as disclosure to independent third parties. The intellectual property assets contained in the source code may have taken literally decades of person-hours to create and can be of significant value. Further, in this age of rapid and global dissemination, parties fear that if proprietary source code being disclosed were to be leaked to the outside world the value could be totally destroyed in a matter of hours, if not minutes, and could never be recaptured.[49] Because of this, there are several important security protocols that should be followed concerning the discovery and handling of that source code.

1. Security in Transit

The source code produced should be produced on an encrypted *disk*, rather than using file-by-file encryption which is overly burdensome on both the producing party and the requesting party as each file must be decrypted/encrypted individually. Such file-by-file processing can lengthen a process that would normally take hours into days.

Products such as TrueCrypt[50] or Pretty Good Privacy (PGP)[51] both provide military-grade encryption for entire hard disks. Such encryption is

---

[48] *See, e.g.*, Thabault v. Chait, No. 85-2441, 2009 U.S. Dist. LEXIS 576 (D.N.J. Jan. 5, 2009). In *Thabault*, the parties agreed to share the cost of daily transcripts during trial. The court held that such an agreement did not preclude an award of costs to the prevailing party. The agreement to share costs of transcripts was "a far cry from agreeing . . . about what costs the prevailing party could recover." *Id.* at *16.

[49] Many high profile leaks have occurred, although none were the result of disclosures related to litigation. *See, e.g.*, United States v. Genovese, 409 F. Supp. 2d 253, 257–58 (S.D.N.Y. 2005) (prosecuting defendant for his involvement in a significant leak of Microsoft Corporation's source code for its computer operating systems, Windows NT 4.0 and Windows 2000). *But see* Robert Lemos, *Cisco Investigates Source Code Leak*, CNET NEWS BLOG (May 17, 2004), http://news.cnet.com/Cisco-investigates-source-code-leak/2100-7349_3-5213724.html?tag=contentMain;contentBody;1n (indicating that the damages might not be significant of leaks of computer source code). *See also* Victoria A. Cundiff, *Reasonable Measures to Protect Trade Secrets In a Digital Environment*, 49 IDEA 359, 395–408 (2009).

[50] TRUECRYPT, http://www.truecrypt.org, (last visited Mar. 11, 2011). Truecrypt is well trusted in the computer industry and is available at no cost. Although counterintuitive to some, the best encryption software is that which is subjected to public scrutiny, thus benefiting from the combined wisdom of the crowd and avoiding any hidden "back doors."

sufficiently strong that in the worst-case scenario that the disk falls into a malfeasant's hands, that malfeasant will not be able to decrypt the source code and associated files. Furthermore, this encryption is "transparent" in that once the disk has been mounted on the computer and the decryption password entered, the disk's contents appear to the computer as unencrypted data even though, on the disk, the data remains encrypted.[52]

To achieve even stronger levels of encryption for individual files, an encrypted file can be doubly or triply encrypted using different passwords—somewhat like stacking Russian Dolls, with one encrypted doll inside another.

Encryption by itself does not prevent unauthorized use of the encrypted data; therefore the encryption/decryption keys have to be managed very carefully.[53] If a key falls into the wrong hands the value of the encryption is vitiated. At least two people at the producing party and two people at the receiving party should be designated key managers, and they should store the encryption keys in an encrypted file using a different encryption key that only the key managers know.

2. Packaging for Shipment

Hard disks must be treated with care like Fabergé eggs. On more than one occasion we have received hard disks that have been shipped with woefully inadequate packaging, arriving at their destination as no more than effective paperweights. Each hard disk must have at least one inch or more of shock-absorbing packaging. Computer system enclosures and RAID[54] enclosures are very poor shipping containers as they are rigid and transmit external shocks to the hard disks within. We have been the recipients of computer systems where the hard disks have come dislodged in transit, destroying themselves and the computer system like a loose cannon on the deck of a tall ship.

---

[51] *Pretty Good Privacy*, WIKIPEDIA, http://en.wikipedia.org/wiki/Pretty_Good_Privacy (last visited July 20, 2012).

[52] If a malfeasant were to burst into the room and disconnect the disk, its contents are still fully encrypted and inaccessible.

[53] NEILS FERGUSON ET AL., CRYPTOGRAPHY ENGINEERING: DESIGN PRINCIPLES AND PRACTICAL APPLICATIONS 257–313 (2010).

[54] Redundant Array of Inexpensive Disks—a group of several hard disks that are combined by hardware or software to present to the computer system as a single larger hard disk.

3. Security for Printed Source Code in Transit

While we strongly urge against production of source code through printed copies,[55] if the parties agree to disclosure by printed copies, that printed source code in transit is vulnerable to being misplaced or falling into unauthorized hands. If a Federal Express envelope is intercepted, then the source code is in full view with nothing to prevent a malfeasant from making use of it. It is far more secure to take the source code, already formatted for printing, and create Adobe PDF files. These files can then be encrypted and burned onto a DVD that is then shipped by Federal Express. The recipient can then decrypt the PDF file to either view it or print it as required. If such a DVD falls into the wrong hands, the encrypted source code cannot be viewed or used.

4. Forensic Analysis: Stand-Alone Computer Isolated from the Internet

The analyst(s) performing the appropriate forensic analysis on the source code should work on a computer that is physically isolated from the Internet to avoid possible unauthorized access to the source code. While the source code will be encrypted on disk, there will be periods of time during the analysis where the source code will, of necessity, not be encrypted inside the computer's memory. This is when the source code will be at its most vulnerable.

Using a stand-alone computer does increase the cost of the analysis, but usually only by the cost of the computer and the costs of additional software licenses for the analytical software tools. This increased cost is appropriate for the level of security it provides.

D. Model Clauses for Ensuring Appropriate and Complete Production

Often the discovery order agreed to by the parties, while titled "protective order," will specify what is to be disclosed and its format, as well as contain clauses that would normally be recognized as being in a protective order. The following recommended language accomplishes the production of computer source code in the most cost-effective and secure manner, permitting forensic software analysis of source code with the minimum of wasted time, effort, and burden (and thus cost) to either of the parties. It is presumed that preceding sections of the protective order appropriately define key terms (such as the Source Code).

---

[55] *See supra* Part III.B.1 and *infra* notes 82–83.

Source Code will be made available for inspection and analysis as follows:

a) The producing party will make available all relevant versions of the Source Code by placing the entire Source Code tree(s) for all versions, as maintained by the revision control system in the format in which it was created and maintained by the producing party, on a hard disk encrypted with TrueCrypt (or equivalent disk-level encryption) (the "encrypted disk"). A secure password consisting of two strings of alphabetic characters, each eight characters or more, separated by one or more digits, shall be used. The strings of characters shall not form words found in the Merriam-Webster dictionary.[56]

b) The Source Code production shall include all relevant Source Code files including header files, makefiles, and third party source code and object code files used by the executable version(s) of the program in format in which it was created and maintained.

c) All produced materials that are in color in their original form shall be produced in color.

d) All produced materials that exist in textual form shall be produced in the original form in which they were created and maintained by the producing party.

e) Prior to producing the entire Source Code tree(s) to the receiving party, the producing party will verify the completeness of the Source Code production by recreating an executable version of each version of the computer program for which the Source Code is being produced using only those files on the encrypted disk. Evidence of such recreation will be provided to the receiving party on the encrypted

[56] This eliminates the use of a brute-force "dictionary attack" on the password. The easiest way to accomplish constructing such a password is to create an acronym from a phrase. For example: mhalltdws—Mary had a little lamb, the doctor was surprised.

disk along with a description of the process used sufficient to permit the receiving party to replicate the recreation.

f) The producing party will provide the receiving party a complete list of the software tools used to create and maintain the Source Code tree and to re-create the executable version(s) of the computer program. If such tools are no longer available for license by the receiving party, the producing party shall provide copies of the tools on the encrypted hard disk, along with documentation on how to use the tools, and with appropriate license keys as required to use the tools.

g) The producing party shall provide a "road map" document that describes the organization of the Source Code on the encrypted disk, explaining which versions of Source Code are contained within the tree and the overall structure of the tree such that the receiving party can understand the various components of the Source Code tree. This road map shall also describe the locations and function all of the non-Source Code elements on the encrypted disk including, but not limited to, the third party components and the software used by the producing party that are on the encrypted hard disk.

h) The producing party shall provide all supporting documents relevant to the Source Code such as design specifications, diagrams, project management and planning documents, budgeting documents, testing scripts and data, and bug reports, in the form in which they were originally created and maintained. Where such supporting documents were created using software that is not available on the open market, then the producing party will provide all necessary software to permit the forensic software analyst to examine and, if necessary, print all supporting documents.

IV. APPROPRIATELY PROTECTIVE AND OVERLY-PROTECTIVE ORDERS

Providing for the appropriate scope of discovery in cases involving allegations of intellectual property infringement is only half the battle. Including clauses that will ensure protection against disclosure beyond the

litigation remains extremely important. After providing an overview of protective orders in federal litigation, we propose a set of provisions that would guard against disclosure while permitting analysis to proceed.

This section concludes with an examination of restrictions that we have seen in recent overly protective orders. For each such restriction we will state hypothetical language that paraphrases actual clauses of extant protective orders. The primary concern with such overly protective order clauses is that they are insidious—they are often little more than "security theater,"[57] offering an illusory benefit while significantly discommoding the receiving party by increasing the burden and cost of performing the source code analysis.

A. Overview of Protective Orders in Federal Court Litigation

In the federal courts, Federal Rule of Civil Procedure 26 governs protective orders.[58] Adopted to safeguard parties and witnesses in response

---

[57] For an overview of the concept of "security theater," see *Security Theater*, WIKIPEDIA, http://en.wikipedia.org/wiki/Security_theater (last visited July 19, 2011).

