INFORMATION SECURITY VULNERABILITIES: SHOULD WE LITIGATE OR MITIGATE?

by
Jeffrey D. Neuburger
Maureen E. Garde
*Brown Raysman Millstein
Felder & Steiner LLP*

Washington Legal Foundation
Critical Legal Issues
Working Paper Series No. 121
March 2004

Visit us on the Web at
www.wlf.org

TABLE OF CONTENTS

ABOUT WLF'S LEGAL STUDIES DIVISION

The Washington Legal Foundation (WLF) established its Legal Studies Division to address cutting-edge legal issues by producing and distributing substantive, credible publications targeted at educating policy makers, the media, and other key legal policy outlets.

Washington is full of policy centers of one stripe or another. But WLF's Legal Studies Division has deliberately adopted a unique approach that sets it apart from other organizations.

First, the Division deals almost exclusively with legal policy questions as they relate to the principles of free enterprise, legal and judicial restraint, and America's economic and national security.

Second, its publications focus on a highly select legal policy-making audience. Legal Studies aggressively markets its publications to federal and state judges and their clerks; members of the United States Congress and their legal staffs; government attorneys; business leaders and corporate general counsel; law school professors and students; influential legal journalists; and major print and media commentators.

Third, Legal Studies possesses the flexibility and credibility to involve talented individuals from all walks of life — from law students and professors to sitting federal judges and senior partners in established law firms — in its work.

The key to WLF's Legal Studies publications is the timely production of a variety of readable and challenging commentaries with a distinctly common-sense viewpoint rarely reflected in academic law reviews or specialized legal trade journals. The publication formats include the provocative COUNSEL'S ADVISORY, topical LEGAL OPINION LETTERS, concise LEGAL BACKGROUNDERS on emerging issues, in-depth WORKING PAPERS, useful and practical CONTEMPORARY LEGAL NOTES, law review-length MONOGRAPHS, and occasional books.

WLF's LEGAL OPINION LETTERS and LEGAL BACKGROUNDERS appear on the LEXIS/NEXIS® online information service under the filename "WLF." All WLF publications are also available to Members of Congress and their staffs through the Library of Congress' SCORPIO system.

To receive information about previous WLF publications, contact Glenn Lammi, Chief Counsel, Legal Studies Division, Washington Legal Foundation, 2009 Massachusetts Avenue, NW, Washington, D.C. 20036, (202) 588-0302. Material concerning WLF's other legal activities may be obtained by contacting Daniel J. Popeo, Chairman.

ABOUT THE AUTHORS

Jeffrey D. Neuburger is a partner in the New York office of Brown Raysman Millstein Felder & Steiner LLP and is the Chair of the firm's Information Technology Practice Group. **Maureen E. Garde** is an Associate at the firm and a member of that practice group. The opinions expressed herein are not necessarily representative of the opinion of the firm.

INFORMATION SECURITY VULNERABILITIES: SHOULD WE LITIGATE OR MITIGATE?

by

Jeffrey D. Neuburger & Maureen E. Garde
Brown Raysman Millstein Felder & Steiner LLP

INTRODUCTION

The latest e-mail virus, "Mydoom" made its rounds of corporate computers this January, bringing with it another flurry of editorials and op-ed pieces suggesting that the solution to the problem of computer network vulnerabilities is litigation against software developers.[1] Proponents of litigation often argue that subjecting software developers to lawsuits for damages caused by malicious hackers will motivate the developers to improve the security of their software products.[2]

This line of argument has the appeal of simplicity, and, at least from a superficial perspective, some basis in logic. However, closer analysis suggests that it is unclear whether there is always a constructive relationship

[1] *See, e.g.*, Murphy, *Commentary: Software Vulnerabilities and the Future of Liability Reform*, LINUXINSIDER (Jan. 22, 2004).

[2] *See, e.g.*, Bruce Schneier, *Liability and Security*, CRYPTO-GRAM NEWSLETTER (Apr. 15, 2002) (taking the position that improvements in network security are dependent "wholly on how quickly security liability permeates cyberspace.")

between liability litigation and product improvement.[3] In addition, the argument ignores the fact that software developers are not by any means operating in a liability-free zone under current law. Software developers are, in fact, responsible for flaws in their products, but perhaps not to an extent that may satisfy certain constituencies.

This WORKING PAPER briefly explores the state of the law with respect to a software developer's potential liability for information security vulnerability. It will examine this issue from the perspective of liability under contract law principles as well as under product liability theories. Assessing the level of risk that a vendor actually does face today, the paper poses a question — will subjecting a vendor to higher liability risk achieve the desired outcome — a safer software environment?