[58] Fed. R. Civ. P. 26(c) specifically provides:

> A party or any person from whom discovery is sought may move for a protective order in the court where the action is pending—or as an alternative on matters relating to a deposition, in the court for the district where the deposition will be taken. The motion must include a certification that the movant has in good faith conferred or attempted to confer with other affected parties in an effort to resolve the dispute without court action. The court may, for good cause, issue an order to protect a party or person from annoyance, embarrassment, oppression, or undue burden or expense, including one or more of the following:
>
> > (A) forbidding the disclosure or discovery;
> >
> > (B) specifying terms, including time and place, for the disclosure or discovery;
> >
> > (C) prescribing a discovery method other than the one selected by the party seeking discovery;
> >
> > (D) forbidding inquiry into certain matters, or limiting the scope of disclosure or discovery to certain matters;
> >
> > (E) designating the persons who may be present while the discovery is conducted;
> >
> > (F) requiring that a deposition be sealed and opened only on court order;
> >
> > (G) requiring that a trade secret or other confidential research, development, or commercial information not be revealed or be revealed only in a specified way; and
> >
> > (H) requiring that the parties simultaneously file specified documents or information in sealed envelopes, to be opened as the court directs.

to the extremely broad right of discovery,[59] protective orders are an important tool allowing parties the chance to restrict the range of discovery in situations that might cause injury and also to restrict subsequent disclosure of information produced in discovery. When a party violates a protective order, a district court may impose appropriate sanctions to remedy the violation.[60]

In intellectual property litigation most protective orders are negotiated between the parties and entered by the court as stipulated orders.[61] Ideally, the attorneys will engage their experts prior to agreeing to a protective order with opposing counsel. This will permit consultation with the expert concerning the appropriate and inappropriate restrictions in any given case.

In the following sections we propose example protective order language that both meets the technical requirements at hand and also minimizes the time required and the costs to the litigants.

B. Model Clauses of an Appropriately Protective Order[62]

It is appropriate to include provisions in the "protective order" designed to guard against disclosure of the source code. The following recommended language provides sufficiently robust protection yet will permit the forensic software analysis of source code to proceed with the minimum of wasted time, effort, and burden (and thus cost) to both of the parties. As with the proposed clauses above, it is presumed that preceding sections of the protective order appropriately define key terms, such as the Source Code.

---

[59] 8a CHARLES ALAN WRIGHT, ARTHUR R. MILLER & RICHARD C. MARCUS, FEDERAL PRACTICE AND PROCEDURE: CIVIL § 2036 (3d ed. 2010).

[60] 6 JAMES WM. MOORE ET AL., MOORE'S FEDERAL PRACTICE § 26.108[2] (3d ed. 2010). *See* Poliquin v. Garden Way, Inc., 154 F.R.D. 29, 31 (D. Me. 1994) (attorney sanctioned for violation of protective order that discovery materials not be disclosed to anyone other than counsel for parties or witnesses). There is a limit to what sanctions are appropriate. MOORE ET AL. § 26.108[2]. *See* Coleman v. Am. Red Cross, 979 F.2d 1135, 1141 (6th Cir. 1992) (court abused discretion in enjoining plaintiff from suing blood donor whose name was obtained in violation of protective order).

[61] The rule governing protective orders encourages such agreement by requiring any party that seeks a protective order from the court to certify that they have "in good faith conferred or attempted to confer with other affected parties in an effort to resolve the dispute without court action." FED. R. CIV. P. 26(c). Additionally, because often these cases require both parties to disclose source code, it is in both parties' interests to stipulate to terms for disclosure that are beneficial.

[62] We begin the lettering here with "i" because the appropriate scope provisions were identified above as a–h. *See supra* Part III.C.

Specifically omitted from the following sections are the sections of a protective order that deal with the designation of confidential materials, the marking of such materials, and restrictions that typically might limit to whom confidential materials may be disclosed. Typically, these sections do not bear on increased or decreased costs of discovery. If they do, the egregious nature of the restrictions is self-evident, as they do not demand knowledge of computer science and forensic software analysis.

i) The receiving party shall disclose to the producing party the identity of any expert who will be given access to the materials disclosed pursuant to this order. Any objection to any individual expert must be made within five days of the identification of the individual. After five days, with no objection from the producing party, the identified expert shall be given access to the materials disclosed, along with a copy of this protective order. If producing party objects to an identified expert, the receiving party shall either select a new expert or shall petition the judge for approval of the expert already selected.

j) The receiving party (counsel and experts) on receipt of the encrypted hard disk containing the production shall immediately create a backup copy of the encrypted disk. All subsequent analysis will be performed on the backup copy of the encrypted disk (the working copy). The original copy of the encrypted disk shall be placed in a bank safety deposit box under the custody and control of the receiving party. It may be removed from the safety deposit box if and when it becomes necessary for the receiving party to make another working copy of the contents of the encrypted hard disk.

k) When not in use, the working copy of the hard disk shall be disconnected from the computer system and placed in a locked container (e.g., a safe or filing cabinet) at the premises of the receiving party.

l) The Source Code analysis shall be performed on a computer in a secure room at the receiving party's premises. Access to this room will require a

physical combination door lock[63] to be unlocked. The combination will only be disclosed to those individuals needing to enter the secure room for the purposes of performing the analysis. Janitorial and maintenance staff will be escorted by a forensic software analyst into and out of the secure room and supervised by that analyst while in the secure room.

m) The computer in the secure room shall not be connected to the Internet, but may be connected to external peripheral devices via a local network provided (a) that network does not extend outside the secure room and (b) such devices are necessary to the performance of the analysis (e.g., external disks, printers).

n) The automated logging capabilities of the operating system on the stand-alone computer(s) shall be enabled to create a date/time stamped log of which users log on to the computer and, if appropriate, which files on the computer system are accessed. Each forensic software analyst shall use a unique user account assigned to them when performing the analysis. The operating system log files shall be transmitted to the producing party upon request by the producing party.

o) A second encrypted hard disk shall be used to store the interim results of the analysis. A backup copy of this encrypted hard disk may also be made to a third encrypted hard disk. This interim results disk and backup interim results disk shall also be removed from the secure computer when not in use and shall be stored in the same locked container as the working copy of the encrypted Source Code disk.

p) TrueCrypt (or equivalent) shall also be used to encrypt the interim and backup interim disks and

---

[63] Physical keys can be misplaced. Combination locks provide better security because (a) the combination can easily be changed, and (b) the combination can be communicated by voice to authorized persons.

the same encryption/decryption passphrases shall be used as for the encrypted Source Code disk.

q) At the discretion of the forensic software analysts performing the analysis, more than one computer may be used for the purposes of analysis if this is deemed more cost-effective or reduces the time for the analysis. Additional computers shall be subjected to the same conditions as above. Each shall be permitted its own working copy of the Source Code disk, interim results disk, and backup interim results disk. Each shall be permitted to be on the same network as other computers subject to the previously stated conditions. Such networks will be through physical connections only and shall not be established or maintained through wireless networking capabilities.

r) Copies of the Source Code and other materials produced, regardless of whether they are electronic or paper, shall only be made for the purposes of litigation. All copies shall be securely destroyed upon completion of the litigation. Optical media copies shall be physically shredded. Electronic copies on hard disks or thumb-drives shall be overwritten using a commercially available program designed for secure erasure in conformance with the National Institute of Standards and Technology SP-800-88.[64]

s) To produce paper printouts of Source Code and/or interim analysis results, the printouts shall first be written to the encrypted interim hard disk (and the backup interim hard disk) to create "printout files" in the form of Adobe Acrobat Portable Document Format (PDF) files. [Optional restriction: No paper printouts of Source Code shall be used for correspondence between the parties, expert reports or deposition exhibits. Instead, reference shall be made to the Source Code by file directory path, file

---

[64] *See* Richard Kissel et. al., *Guidelines for Media Sanitation*, NAT'L. INST. OF STANDARDS & TECH. Spec. Publ. 800-88 (2006), http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf.

name, and line numbers within the Source Code file.]

1) All such printout files shall bear a heading line that shows, at a minimum, the full "file path" and file name of the file from which the Source Code is being printed. Marginal numbers starting at 1 and increasing by 1 for the entire Source Code file will identify the Source Code line numbers. Any redactions or resequencing of Source Code lines shall preserve the original Source Code file line numbers and will bear an interlineated legend identifying that a redaction or resequencing has occurred.
2) The heading line shall also bear, at a minimum, a page number [and, optionally, the date and time when the printout was produced] designation for ease of reference.
3) The footing line shall bear, at a minimum, the appropriate confidentiality designation.
4) All such printout files shall be stored in a specific subdirectory on the interim and backup interim hard disks to facilitate review by the both parties.
5) All such printout files shall have meaningful names that describe the contents of the files, the date and time when they were created, a version number, and the initials of the person creating the file.
6) All such printout files shall be preserved and will be produced to the producing party on request.
7) Printout files may only be shipped or transported in encrypted files on computer media or on encrypted hard disks. The same encryption software used for the Source Code disk shall be used and the same encryption/decryption passphrase shall be used.
8) Upon termination of the litigation all such printout files shall be destroyed as specified in paragraph p, above.

t) Physical paper printouts of printout files shall only be created where it is necessary for the purposes of the analysis to have them on paper or for exhibits at deposition or trial.
    1) When not actually being used these physical paper printouts shall be stored in a locked container along with the interim and backup interim hard disks.
    2) Physical paper printouts shall be destroyed using a chipping shredder once they have been superseded or are no longer relevant.
    3) Except as required for deposition exhibits and demonstrative exhibits at trial, no physical paper printouts shall be shipped or transported by experts or counsel.

C. Overly Protective Orders

Our actual measured experience is that overly protective orders can increase the cost of forensic software analysis by somewhere between six and ten times the cost of the forensic software analysis conducted using the suggested protective order language above. In this section, we summarize clauses that might typically occur in an overly protective order pertaining to source code analysis and discuss the presumed intent of that language as well as the consequences. Where possible, we then explain how a specific model clause or clauses appropriately addresses the producing party's legitimate concerns. In some cases we provide an alternative clause that should achieve the original intent of the overly protective clause. We also identify potential objections that a producing party's counsel may raise concerning why the model or suggested alternative is insufficient. In this discussion it will be clear that some of the clauses often appearing in protective orders seem to have no legitimate basis, but rather are designed to make analysis maximally inconvenient and expensive. Inappropriately driving up the cost of discovery is, as discussed above,[65] prohibited under the Federal Rules of Civil Procedure.

[65] *See supra* note 5.

1. Stand-Alone Computer Not At Forensic Software Analyst's Office: Overly Protective Clauses

a. Stand-Alone Computers: Location

It is not unusual, for security reasons, for the parties to agree that the confidential source code be examined on a computer at a mutually agreed upon location. Typical language is:

> Computer source code will be made available on a stand-alone computer at a mutually agreed-upon location.

The presumed intent is that a major risk for proprietary source code is that the computer on which it is stored will be accessed ("hacked") and the source code copied from the computer for nefarious use. The presumption is that this risk is eliminated by using a stand-alone computer that has no connections whatsoever to the Internet.

The innocuous phrase "a mutually agreed-upon location" is a major cause for concern if that location is anywhere other than the offices of the forensic software analyst. If the location requires travel, or access to the stand-alone computer is regulated in any way, this will increase the forensic software analysis time and cost dramatically. The two major cost drivers are (1) potential travel to and from the remote site, and (2) the implicit fact that access to the stand-alone computer will be subject to date- and time-based restrictions.[66]

Forensic software analysis usually requires many hours in front of the stand-alone computer, along with many hours during which the computer is running software analyses unattended. In the early stages of analysis it is not unusual that the stand-alone computer will be left powered on and processing the source code for several days at a time. [67]

Thus, access to the stand-alone computer anywhere other than the forensic software analysts' own office will likely necessitate significant travel and accommodation costs, as well as considerable "dead time" while the analyst waits for the computer to grind through the multi-day analysis process. During such processes, there is often little useful work that can be done, and, of course, given Murphy's Law and Johnson-Laird's Sixth Law

---

[66] The problem of limited hours of access is addressed in the next subsection.

[67] If, as unfortunately is usually the case, the source code production is incomplete and new "code drops" are received once the forensic software analysis has started, then these multi-day processing runs will need to be repeated, requiring multiple journeys to the stand-alone computer's location. *See supra* Part III.B.1. Utilizing the discovery plan provided in the sample clauses in this article should reduce the risk of this situation occurring.

of Forensics,[68] should the analyst leave the stand-alone computer's location, there is an increased probability that something will go wrong with the processing run.

The most significant way to reduce the cost of forensic software analysis is to permit the stand-alone computer to be located at the forensic software analyst's own office. This simple change slashes the time and cost for the analysis when compared to having the stand-alone computer hundreds of miles away, or even just a few miles away. Clause l, above, achieves this solution.[69]

It is not unusual for the producing party to object to having the computer at the forensic software analyst's office on the basis that the source code will be less secure. However, what keeps the source code secure is encryption[70] not physical access. Utilizing the model clauses provided above, the stand-alone computer will be devoid of the source code as the source code is only ever stored on external disks that are placed in a locked safe when not in use.[71]

Even if a malfeasant were to steal the stand-alone computer and the encrypted disks, it would require decryption of the source code. Given that TrueCrypt or PGPDisk is used and that the decryption keys are kept secure, such unauthorized decryption will require a huge amount of computer power and is, for all practical purposes, infeasible. In 2008, the FBI tried for five months to break TrueCrypt encryption and failed.[72] That said, no digital encryption scheme should be viewed as completely impregnable, but the issue here is that the source code should be protected to the same level or better than when it is on the servers or computers of the producing party.[73]

---

[68] Which states that a computer is more likely to crash when one is not watching it than when one stares at it.

[69] *See supra* Part IV.B, cl. l.

[70] *See* TRUECRYPT, *supra* note 50 and accompanying text. Sample clauses a and p above require the use of sufficiently robust encryption to protect the files.

[71] Of course, the stand-alone computer is required to not be connected to the Internet. *See supra* Part IV.B, cl. m and o; *see also supra* note 70 and *infra* notes 72–73 and accompanying text.

[72] *See Not even FBI was able to decrypt files of Daniel Dantas*, G1: O PORTAL DE NOTICAS DA GLOBO (Jun. 25, 2010, 10:35 AM) http://www.webcitation.org/query?url=g1.globo.com/English/ noticia/2010/06/not-even-fbi-can-de-crypt-files-daniel-dantas.html. This is not to say that TrueCrypt encryption can *never* be broken, but it does indicate that the encryption is very, very hard to crack.

[73] Indeed, many of the provisions in the model clauses are designed to ensure the physical security of the computer on which the analysis is performed and the security of all storage media on which the code or portions of the code are reproduced.

There is also a sub-text at issue in the overly protective clause and objections to not using such a clause. At its base, the producing party may be concerned about the trustworthiness of the opposing party's expert. Can the forensic software analyst employed by the opposing party be trusted sufficiently to be given access to the source code, or will the expert indulge in wholesale copying of the source code for nefarious reasons? The discovery and protective order agreed to by the parties requires all persons that will be given access to the disclosed files to be identified.[74] The order further specifies that the disclosing party have a set period (e.g., five days) in which to object to the expert before that expert may be given access to the disclosed materials. The disclosing party may object to the selected expert on the grounds of trustworthiness or expertise.[75] If the forensic software analyst cannot be trusted, then counsel for the receiving party should engage different forensic software analysts rather than permit their client to suffer the massive increase in costs imposed by the producing party's demand that the stand-alone computer not be located in the forensic software analyst's offices.[76]

b. Stand-Alone Computer(s): Hours of Access

Given that counsel have the mind-set that it will be cost-effective for the stand-alone computer to be located at either law offices or a software-escrow firm's offices in spite of the increased time and cost of so doing, it is not unusual for the protective order to contain a clause such as:

> The producing party shall make reasonable efforts to provide access to the stand-alone computer(s) during from 8:00 a.m. through 6:00 p.m. on normal working days.

---

[74] *See supra* Part III.D, cl. i. The Federal Rules of Civil Procedure provide for the disclosure of testifying experts. FED. R. CIV. P. 26(a)(2). The model order requires this disclosure occur earlier than the rules require, and also requires disclosure of all experts, regardless of whether they will testify at trial.

[75] "There are only a few ways to completely ruin your reputation as a forensic software analyst and breaching a protective order is the best one." The late Stephen J. Davidson, Esq.

[76] While the selection of a forensic software analyst is outside the scope of this paper, some fundamental questions may help to weed out the inappropriate analyst:

> a) Is the office where the stand-alone computer going to be located protected by a professionally installed and monitored building alarm system?
>
> b) Is this office protected by doors with deadbolts?
>
> c) Has the forensic software analyst ever had prior issues regarding maintaining source code securely?
>
> d) Does the forensic software analyst understand how to use and maintain encrypted source code?

The presumed intent of this clause is that the forensic software analysis should be confined to normal business hours. This is further presumed to avoid issues of providing access to facilities and/or logistical issues for supervision of the analyst.

The consequence of this restriction can be disastrous in terms of the burden—and the resultant cost increase—placed on the receiving party. The software tools used to perform the forensic software analysis on the stand-alone computer will usually need to run for more than the normal working day and it is rare that the analyst will be able to leave the stand-alone computer running overnight. Thus, there is a Catch-22: the computer needs to be left running an extended source code analysis program for more than eight hours, but the computer cannot be left powered on for more than eight hours. Therefore, the analysis run can never be run. Of course, even if the particular processing run is likely to take, say, four hours, then it can be started no later than 2 p.m.

Unfortunately, it is hard to predict how long such an analysis run will take—the duration depends in large part on the data being analyzed. We have seen situations where many hours have been wasted because the analysis software processing has had to be aborted at 6:00 p.m., only to be restarted from the beginning the next day in the hope that it will finish in time.

A further presumed unintended consequence is that the forensic software analyst must suspend all analytical work during weekends and holidays. Given that the stand-alone computer's location requires travel to a distant city, this is maximally inconvenient to the analyst and serves to increase the time and cost that the analysis takes.

The simple solution is to permit the stand-alone computer to be located at the forensic software analyst's office where access is not fettered by the day and time. Placing the source code on a disk-level encrypted external hard disk and permitting the stand-alone computer and encrypted disk to be at the forensic software analyst's own offices where it will be placed in a locked room will allow long duration analysis processing to continue uninterrupted seven days a week, should that be necessary (and it often is).

As before, the producing party typically objects based on the misguided assumption that the security of the source code is based on physical access to the stand-alone computer and the hard disk that contains it. As explained above, it is the encryption of the source code that provides

the security, not controlled access to the encrypted hard disk or the stand-alone computer.[77]

c. Stand-Alone Computer(s): Proscribed Items in Room Containing Stand-Alone Computer

When appropriate concern to prevent inappropriate copying turns into paranoia, a clause like the following appears:

> No recordable media or recordable devices, such as sound recorders, computers, cellular telephones, peripheral equipment, cameras, CDs, DVDs, or drives of any kind, shall be permitted into the room with the stand-alone computer.