I. CONTRACTUAL LIABILITY

A. Is Software a Good or a Service?

Vendors of computers, computer systems, and software, like vendors of other goods and services, may be liable to their customers for breach of contract if their goods or services fail to meet contract specifications.

[3] While concern over liability lawsuits may have been a factor in some beneficial product improvements, for example, safer child automobile restraints, some believe that it also can be counterproductive to efforts to improve health and safety. *See, e.g.*, Ufkes, *When Is Compliance Necessary? - Pharmaceutical Manufacturers and Prop. 65*, PHARM.

Contracts and agreements between vendors and purchasers are governed by two parallel bodies of commercial law: Article 2 (Sales) of the Uniform Commercial Code, and the common law applicable to services contracts. Potential liability to customers for software defects may attach under either body of law.

Transactions in products, or "goods," such as computer hardware, are covered by UCC Article 2, while transactions involving software alone fall under the general common law applicable to non-goods transactions. But transactions involving software often involve both hardware and software. In transactions involving both hardware and software (for example, a "turnkey" computer system), some courts have applied Article 2 of the UCC. Thus, for example, in *Chatlos Systems, Inc. v. National Cash Register Corp.*, 479 F. Supp. 738 (D.N.J. 1979), the acquisition of computer hardware and software, together with computer programming services, was treated as a sale of goods covered by the UCC "notwithstanding the incidental service aspects" of the transaction. And in *Advent Systems v. Unisys Corp.*, 925 F.2d 670 (3d Cir. 1991), the court concluded that software should be treated as a good, subject to the UCC because such treatment "offers substantial benefits" to litigants in the form of "a uniform body of law on a wide range of

& MED. DEVICE LAW BUL. (July 24, 2003) (reporting on a California Prop 65 suit against nicotine patches).

questions likely to arise in computer software disputes," such as implied warranties, consequential damages and disclaimers of liability.

In transactions in which software is treated as a good, Article 2 of the UCC imposes certain warranties, and regulates the extent to which a vendor may disclaim those liabilities and limit a purchaser's remedies for breach of those warranties. Under the implied warranty of merchantability applicable to transactions in goods, a product must be fit for ordinary use, conform to claims on the product packaging, and "pass without objection in the trade." UCC § 2-315. An implied warranty of fitness for a particular purpose may also arise if the vendor has reason to know the purpose for which the product will be used and that the vendee is relying on the skill of the other party to select or furnish suitable goods. UCC § 2-314. The UCC also governs express warranties in transactions in goods: if a vendor makes an affirmation of fact or promise relating to goods and those affirmations or promises become part of the basis of the bargain, an express warranty is created that the goods will conform to the affirmation or promise. UCC § 2-313.

Under the UCC, implied warranties may be disclaimed or modified if the disclaimer or modification is in writing and is conspicuous. However, the UCC also contains a backstop provision that invalidates certain disclaimers and limitations of remedy. UCC § 2-719 invalidates disclaimers that "cause an exclusive or limited remedy to fail of its essential purpose," as well as limitations on consequential damages, if they are "unconscionable."

Software developers, like most other vendors, usually seek to disclaim implied warranties; the extent to which they succeed in doing so is catalogued in numerous judicial opinions. If the vendor satisfies the UCC disclaimer requirements, then the disclaimer and accompanying limitations on remedies may be enforceable, just as it would be with respect to other types of transactions in goods. Thus, in *M.A. Mortenson Co. v. Timberline Software Corp.*, 140 Wn.2d 568, 998 P.2d 305 (2000), a purchaser's damages for defective software were limited to the purchase price of the software in accordance with the integrated agreement entered into between the parties.

But a court may invalidate disclaimers of warranty and limitations of remedy under UCC § 2-719 if it finds that an exclusive or limited remedy has failed of its essential purpose. For example, in *Caudill Seed & Warehouse Co. v. Prophet 21, Inc.*, 123 F. Supp. 2d 826, 832 (E.D. Pa. 2000), the court dealt with the unsettled question of whether a disclaimer of implied warranties should be invalidated under UCC § 2-719 where a software licensing agreement also limited the purchaser's remedies to repair or replacement of the product, and the vendor failed to repair or replace defective software.

In software transactions that are not governed by the UCC, courts will also enforce disclaimers of warranty and limitations of remedies, although the roadmap to that result may not be as well-defined as it is under the codified law of the UCC. State common law and other laws vary, and along with them the extent to which disclaimers and limitations on remedies will be enforced.