The presumed intent is to prevent copying of the source code from the stand-alone computer by prohibiting anything apparently capable of copying the source code near the stand-alone computer.

The primary unintended consequence becomes clear when there is a hardware or software problem with the stand-alone computer. The forensic software analyst can now only make notes about the failure, then leave the room to make a cell phone call—only to have to dash back into the room and try something—and then leave the room to get back to the phone.

We have seen situations where it took five calendar weeks to get the stand-alone computers working reliably, and, in large part we were significantly discommoded by not being able to talk to colleagues, technicians or even the producing parties' counsel while being seated in front of the stand-alone computer.

We have also seen cases where one forensic software analyst is in one city analyzing part of a software product, and, at the insistence of the producing party, another forensic software analyst has been forced to go to another city to analyze another part of the same software product. The only way to discern which were the relevant parts of the source code was to have these two forensic software analysts leaving their respective stand-alone computer rooms with hand-written and memorized information, speaking with each other on their cell phones, then returning to the stand-alone computer to refine their respective analysis. This continued for the better part of a week and created activity more in line with the Keystone Cops than with 21st century forensic software analysis—and, of course, the receiving party was footing the increased bill for the unnecessary burdens imposed by the overly-protective order. The simple solution is to permit the stand-alone

---

[77] *See supra* notes 70–73 and accompanying text. And, of course, the stand-alone computer is not connected to the internet. *See supra* Part IV.B, cl. m.

computer to be located at the forensic software analyst's office where access is not fettered by the day and time.

If, on the other hand, the parties stipulate to locating the stand-alone computer at a third party or law office, then the minimum cost solution for this presumed intent requires a multi-pronged approach:

a) Modify the stand-alone computer so that no devices are attached to it that can create copies (e.g., remove the DVD burner and replace it with a DVD reader, disable the unused USB ports, etc.); and

b) Permit the use of cell phones without built-in cameras in the stand-alone computer room and have a proctor be in the stand-alone computer room during any cell phone calls.[78]

Source code is such unusual text, so littered with symbols, that dictating it into a cell phone is impractical in the extreme, and if the proctor observes the phone call it will be immediately apparent if any kind of systematic dictation of source code is occurring.

d. Stand-Alone Computer(s): Hardware Configuration

In situations where a fundamental premise is that the stand-alone computer(s) is/are not provided by the forensic software analyst, typically there is language in the protective order to the effect of:

> The stand-alone computer must be sufficiently state-of-the-art (in terms of processor speed, memory, etc.) to support a review of the source code.

It is presumed that this clause will ensure that suitable computers are provided that will be up to the task of the analysis. However, unless the forensic software analyst is involved in the specification of the computers in question, there is only a small chance that they will have sufficient computing power and storage, as almost no attorneys and few computer scientists understand the intense computing load that forensic software analysis places on a computer system.

While the presumed intent of this clause appears well intentioned, in fact the clause is effectively meaningless. For example, a given computer could "support a review of the source code," but be so underpowered that that review might take four or six times longer than it needs to. The clause makes it appear that the producing party is cooperating with the other party without actually doing so.

---

[78] If the proctor is physically in the room for the cell phone calls, the proctor could also ensure that no images are being captured if the cell phone has a camera. The requirement of a proctor is discussed in more detail below. *See* discussion *infra* part IV.C.1.j.

We have repeatedly seen problems with computers that are underpowered for the task at hand. In one case, the computers were purchased from Fry's Electronics and chosen because they were the only computers available the day before the forensic software analysis was due to start. The computers were running Microsoft Windows Home Edition, which is a completely suboptimal choice for the heavy-duty computational requirements imposed by forensic software analysis. In another case, the hard disk of the computer was barely large enough to contain the *compressed* version of the source code, and nowhere near large enough to permit the source code to be completely uncompressed for analysis.[79] Purchasing computer hardware of sufficient computing power is no longer just a case of buying the hardware and starting to use it – the hardware must be thoroughly tested before being subjected to the typical forensic software analysis workload. Such workload will keep the central processing units running flat-out for days on end, and will subject the hard disks to intense activity.

Again, the best solution is to permit the analysis to occur at the analyst's office. The specification of the hardware configuration would not be required and the model clauses address the appropriate security and copying concerns. If the producing party is unwilling to permit the analysis to be performed at the forensic software analyst's office, the next best solution is to permit the analyst to arrange for suitable computer(s) to be shipped to the appropriate location, pre-configured, tested, and ready for use. Such hardware will have been selected to ensure that it is capable of the analysis task at hand and will have a fast enough central processing unit, sufficient random access memory, and large enough disk storage.[80]

---

[79] Data is represented in a computer in a somewhat inefficient way, taking up more memory space than it really needs. As a real-world human example, we use abbreviations for compression: RSVP, ASAP, OK, LOL, and so on. The same principles can be applied to computer source code using modestly priced (or even free) software that compresses an original file to a fraction of its size. For an introduction to the concept of data compression, see *Data Compression*, WIKIPEDIA, http://en.wikipedia.org/wiki/ Data_compression (last visited Oct. 26, 2011).

[80] The analysis computer must have an external disk drive of sufficient capacity to make backup copies of the interim results obtained during analysis. The use of an external backup drive means that the probability of failure is reduced as it is a separate drive from the primary one, and also means that results can be moved quickly and easily to a second computer if the first analysis computer fails. The analysis computers also must be provided with a sufficiently large capacity uninterruptible power supply (UPS) to permit the computer to keep running for enough time to perform a controlled shut down in case of a power loss. We have seen hours of analysis wasted due to a brief "power hit" induced by a thunderstorm, or by a proctor inadvertently spilling coffee on a power distribution bar. It is a completely false economy not to provide a UPS. There is an adage in the computer business that having only one backup is not enough: "You can never be rich enough, thin enough, have enough RAM, or enough backup copies." Therefore, considering the cost of the analysis, it would be prudent to purchase a second external backup disk to hold an additional redundant copy of the interim results of the analysis. The analysis computer also

The producing party may feel the need to provide "neutral" hardware, freshly purchased for the litigation, so that these computers can be destroyed after the litigation, thus ensuring that no trace of the source code outlives the litigation. This need, however, is technically incomprehensible. Similar thinking might also induce the producing party to object to providing dual large screen monitors or a sufficiently fast and robust laser printer. These objections are typically borne of a lack of appreciation for the complexity of reviewing complicated source code on computer screens and the need to have reliable, high speed printing capabilities.

If the true concern of the producing party is to ensure that every last trace of the source code is removed from the hard disks, those disks can easily and cheaply be replaced with brand new hard disks at the end of the litigation. Model clause r, above, requires such destruction.[81] There is no technical basis whatsoever for the producing party (or the receiving party, for that matter) to have to provide brand new, untested, and possibly underpowered computers for the analysis.

e. Controlling Printing and Copying of Source Code

A clear concern when dealing with source code (or anything else stored on a digital computer) is the ease with which copies can be made and propagated. Therefore, it is not unusual to see a clause that attempts to restrict copying:

> No electronic copies of the source code shall be made (e.g., creating Adobe Acrobat Portable Document Files, or photographing the source code, or capturing screen shots of the source code), other than the temporary copies necessary for analyzing the source code. Whenever possible, no printed or electronic images or copies of the source code shall be used in correspondence, pleadings, expert reports, deposition, and trial exhibits.
>
> In the event that paper copies of source code are made, no more than five copies shall be made.

---

requires a high speed, high quality printer. We acknowledge that there appears to be an internal contradiction insofar as we have previously stated that forensic software analysts abhor printing out source code. What has to be printed can often be documentation, or interim results, thus necessitating a printer attached to the analysis computer.

[81] *See supra* Part IV.B, cl. r.

We have also seen clauses that restrict the number of pages of paper printed copies:

> A receiving Party shall be allowed to make up to 500 pages of hard (non-electronic) copies of those portions of source code that it, in good faith, considers necessary to the preparation of its case, and may remove the hard (non-electronic) copies from the premises of the source code custodians. Notwithstanding the foregoing, a receiving Party may not make any hard copy of more than 20 consecutive pages of source code absent express permission of the producing Party or an order from the Court. The receiving party shall maintain a written log of those portions of source code that it considers necessary to the preparation of its case and of those pages it prints.

The presumed intent of these types of clauses is to reduce the risk that inappropriate copying of the source code occurs and the risk that a copy falls into unauthorized hands. A further inferred intent is that keeping a log file will permit oversight of the recipients of printed source code.

On the face of it, this language appears to be well meant, but in practice, when dealing with source code it is onerous in the extreme. For example, consider the case of a copyright infringement action. The plaintiff's expert performs a forensic software analysis and discovers large quantities of identical source code and large quantities of substantially similar source code. The first part of demonstrating that the source code is identical is easy: plaintiff's expert can use *plaintiff's* source code (which is not subject to the same restriction as defendant's source code) and, with plaintiff's approval, put that source code into pleadings, expert reports, depositions, and trial exhibits. The problem comes when dealing with substantially similar source code. How can such substantial similarity be communicated fairly without a side-by-side comparison? A list of file paths, file names, and source code line numbers simply does not demonstrate the nature and quantity of substantial similarity. No description of the level and nature of the similarity will bring home the point as well as a side-by-side comparison where identical fragments are shown, for example, with red highlighting, and substantially similar fragments are shown with yellow highlighting. Only then can the viewer obtain a quantitative and qualitative sense of what is meant by "substantial similarity."

Restrictions on the number of pages for printed copies are curious because it is somewhat unusual for a forensic software analyst to print out

source code. Printing out source code is a remedy of last resort.[82] Thus it is generally true that forensic software analysts, in common with the original computer programmers who wrote the source code, abhor the notion of printing out source code except for the purposes of creating exhibits for expert reports, deposition exhibits, and trial demonstrative exhibits.[83] It serves no purpose to print out more source code than is required for exhibits or for studying extremely complex logic.

Imposing limits on how much source code can be printed via a protective order appears, again, as a type of "security theater."[84] Very little security benefit is gained by this restriction, and the burdens such a restriction can impose are significant. Whatever amount is chosen as a limit is likely to be wrong because the amount will be chosen based on theory not on practice. The amount of source code printed will be determined by how much source code is relevant to the issues. Any arbitrary limits placed on the number of pages that can be printed simply miss the point and, by definition, will likely be wrong. Such limitations appear to offer some reassurance that wholesale printing will be prevented, but in practice the limitations only serve to increase the cost, as further negotiation between the parties' counsel will be required to increase the limits.

Finally, maintaining a log of the recipients of printed source code is also part of the security theater—it is unlikely that such a log would actually identify malfeasance. If there is going to be a leak, the leaker is unlikely to self-report his or her access to the source code. Additionally, computer access logs are far better at tracking every incident of access to files on the computer.

So long as the protective order ensures that only source code relevant to the dispute is printed out, as model clauses r and t provide, there need be no limits on the amount of source code that can be printed. The model clauses further provide for proper handling of those copies[85] and for

---

[82] The typical reason for needing to print source code and study it is where the source code is so complex, tangled, and widely spread across different source code files that the computer display(s) on the stand-alone computer act like a keyhole and prevent understanding of the larger picture of the programming logic.

[83] Additionally, all of the benefits of printing out source code on paper, with or without highlighting, can be obtained without compromising security if the source code is "printed" to an Adobe PDF file that is then either encrypted as a file or placed on a disk protected by disk-level encryption. Such a file can then be transmitted or hand carried; it can be filed with the court under seal; it can be placed in front of a deponent on a computer screen; or it can be displayed in a courtroom using a display projector—all without having to leave a decrypted version of the file on a computer system where it would be put at greater risk of falling into the wrong hands.

[84] SECURITY THEATER, *supra* note 57.

[85] *See* Part IV.B, cl. r, s, and t.

the destruction of the copies produced once the litigation has concluded.[86] Overcoming an attorney's lack of understanding of what an Adobe Acrobat PDF file is (although such understanding is now much more widespread), and the efficacy of strong encryption[87] are the biggest hurdles in convincing counsel for the producing party that overly restrictive clauses designed to limit copies are inappropriate and unnecessary.

f. Printing Source Code on Pre-Bates Numbered Paper

Until relatively recently, the practice of law has placed significant emphasis on paper-based information. Bates numbered paper has long facilitated tracking and referencing of documents in a paper-based environment. Therefore, the following overly protective clause has appeared in more than one protective order:

> The source code may only be printed out on paper using paper that has been pre-numbered with Bates numbers. A log file will be created showing the Bates number ranges of all such printouts with a description of the contents of the printouts.

We presume that the intent behind using pre-Bates-numbered paper is to ensure that all source code printouts are easily identifiable as they are created and easily referenced during the litigation. The presumption is also that the log file will permit all printouts to be accounted for.

Source code that is printed out just as it exists in the revision control system is effectively unusable for exhibits as it is cluttered with notations used by the revision control system and there are no easy-to-use reference markers by which attention can be drawn to a specific line of source code. From the computer science point of view, to identify a given line of source code unambiguously, one needs to identify the directory structure[88] in which the source code file is contained plus the line number of the source code line within the file that contains it. Bates numbers serve to identify a given page in the record, but are irreversible—given a Bates number, one cannot backtrack to the actual source code file and the specific line of source code.

---

[86] *See* Part IV.B, cl. s.8 and t.2.

[87] *See* discussion *supra* Part III.C.1. As described previously, no strong encryption is effective if the encryption keys are not managed properly. Key management, however, is less burdensome that managing physical printouts of source code and log files.

[88] Source code files are typically stored in a hierarchical directory structure such as /productName/version/functionalArea/sourceCodeFileName. The source code file name is an ambiguous identifier as it is not unusual for more than one file to have the same name, albeit in different functional areas.

Thus, Bates numbers do not provide the same assistance in locating particular material referenced in a subsequent report, pleading, or other document. Numbering blank paper is a remnant of "paper thinking" and leads to increased costs and unnecessary suspicions about the numbers when the almost inevitable paper jams occur.[89]

The minimal cost solution is to "print" all the source code to Adobe Acrobat PDF files and then number the electronic pages with Bates numbers before any physical printouts occur. Adobe itself provides specific instructions how to add Bates numbers to Adobe Acrobat PDF files.[90]

This solution allows electronic "pages" in the PDF files to bear unique Bates numbers that correspond exactly with the printed versions of those pages. As the numbering is electronic it can be done at high speed and without any need for preprinting, thus avoiding the multi-sheet feeding and paper jams that otherwise tend to occur. The typical objection will come from attorneys unfamiliar with PDF files and the advantages that they offer over physical paper.

g. Forensic Software Analyst May Not Study Printed Source Code

The desire to minimize printed source code manifests in several forms. This is one of the more unusual clauses we have seen:

> The forensic software analyst may print portions of the source code only when reasonably necessary to prepare pleadings, expert reports, deposition, or trial exhibits. The forensic software analyst shall not print source code as an alternative to analyzing that source code on the stand-alone computer.

---

[89] Significantly, using pre-Bates-numbered paper is a logistical nightmare for several reasons. First, when paper is passed through a laser printer to pre-print the numbering on it, the fusing rollers that heat the toner to fuse it into the paper, dry out the paper. This dryness then appears to increase the risk that, when printing the source code on this pre-printed paper, the printer will feed more than one sheet at a time, or cause a paper jam. Either of these events then causes discontiguity in the page numbering that raises suspicion that pages have been redacted for sinister reasons. Additionally, if not enough paper has been pre-Bates-numbered, the forensic software analysis must stop while more paper is pre-printed. This, of course, tends to happen when time is short and the delay is most inconvenient.

[90] ADOBE HELP RESOURCE CENTER, http://help.adobe.com/en_US/Acrobat/8.0/Professional/help.html? content=WS6DE1E376-6A82-406c-A711-6C5E5207A1F2.html (last visited July 19, 2011).

The presumed intent is to limit the amount of source code printed out on paper by only allowing that printed source code to be read in detail by people *other* than the forensic software analyst, even though that analyst is likely to base their testimony on what is on the paper. It is difficult to conceive of a legitimate concern that such a provision is attempting to address.

This clause displays a lack of understanding of the forensic software analysis process. Under normal conditions, an analyst does not require and will avoid printing out source code because it is so unmanageable. Printed source code is a remedy of last resort especially when it comes to source code that is extremely complex or tangled—and when that resort is needed, printing the source code is the only way to proceed.

The typical reasons for needing to print source code and study it will be where the source code is so complex, tangled, and widely spread across different source code files that the computer display(s) on the stand-along computer act like a keyhole and prevent understanding of the larger picture of the programming logic. If this clause is allowed to remain in the protective order, and the source code is complex, it will only serve to increase the time and cost of the forensic software analysis.

This restriction should not appear in properly drafted protective orders of appropriate scope. The appropriate protections for printed copies address the legitimate concerns.

h. Forensic Software Analyst May Only Take Handwritten Notes

One of the more restrictive clauses we have seen in protective orders is:

> The forensic software analyst may take notes relating to the source code but may not copy the source code into the notes and may not take such notes electronically on a computer.

The presumed intent is to reduce the probability that source code will be copied during the forensic software analysis.

This clause creates major inconveniences for the forensic software analyst, and is much like asking a landscaper to cut the lawn with nail scissors—it can certainly be done, but it drives the cost up by orders of magnitude. First, any computer-assisted analysis is going to create intermediary results files, and these files, depending on the number of lines of source code, could be huge. Second, making hand-written notes about source code is completely impractical. Source code contains long textual strings that must be written with absolute accuracy to withstand any expert

cross-examination. Third, source code can be incredibly difficult to deal with by writing handwritten notes. Consider this from a Microsoft example program: [91]

```
12  unsigned wHash=0;
13  pch=sz;
14  while (*pch!=0
15    wHash=(wHash<>11+*pch++;
16  cch=pch-sz;
17
pbsy=&rgbsyHash[(wHash&077777)%cwHash];
18  for (; *pbsy!=0; pbsy = &psy->bsyNext)
19    {
20    char *szSy;
21       szSy=  (psy=(struct  SY*)&rgwDic[*pbsy])-
>sz;
22    pch=sz;
23    while (*pch==*szSy++)
24     {
25     if (*pch++==0)
26       return (psy);
27     }
28    }
```

Unlike the example above, source code can also use some very long strings of characters, each of which must be transcribed absolutely accurately if they are to form the basis for an exhibit: [92]

```
static  void  writeTree(XmlNode  xmlElement,  int
level) {
  String levelDepth = "";
  for(int i=0;i<level;i++)
  {
   levelDepth += "  ";
  }

Console.Write("\n{0}<{1}",levelDepth,xmlElement.Name)
;
```

---

[91] *See* Charles Simonyi, *Hungarian Notation*, MSDN (reprinted Nov. 1999), http://msdn.microsoft.com/en-us/library/aa260976(v=vs.60).aspx (describing variable naming conventions).