B. Negotiated Software License Agreements

Particularly in a negotiated software license agreement, the nature of the warranties offered by a software developer will likely be tailored specifically to the agreement between the parties, as will the extent and availability of various remedies for breach of those warranties. A warranty measuring system performance according to highly specific criteria may replace any implied warranties, for example, and the remedy for breach may be limited to a mitigation or rebate of certain fees. Such negotiations are common in enterprise software licensing agreements and the resulting contracts will often be enforced according to their terms. For example, in *i2 Techs., Inc. v. Darc Corp.*, 2003 U.S. Dist. LEXIS 16655, 16-17 (N.D. Tex. Sept. 23, 2003), the court enforced a disclaimer of implied warranties as well as limitations on other remedies where it found that the parties were sophisticated business entities familiar with software and consulting agreements, and the agreement was the result of protracted negotiations.

C. Consumer Protection Laws

In "vendor-to-consumer" transactions, the ability of a software developer to disclaim warranties and limit remedies may be governed by federal and state laws that apply specifically to transactions with consumers. The Magnuson Moss Warranty-Federal Trade Commission Improvements Act, 15 U.S.C. § 2301-2312, which establishes minimum standards for consumer product warranties, may apply to software sold to consumers. The applicability of the Act to software is not currently clear[4], although many software developers that market software to consumers comply with its provisions on the assumption that it does, or may, apply.

Like many other enterprises, software developers have been the target of suits brought under a variety of state consumer protection laws. The distinctions drawn in other bodies of law between computer hardware and computer software are irrelevant in many such cases under these broadly-drawn statutes. The California False Advertising Act, CAL. BUS. & PROF. CODE § 17500, for example, covers false or misleading statements in transactions involving goods, real property and "services, professional or otherwise," and the California Consumers Legal Remedies Act. Cal. Civ. Code § 1750, covers unfair business practices, unfair competition and false advertising in any

[4] The Federal Trade Commission held a symposium in October 2000 on " Warranty Protection for High-Tech Products and Services," at which this topic was addressed.

"sale or lease of goods or services to any consumer." In *Wershba v. Apple Computer, Inc.*, 91 Cal. App. 4th 224 (Ct. App. 6th Dist. 2001), the court approved a multi-million dollar settlement of class action claims alleging violations of these statutes when the company altered its support policies for both hardware and software products.

These California statutes underlie class action claims recently brought against Apple Computer alleging misrepresentations concerning the battery life of its iPod digital music player and against Microsoft with respect to alleged security flaws in the Windows operating system. In the very broadly drawn complaint in the latter case, *Hamilton v. Microsoft Corp.*, (Cal. Super. Ct. L.A. Cty), the plaintiff, on behalf of a class of plaintiffs, alleges that the ubiquity of Microsoft software coupled with its security vulnerabilities "has created a global security risk." The proposed class representative seeks, among other things, damages for financial losses she claims to have suffered as a result of identity theft that occurred due to the "failure of Microsoft to provide adequate security."

California's consumer laws are notoriously broad in their applicability, but other states have similar consumer protection laws that span the product/service distinction and encompass defective software claims that allege some false, misleading or deceptive act. *See*, for example, the Texas Deceptive Trade Practices-Consumer Protection Act, TEX. BUS. & COM. CODE § 17.41 et seq. In Texas, by the way, the definition of a "consumer" entitled to the protections of the Act includes partnerships, corporations and other organization with assets that do not exceed $25 million.[5]

II. PRODUCT LIABILITY AND COMPUTERS

From the perspective of plaintiffs' lawyers framing a lawsuit, product liability law offers important advantages over breach of contract and consumer protection causes of action. If software developers are subject to product liability lawsuits, they may be held liable to parties who neither purchase nor use the developers' software, but who allege to have suffered damages as a result of defects in the software. In addition, such plaintiffs are not bound by the bilateral contractual limitations and disclaimers as agreed to between the developer and its customers.

Allowing such suits in the case of software defects that are found to contribute to network vulnerabilities would, however, be a radical departure from current law in two important respects. Allowing recovery for such claims would contravene the "economic loss doctrine," and the current

[5] *See, e.g.*, *Henry Schein, Inc. v. Stromboe*, 102 S.W.3d 675 (Tex. 2002), a case in which class action status was sought in an action involving allegedly defective dental practice management software, and alleging claims under the Texas statute.

limitation of strict liability in tort claims to physical products.