[92] Matt Fincher, *Learning C*, MATT FINCHER'S HOME PAGE http://www.fincher.org/tips/Languages/csharp.shtml (last updated Jun. 14, 2012).

```
            XmlAttributeCollection xmlAttributeCollection =
xmlElement.Attributes;
            foreach(XmlAttribute x in xmlAttributeCollection)
            {
             Console.Write(" {0}='{1}'",x.Name,x.Value);
            }
            Console.Write(">");
            XmlNodeList xmlNodeList =
xmlElement.ChildNodes;
            ++level;
            foreach(XmlNode x in xmlNodeList)
            {
             if(x.NodeType == XmlNodeType.Element)
             {
               writeTree((XmlNode)x, level);
             }
             else if(x.NodeType == XmlNodeType.Text)
             {
               Console.Write("\n{0}
{1}",levelDepth,(x.Value).Trim());
             }
            }

Console.Write("\n{0}</{1}>",levelDepth,xmlElement.Nam
e);
          }
```

A clause permitting only handwritten notes is burdensome in the extreme. Modern computer source code was never intended to be handwritten even by the original programmer—so much so that a programmer will, in all probability, use Microsoft development programs that automatically complete many of the long strings of characters that occur in source code as they are too long to type accurately.[93]

The only rational minimum cost solution is to eliminate these inappropriate clauses. Given a reputable forensic software analyst whose reputation and future livelihood rests entirely on demonstrating his or her ability to handle proprietary source code responsibly and in compliance with a protective order, the risks of inappropriate copying are minimal and do not justify such a burdensome provision in a protective order.

---

[93] The autocompletion feature is called IntelliSense. For an overview see *Intellisense*, WIKPEDIA, http://en.wikipedia.org/wiki/IntelliSense, (last visited July 21, 2011). *See also, Autocomple*, WIKIPEDIA, http://en.wikipedia.org/wiki/Autocopletion, (last visited Oct. 26, 2011).

i. Source Code Access Logs

Given the importance of the source code on the stand-alone computer we often see an overly protective order clause of the form:

> A written log shall be created identifying those individuals who access the source code on the stand-alone computer. This log shall record the name of the individual, and the date and time of the access to the source code.

The presumed intent of such an access log is to police who accesses the source code on the stand-alone computer, thereby ensuring that there will be no unauthorized access to the computer.

The unintended consequences of such an access log are the overhead of creating and maintaining such a log. While the overhead is not great, there does not appear to be any significant benefit in creating a paper-based log when the computer operating systems can do this far more efficiently, accurately, and automatically. The inefficiency of the paper log is especially apparent when one considers that any signs of malfeasance are unlikely to be placed into a manually kept log.

By assigning each individual authorized to access the stand-alone computer a unique user account, the automated logging function records which user accesses which files along with the date and time of the access. Model clause n, above, requires the use of this logging capability.[94] The usual objection to abandoning manual logging, and instead relying upon the automated logging, is based on fear of the unknown—many computer scientists are unaware that the operating systems can actually provide this level of auditing/logging as these features are rarely used.

j. Proctors

We have seen an increasing number of situations where counsel for the parties have stipulated to the presence of a proctor to supervise the source code analysis. The duties of the proctor have ranged from being responsible for powering on and off the computers, entering the access passwords, or even visual supervision of the actual analytical process. In this latter case the protective order language was of the form:

> The producing party shall designate a proctor to visually monitor the activities of the receiving party's representatives during any source code analysis but only for

[94] *See supra* Part IV.B.

> the purpose of ensuring that no unauthorized copying of the source code occurs and no information about the source code is being created or transmitted.

The presumed intent of a proctor is to ensure that the forensic software analyst will not make unauthorized copies of the source code and remove them from the room containing the stand-alone computer. However, the concept of the proctor operating as a supervisor of the source code analysis process seems to be excessively paranoid, especially given that, in our experience, the people chosen to be proctors rarely are skilled computer scientists[95] and almost never maintain their vigil inside the same room as the stand-alone computer.

Expressed in its most brutal form, the proctors we have seen have not had the technical know-how, nor have they been in the right place,[96] to detect any copying of anything. When it comes down to it, the proctors we have encountered have placed their trust in the idea that that the forensic software analysts have agreed to the terms and conditions, and therefore do not attempt to copy the source code. We lament the fact that the proctors place their trust in the forensic software analysts, but the counsel for whom the proctors work do not.

When one considers that forensic software analysts base their reputations and their future careers on their integrity, one must ask whether there is any rational basis for such proctoring. A clause in a protective order requiring a proctor seems more a case of attempting to deal with "imaginary horribles" that exist only in the minds of counsel rather than in the real world. With "imaginary horribles" it is hard to overcome irrational fear with reason. As a practical matter, ineffectual proctors simply serve to slow the analysis process down and to increase the costs, with no perceptible benefit to either of the litigants. Nevertheless, counsel will argue for having a proctor in the irrational belief that such a proctor will act as the last bastion to protect their client's source code. In our real-world experience, reality rarely comports with this notion.

---

[95] In our experience it is not unusual that they be employees of a temporary agency and be paid minimum wage.

[96] The proctor usually is more of a gatekeeper—well, a doorkeeper—who sits outside the room that contains the stand-alone computer. There is usually only one proctor and he or she takes bathroom, lunch, and phone-call breaks without requiring that the source code analysis cease.

k. Forensic Tools

A recent trend in overly protective orders has been the appearance of language restricting the tools an analyst may employ in conducting the analysis. Here we identify by letter and number individual clauses for subsequent discussion:

> 1.a) The parties shall agree in advance on the software tools to be installed on the stand-alone computer for analyzing the source code.
>
> 1.b) If, at a later date, additional software tools need to be installed, the receiving party shall make a request to the producing party for such tools.
>
> 1.c) The producing party may object within three days of that request and if such an objection is raised the receiving party shall not install the additional software tools on the stand-alone computer.

An alternate form reads:

> 2.a) The producing party shall install software tools on the stand-alone computer that are sufficient for viewing and searching the source code, if such tools exist and are presently used in the ordinary course of the producing party's business.
>
> 2.b) The receiving party's outside counsel and/or experts may request that commercially available software tools for viewing and searching the source code be installed on the secured computer, provided, however, that such other software tools are reasonably necessary for the receiving party to perform its review of the source code.
>
> 2.c) Where executable source code [sic] is installed on the stand-alone computer, the receiving party shall be entitled to install and use appropriate compilers, debuggers and text editors so long as the receiving party agrees that no edits may be performed on the source code.
>
> 2.d) The receiving party must provide the producing party with the licensed software tools at least ten days in advance of the date upon which the receiving party wishes to have the additional software tools available for use on the stand-alone computer.

The presumed intent of these clauses is to afford the producing party control over what software tools are used to analyze the source code on the stand-alone computer. There seems to be no valid technical or legal reason why

the producing party should have such control over the means by which the source code is analyzed.

The source code exhibits in the expert report will speak for themselves regardless of what tools were used to prepare the report. The tools are therefore not the objects of scrutiny. In conducting expert analysis of computer source code it is the results, and only the results, that matter: does the exhibit demonstrate support for the assertion made by the forensic software analyst or not?

Controlling the means by which relevant evidence is located should be outside the purview of the producing party. It is both irrelevant and inappropriate that the producing party should have any say in *how* the forensic expert's investigation and analysis is completed. The producing party will have a full opportunity to dissect and rebut the *results* of that analysis. Additionally, if it is relevant in a particular case, through deposition or at trial the producing party may inquire about the methods employed to arrive at the results. However, controlling the methodology *ex ante* is inappropriate.

Contrary to what clause 1.a above suggests, it is usually impossible to predict the specific software tools that a forensic software analysis will require until the forensic software analyst has examined the produced source code in some detail. If a clause like 1.a is in place, the receiving party will be deliberately over-inclusive in the list of desired software tools rather than be penalized with delays imposed at the whim of the producing party. Such over-inclusiveness might provoke the producing party to protest and inappropriately shift the debate to the issue of the *number* of the tools, rather than what the debate should focus on: the *probative findings* of the tools.

The alternate form of clause 1.a, clause 2.a, is predicated on the fact that all of the software tools to be installed on the stand-alone computer are presently in use in the "ordinary course of the producing party's business." This is clearly a specious presumption that has little hope of being true even if the producing party happens to be in the forensic software analysis business. Such a clause is so off the mark that, to quote Wolfgang Pauli, "it's not even wrong."[97]

Clause 2.b is predicated on the notion that all likely forensic software analysis tools are commercially available rather than (a) being publicly available free software, or (b) being proprietary tools developed by the forensic software analyst. Again, this clause fails to rise to the level of

---

[97] Michael Schermer, *Wronger Than Wrong: Not All Wrong Theories are Equal*, SCIENTIFIC AMERICAN (Oct. 16, 2006), http://www.scientificamerican.com/article.cfm?id=wronger-than-wrong.

even being wrong. Certain software tools are perfectly viable forensic software analysis tools but are no longer commercially available.[98] Additionally, forensic software analysts routinely create additional software tools either *ad hoc* or as augmentations or tailoring of standard third party software tools.[99] This augmentation or tailoring can represent the stock-in-trade of the forensic software analyst and are therefore proprietary products in their own right.

Paragraph 2.c is a further curiosity because of the internal contradiction of "executable source code." A computer cannot execute source code. That aside, it is curious that the presence of this mythical executable source code is the predicate condition for being able to use software to compile (translate source code into executable *object code*) and then debug that object code.

Paragraph 2.d exemplifies a producing party wishing to overreach by having ten working days to consider whether a particular software analysis tool can be installed. As previously stated, it is inappropriate to confuse the software tool with the results that it produces. Furthermore, providing the producing party with the power to delay the analysis by ten days for each product grants the producing party powers which are inconsistent with the spirit, if not the letter, of the Federal Rules of Civil Procedure governing discovery.[100]

A further presumed unintended consequence of these clauses is that the cost of licensing the necessary software tools must be borne (usually) by the receiving party. The typical software tools required for source code analysis can easily total several thousands of dollars per computer. This is needlessly expensive if the forensic software analyst already has licenses for the required software tools.

The only rational minimum-cost solution is to eliminate these inappropriate clauses and focus on the results, not the means by which they were obtained. The forensic software analyst should be free to install whatever tools are required to complete the analysis cost-effectively. The expert witness will be exposed to the rigors of cross-examination at deposition and at trial to determine whether the resulting exhibits, and the expert opinions based upon them, are valid.

---

[98] The Thompson Toolkit is one example of viable software no longer available. *See, e.g.*, THOMPSON AUTOMATION SOFTWARE, http://www.tasoft.com/ (last visited July 16, 2011).

[99] *See supra* Part II.B.2.

[100] *See supra* note 5.

As we are at a loss to understand the presumed intent of the producing party to control the forensic software analysis tools, it is hard to speculate as to what objections the producing party might have to abandoning such control. The primary objection to denying the producing party this degree of scrutiny over the analysis tools may be that the producing party will not know how the results were obtained. While this is true, what matters more is whether or not the exhibits produced as a result of the analysis meet the burden of providing a basis for the expert opinions regarding infringement or misappropriation. Source code analysis is not subject to the same issues of reproducibility as, say, chemical experimentation or physical experiments[101]—it is purely textual: either the source code cited in the expert's report exists in the code and provides adequate basis or it does not. Moreover, knowing how such infringement or misappropriation was detected might provide an opportunity for an unscrupulous party to hide their unlawful behavior in the future.

l. Prohibition on Compiling the Source Code

We have seen the following paragraph in a protective order:

> The source code will be produced for inspection in native format or in an electronically-searchable form, together with all necessary libraries and make files necessary to compile the source code, but the receiving party shall not compile the source code or any portion thereof.

The presumed intent is to ensure that all of the source code files and libraries of previous compiled source code are provided, hence the notion that all necessary libraries and header files will be produced. However, the prohibition on actually compiling the source code is mystifying.

This prohibition results in the receiving party having no practical, cost-effective way of determining whether all of the source code for a product has been produced.[102] It bears repeating that there is no cost-

---

[101] In one case, the producing party suggested as an alternative that its representative be allowed to watch the forensic software analyst repeat the use of the software tools to see if it produced the same results twice. This suggestion is as bizarre as it is futile for it does not test the accuracy of the source code exhibits, but merely the reproducibility of the software tool's results.

[102] A program such as Understand can be of some use to determining missing source code files, but only the act of actually trying to convert the source code into an executable program is truly dispositive. Understand works strictly with source code files. It cannot and does not check to see if any third party components, linked together with the output of the compiled source code, are present or not. UNDERSTAND, *supra* note 25.

effective alternative to compiling the source code to determine the completion of a source code production.[103] Additionally, beyond assessing the completeness of production, for all causes of action there are reasons why compilation and execution of the resulting program is a valid task for the expert to perform—all of them relating to the dynamics of the behavior of the program as it executes, rather than to the static expression in the source code. In cases involving assertions of copyright infringement, the displays created on the computer screen—the audio/visual displays, the printouts, or the on-line help—may be probative of the issue of inappropriate copying. In cases involving allegations of patent infringement or trade secret misappropriation, the claimed invention or asserted trade secret may be, or may include, the behavior of the resulting executable program rather than the static aspects of the source code.

Furthermore, there is a curious alternative provided by this language: source code in "native format or in an electronically searchable form." These alternatives, without showing their hand, cover everything from the actual format in which the source code is created and maintained to printing out the source code, scanning in these printouts, and then converting the printouts to text using optical character recognition.[104] These PDF files can be searched, but not with the same facility and with nowhere near the accuracy as the original source code. Again, one wonders why a producing party would go to so much trouble to "produce" this form of the original source code unless it is to hide incriminating evidence or to significantly increase the discovery costs of the receiving party.

The only rational minimum-cost solution is to eliminate these inappropriate clauses.

2. Stand-Alone Computers Not at the Analyst's Location: Key Issues Usually Omitted from the Protective Order

We have seen some incredibly burdensome situations develop because insufficient attention was paid to the stand-alone computer and/or the logistics of using it at a remote site. As described in detail above,[105]

---

[103] *See supra* Part II.B.1.

[104] This latter method of creating "electronically searchable files"—one we have seen quite often even though it is fraught with the errors that optical character recognition produces—is made all the worse by the fact that source code is completely unlike normal written English and the optical character recognition software thus completely botches the conversion to text. However, such garbled text falls within the definition of "electronically searchable," even though the text is little more than an artist's impression of the original source code.

[105] *See supra* Part IV.C.1.

requiring that the computer be located anywhere other than the analyst's office significantly increases the cost of discovery with little gained in return. It thus suggests the need for additional language in the protective order to reduce the burden and expense of using a remote location if a remote stand-alone computer must be used.

a. Administrative or User Accounts

We have often seen the producing party use the strategy of creating a user account for the forensic software analyst that merely has the access privileges of an ordinary user. The producing party argues that the forensic software analyst has no need for administrative privileges. Unfortunately, this argument is based on ignorance of the fact that many of the software tools used for forensic software analysis can only be run if the account being used has administrative privileges. Thus, such a restriction stops the analysis dead in its tracks. Lack of administrator privileges causes delays as counsel for the producing party must understand the issue and then realize that their positing is technically invalid and untenable. We therefore recommend the inclusion of this clause in any protective order requiring a remote site computer:

> On the stand-alone computer the forensic software analyst will have a user account with administrator privileges to permit the analysis to proceed unimpeded.

b. Operating System

The operating system installed on the stand-alone computer must be fit for the task it will be called upon to perform. We have seen hastily purchased stand-alone computers with Microsoft XP Home installed on them. Such "toy" versions of this operating system have no place in the context of forensic software analysis. As it takes several days to install and test the operating system and the required forensic tools, the operating system must be identified in detail in the protective order, otherwise the moment the forensic software analysis starts, the operating system goes into meltdown under the load. The actual choice of operating system will be one that can only be determined by the forensic software analyst, but a typical protective order clause might be:

> The operating system installed on the stand-alone computer shall be Microsoft Windows XP 64-bit with all current upgrades installed and tested.

c. Proprietary Third Party Software

Proprietary third party software will be required to perform the analysis and, depending on the restrictions imposed on the analysis, to

record the results, prepare exhibits, and create Adobe Acrobat PDF files in lieu of physical printouts.[106]

A brief list of the software likely to be required might be:

a) Revision control software (to access the source code).
b) File compression/decompression (to expand out compressed source code archives).
c) File decryption software for those files produced in encrypted form (along with appropriate decryption passwords).
d) Specialized tool systems, e.g. Cygwin, UNIX command-line tools, and scripting for Microsoft Windows. These will be used to run prepared analytic processes (called scripts).
e) Specialized scripts to effect the analysis.
f) Microsoft Office (usually Microsoft Word and Excel are all that are required).[107]
g) Adobe Acrobat Pro (to create PDF files and Bates number them).
h) Understand – a source code navigation tool.
i) Beyond Compare – a text file comparison tool.
j) Adobe Photoshop (if any images need to be examined).

Most of the software tools identified above are subject to paid license agreements and therefore the parties need to consider (a) who will purchase the licenses, (b) in whose name the software will be registered, and (c) at the end of the discovery period, who will de-active the licenses and make them available for use on other computers.

The software identified above (or alternatives for Apple or Linux operating systems) take a significant amount of time to install and test. Such installation and testing should not be taken lightly; we have seen weeks wasted because the stand-alone computers were inoperable or unreliable because of mistakes made during the software installation.

---

[106] *See* discussion *supra* Part IV.C.1.e.

[107] There are some additional add-in tools for Excel that are useful for forensic analysis, such as Ablebits. ABLEBITS, www.ablebits.com (last visited July 21, 2011).

We would recommend that the following language be used as a model:

> The producing party shall provide the stand-alone computer(s) and shall install the following list of computer software on the computer(s): <Insert list of software>.
>
> The licensing fees for the software shall be paid by <producing/receiving> party, and, on conclusion of this litigation, the software shall be deactivated and the licensing information and original media and documentation shall be made available to <producing/receiving> party so that the software can be redeployed on other computers.
>
> Following the initial installation of the software identified above, the stand-alone computers shall be tested running third party stress tests to ensure that the computer hardware and operating system are correctly installed (this test shall be run for 24 hours continuously), and each individual software package shall be tested to ensure that it has been installed correctly, registered correctly, and operates correctly.
>
> Once testing has confirmed that the stand-alone computer hardware and software is operating correctly, a hard disk backup image will be created to avoid the need to re-install the operating system and application software in the event of a catastrophic hardware or software failure.

There is an absolute need to create a backup copy of the stand-alone computer once the operating system and applications have been installed. When the stand-alone computer fails, this backup copy can be used to configure replacement hardware in a matter of minutes rather than days.

d. Technical Support

Even with the best of intentions, and with high-quality hardware and software, the stand-alone computer is likely to exhibit problems at some point during the analysis. Unlike most typical day-to-day use, forensic analysis puts an extreme load on a computer, increasing the probability of hardware and/or software problems.