A. The Economic Loss Doctrine

The economic loss doctrine is a principle that derives from the fundamental nature of product liability law. The doctrine bars tort liability where only economic losses, and not personal injury or property damage, result from a defective product. The doctrine was evolved by courts concerned with imposing responsibility for physical injuries and property damage caused by "unreasonably dangerous" goods such as collapsing automobile tires[6] and exploding soft drink bottles.[7]

Product liability law can extend to computer hardware, although claims of property damage and personal injury from defective computer hardware appear to be fairly rare. A flurry of cases claiming "repetitive stress injury" from allegedly defective computer keyboards are examples of hardware tort liability cases.[8] Another example is a case involving an electrical fault that damaged the on/off switch of a computer's power supply, causing a fire that damaged both the computer and the user's facility. *Cadwell Indus. v. Chenbro Am., Inc.*, 119 F. Supp. 2d 1110 (E.D. Wash.

[6] *E.g., MacPherson v. Buick Motor Co.* 217 N.Y. 382 (1916).

[7] *E.g., Smith v. Peerless Glass Co.*, 259 N.Y. 292 (1932).

[8] *See, e.g., Irizarry v. DEC.*, 1996 U.S. Dist. LEXIS 11715 (N.D. Ill. Aug. 14, 1996).

2000). However, in most cases, when computer hardware or software malfunction, the resulting loss typically involves lost data, lost employee time or expenditures for remediation, rather than property damage or personal injury. Such losses fall within the economic loss doctrine and cannot be recovered in a product liability action.

So, for example, in *Affiliates for Evaluation & Therapy, Inc. v. Viasyn Corp.*, 500 So. 2d 688 (Ct. App. Fla., 3d Dist. 1987), the court dismissed a computer purchaser's product liability claim against a manufacturer for a defective computer that suffered mechanical breakdowns. Because the purchaser sought recovery for lost employee time and business losses resulting from the malfunctioning hardware, the court ruled that the economic loss doctrine barred the purchaser's product liability claim. See also *S.A.I., Inc. v. General Electric Railcar Services Corp.*, 935 F. Supp. 1150, 1154 (D. Kan. 1996) (discussing the applicability of the economic loss doctrine to computer software).

B. Application Only to "Products"

Another fundamental limitation of product liability law is that it applies to *products*. The Restatement (Second) of Torts § 402A, the wellspring of modern product liability law, provides the following examples of products: "a water heater, a gas stove, a power tool, a riveting machine, a chair, and an insecticide." As the court noted in *Winter v. G.P. Putnam's Sons*, 938 F.2d 1033, 1034 (9th Cir. 1991), "[t]he purposes served by products liability law ... are focused on the tangible world...."

Computer software, however, has generally not been viewed as subject to product liability suits under current law because software, at least as distinct from any hardware on which it might run, is not considered a "product." Judging by a dearth of reported decisions, no court thus far appears to have held that software independent of hardware may be subject to a product liability lawsuit. Commentators have suggested, or perhaps even assumed, that software that is part of a hardware product (such as software embedded in an automobile braking system) may be subject to a product liability lawsuit if defects in the software render the product defective in its operation and a personal injury results. An example discussed in the literature is a medical device[9] that malfunctions due to a defect in embedded software causing personal injury[10], and the potential for

[9] *See*, *e.g.,* Laura McNeill Hutcheson, *The Exclusion of Embedded Software and Merely Incidental Information from the Scope of Article 2B: Proposals for New Language Based on Policy and Interpretation*, 13 BERKELEY TECH. L.J. 977 (1998).

[10] Note that software contained in medical devices is highly regulated by the Food & Drug Administration. *See* FDA, General Principles of Software Validation; Final Guidance for Industry and FDA Staff (Jan. 11, 2002).

such liability is borne out in at least one reported jury verdict.[11]

C. Does Tort Liability Drive Product Improvement? – The Lessons of Y2K

Assuming that legislatures or courts expand the apparent limits of current product liability law and allow such claims against software developers, will the prospect of mega-million-dollar product liability judgments cause software developers to improve the security of their products as some have urged? The debate on that point is often a political one, framed by the battled-hardened views of consumer protection advocates versus free market thinkers, or class-action lawyers versus corporate lobbyists. But in the case of computer software, we can get beyond the political debate, and refer to a recent, real-world example in examining this question: the so-called Y2K software problem that emerged in the 1990s.