Given that the stand-alone computer is not at the forensic software analyst's office, it is imperative that there be a well-defined plan to repair the computer hardware or deal with software related issues. On more than

one occasion we have seen the stand-alone computer simply stop working and need to be replaced, causing delays of a week or more.[108]

Thus the protective order should describe a procedure for providing technical support:

> In anticipation of hardware or software issues with the stand-alone computer, producing party will provide continuous technical support to the forensic software analyst during the analysis.
>
> In the event of any failures with the stand-alone computer or associated hardware, the forensic software analyst will call <name of person> at the following numbers: <office number, home number, cell phone number, email/texting address>. If <name of person> is not available, then the backup person shall be: <name of person, office number, home number, email/texting address>. The producing party shall use all reasonable effort to remediate the problem within 24 hours. If this is not possible, rather than delay the forensic software analysis, the computer hardware and software will be replaced.

e. Physical Environment for Stand-Alone Computer

All too often we have found that the stand-alone computer is placed in an office where the heating or cooling systems have been turned off during the hours of the forensic software analysis, e.g., during evenings and weekends. We have endured temperatures as high as 105°F and as low as 40°F, neither of which are appropriate for the kind of detailed analytical work at hand. Therefore, we would recommend the following clause in the protective order:

> During the period when the forensic software analyst is performing the analysis the heating, ventilation, and air conditioning system will be kept running to ensure comfortable working conditions.

[108] In one case the delay was exacerbated by the fact that the protective order had failed to require the creation of a working backup of the stand-alone computer once the operating system and application software had been installed and tested.

3. Stand-Alone Computers Located At Forensic Software Analyst's Office

a. Network and Internet Connections

Prohibiting the stand-alone computer from being connected to the internet or existing internal networks is an appropriate solution to guard against unauthorized access via the "back door" of the Internet or an Intranet in the same office. However, protective orders that ban network connection completely can present problems for appropriate forensic analysis of the software. For example, a typical clause might provide:

> Stand-alone computer shall not be connected to a network of any kind (e.g. local area network, intranet or the Internet.)

The presumed intent of this language is to ensure that the stand-alone computer is truly isolated from other computers from which it could be attacked and from which unauthorized access to the source code might occur.

However, in some circumstances a limited network may be appropriate. For example, if the amount of source code is sufficiently large, more than one computer may be required to perform the necessary forensic software analysis. If this is the case, it will be far more cost-effective to create a diminutive local area network that links together these analysis computers and one or more printers. Provided that this network is created out of physical cables between the computers and printers rather than using wireless technology, there is no increase in the risk of unauthorized access. However, by insisting that no local area network can be created, the results from the analyses cannot be combined on a single computer, nor can a single printer be shared between the computers. The lack of such a local area network between the analysis computers serves to slow down the analysis and increase the time and cost required with absolutely no gain in security.

Model clause q, above, provides for the appropriate security protection while at the same time permitting the analyst's work to proceed in a cost-effective manner.[109] The producing party typically objects because of the misplaced perception that any kind of network increases the risk of inappropriate access to the source code. Provided that the network is confined to the same room as the analysis computers and is created using

---

[109] *See supra* Part IV.B, cl. q.

data cables and not wireless connections, this perception does not comport with reality.

b. Stand-Alone Computer Located at Expert's Facilities: Prohibitions on Printing

As discussed above,[110] concern with wholesale printing of proprietary source code is valid and it is proper that printouts of source code should be considered in a well-drafted protective order, but we have seen such restrictions as:

> Stand-alone computer shall not be connected to a printer. Expert shall identify those parts of files that are to be printed.

The presumed intent of such a clause is similar to prohibitions on printing copies found in orders where the computer is not located at the forensic analyst's office: the producing party can both monitor and control what source is printed. Implicitly that control would permit the producing party to limit the amount of material that is printed. Presumably the fear is that the more printed copies that exist, the greater the security risk that a copy might fall into the wrong hands.

Printing by request only significantly increases the workload of the forensic software analyst. The analyst must first request that printouts be made. When the printouts arrive, the analyst must verify that everything that was requested indeed has been printed.[111] Next, depending on the amount of source code that needs to be analyzed, this prohibition on printing can add significant time and cost to the analysis. The amount of source code to be printed will be higher because the forensic software analyst will tend to be over-inclusive rather than run the risk of having to request additional portions be printed, and then wait for the additional print-outs to arrive,[112] if the analysis of the printed source code reveals the need for additional source code.

The minimum-cost solution is to have the forensic software analyst "print" source code to PDF files that are stored on an encrypted external hard disk. These PDF files can then be physically printed out as required. If the PDF files are preserved and copied to another external encrypted hard

---

[110] *See supra* Part IV.C.1.e.

[111] We have seen situations where entire files have not been printed, or printer jams have caused one or more pages of source code be omitted from printouts.

[112] We have seen a delay of a week or more between the request for source code and the arrival of the printouts.

disk that can be sent to the producing party on a regular basis, then the producing party can monitor which files are being printed without the unintended consequences of slowing down the analysis and increasing the costs. Model clauses s and t, above, address the need for printing and establish this method for monitoring the printing activities of the software analyst.[113]

The typical objection to this strategy is a loss of control over what is printed and the volume of such printing. However, the producing party can still monitor the quantity of files printed without delaying the analysis and increasing the time and cost of that analysis.[114] The legitimate concern for security is addressed by the model clauses that require all printouts to be treated extremely carefully by securely locking them away when not in use, and destroying them at the termination of the litigation.[115]

4. Transmission of Source Code and Expert Work Product

As discussed above, it is appropriate to be concerned with the security of the source code and documents containing source code. Transmission of the source code may occur not only upon initial production, but also there may be portions of source code in the expert's reports or other pleadings. Thus it is not unusual to see a clause like this:

> Anyone receiving source code or documents containing source code will transport these documents either by hand, FedEx, or other similarly reliable courier. The source code and documents will not be transmitted by email, FAX, or other electronic means.

The presumed intent is to protect the source code from unauthorized disclosure post-production by preventing it from falling into the wrong hands while it is being shipped or transported. The unintended consequence is that there will be delays caused by shipping printed materials or computer

---

[113] *See supra* Part IV.B, cl. s and t.

[114] If there is a compelling need for the producing party to have near-real-time visibility of what is being printed, encrypted copies of the PDF files can be transmitted electronically to the producing party at the end of each day—recall that it is the encryption that provides security, not physical access. If the concern is that email is an unsafe means for transmitting encrypted source code, then a cloud-based, encrypted file sharing system such as SpiderOak may provide a viable alternative. With SpiderOak the encrypted PDF files are encrypted again both for transmission to and from SpiderOak's servers and remain encrypted while stored on those servers. *See* SPIDEROAK, https://spideroak.com/ (last visited July 17, 2011). SpiderOak is presently the only cloud-storage system to use encryption in this way.

[115] *See supra* Part IV.B, cl. q, r, and s.

media physically rather than electronically. We have seen weather related delays of several days during the summer and winter with both FedEx and UPS. Accidents or simple mistakes, such as failing to specify Saturday delivery or misaddressing, can also delay packages.

Furthermore, this clause fails to consider that there may be people at the producing or receiving party's expert's office who are less scrupulous than either party's counsel, who will copy the source code, which is in plain text on paper, before it goes into a FedEx envelope or after it has emerged from it.

Using strong encryption, electronic transfer of documents that contain portions of source code carries less risk of misappropriation than physical transport. The source code is unusable from the moment the sender encrypts it until the recipient decrypts it, regardless of any nefarious hands through which it might pass.[116] At the same time that the risk is reduced, the delay is also eliminated. Instead of taking two or three days to move atoms around the country, the task can be accomplished in two or three minutes by moving electrons. Electronic mail, while subject to some delays, is not the only option. Cloud-based files transfers using Dropbox[117] or SpiderOak[118] are faster and more secure[119] alternatives than electronic mail and allow for file transfers of up to 100GB and more than 100GB (in 100GB increments) respectively.

The most usual objection that encryption is not sufficiently secure is usually stated as sophistry: "There is no such thing as absolutely secure encryption." However, such thinking must be taken in context. The problems is that the physical documents themselves contain source code in clear text, easily readable by anyone who happens to see them, and at the sending and receiving end they can be mishandled, perhaps even sent (albeit reliably) to the wrong recipient.

In sum, given proper encryption key management,[120] encryption provides better sender-to-recipient security than shipping source code in plain text, even when using a reliable shipping means.

---

[116] *See supra* Part III.C.1.

[117] DROPBOX, http://www.dropbox.com (last visited July 17, 2011).

[118] SPIDEROAK, *supra* note 114.

[119] The additional security comes, in SpiderOak's case, from an additional layer of encryption as files are uploaded from the sender, stored on SpiderOak's server, and then downloaded from the server to the recipient. Additionally, SpiderOak uses a more secure and rigorous authentication scheme before permitting access to user data on the server. *Id.*

[120] *See supra* note 53 and accompanying text.

V. CONCLUSION

When the complicated world of computer software intersects with complex intellectual property litigation, having appropriately scoped discovery and protective orders will assist in minimizing the costs associated with discovery. We have seen a clear relationship between the number of restrictions placed on forensic software analysts and the cost of the resulting analysis. These restrictions appear to have become "fashionable" only relatively recently, seemingly related to counsels' increased, but still partial, understanding of the analytical process needed in computer software cases. We assume that the motivation for these restrictions is well meant, however, several of them leave us puzzled. Adhering to the wisdom of not attributing to malice that which can be explained by a lack of competence, we have sought to increase the reader's competence concerning the true effect of overly restrictive protective orders.

Attorneys should pay careful attention to the provisions addressing the requirements of production and analysis of computer software. Additionally, attorneys must understand the consequences of the clauses contained in protective orders in these types of litigation. As described in this article, it is possible to provide robust protection for disclosed source code while at the same time not unnecessarily increasing the cost of discovery by weaponizing the protective order.