The Y2K problem arose because computer software written in the 1960s and subsequently, thought when it was written to have a limited useful life, could only handle two digits to indicate the year in a date field. Software engineers predicted that when the year 1999 turned to the year 2000, legacy computer systems ranging from desktop PCs to mainframes

[11] *Tamoozi v. Sofamor Danek Group*, No. 1999-46214 (Tex. Dist. Ct. Harris Cty, June 19, 2002) (multimillion dollar product liability verdict for injuries caused by design defect in software algorithm in image-guided surgical navigation system).

running enterprise systems would think that the clock had been turned back to 1900 and would misinterpret date information, and consequently fail when doing date-related calculations. Prophets of doom envisioned unpredictable and possibly catastrophic consequences to critical systems. Plaintiffs' lawyers envisioned a liability litigation flood of biblical proportions, causing law firms around the country to form specialized Y2K practice groups to deal with the expected litigation bonanza.

The Congressional response to the prospect of Y2K was an unusual piece of legislation that encouraged enterprises to address the Y2K problem in advance, and limited private suits for Y2K software defects. The Y2K Act, 15 U.S.C. §§ 6601-6617, embodied four major points. The Act (1) encouraged alternative dispute resolution; (2) provided incentives for companies to actively prevent Y2K-related failures; (3) required litigants to mitigate potential damages; and (4) required plaintiffs to comply with a pre-litigation notice period.

On the face of events, it appears that limiting liability for software defects may have been part of the solution to the Y2K problem. As we all know, the Millennium came and went without significant Y2K events. Soon thereafter, the Y2K practice groups were disbanded, and only a small number of Y2K related lawsuits ultimately were filed.[12] Perhaps the economic resources that would have been devoted to litigating Y2K issues went instead to mitigating Y2K problems.

III. WHO IS REALLY RESPONSIBLE FOR SECURITY VULNERABILITIES?

The fact is that there is no easy answer to the problem of computer network security vulnerability. Software developers strive to produce more secure products. They issue patches and updates for existing applications and develop new and more secure applications to address these vulnerabilities as they become apparent. But computer network security is and will continue to be a moving target, as highly-motivated hackers constantly probe and attack, inventing new exploits almost daily.

The notion that imposing new forms of liability on software developers will lead to a solution ignores the fact that often, insecure software is not the only cause of network security problems. Individual network operators and users, whether they are worldwide enterprises, Internet access providers or individual home users, have responsibility for securing their systems against attack or exploitation. Studies of network vulnerabilities have shown that a large percentage of systems, ranging from

[12] *See* Tuma, *It Ain't Over 'Til … A Post-Y2K Analysis of Litigation & Legislation*, 31 Tex. TECH L. REV. 1195 (2000).

the smallest to the largest, remain unprotected against known security vulnerabilities even when a patch or fix has been made available by a software developer.[13]

The root cause of network security vulnerabilities may be a fundamental design flaw that is attributable to no one in particular. Dr. F. Thomas Leighton, the Chief Scientist for Akamai Technologies, Inc., and a professor of Applied Mathematics at MIT, who testified before a Congressional subcommittee in October 2003, addressed this root cause. He expressed the view that the Internet, being a system of networked computers, is uniquely vulnerable to malicious attacks. As he pointed out, the Internet began as a small network of *trusted* systems. There is no central architect who designed the complex, interactive systems that subsequently developed, and there is no central command system that watches for and keeps out the bad actors.

And, perhaps most important are the commercial realities of the marketplace. Software development is a dynamic process in which product improvement is driven largely by customers seeking more functionality at a lower price. It is unclear whether customers would be actually willing to pay

[13] *See*, *e.g.*, Testimony of Gerhard Eschelbeck, Ph.D., Chief Technology Officer and V.P. of Engineering, Qualys, Inc., at hearings on “Worm and Virus Defense: How Can We Protect Our Nation's Computers From These Serious Threats?" before the Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, House Government Reform Committee (Sept. 10, 2003).

the price for more focus on security issues.

CONCLUSION

Information security vulnerability is, undeniably, one of the most significant problems we face as a society in this young millennium. While there is an emotional appeal to the argument that software developers should be held liable for security vulnerabilities, careful consideration should be given before current law is extended to impose special liability on developers. Re-designing liability law on the theory that one group of actors should accept a greater share of the risk for security failures — and perhaps even strict liability — is likely to divert important resources, including the time and attention of the computing community, from the complicated task of addressing security problems. Perhaps we should learn from the Y2K experience, and consider encouraging remediation, and not necessarily litigation.