

HEINONLINE

Citation:

Kevin R. Pinkney, Putting Blame Where Blame Is Due:
Software Manufacturer and Customer Liability for
Security-Related Software Failure, 13 Alb. L.J. Sci. &
Tech. 43 (2002)

Provided by:

UGA Law Library

Content downloaded/printed from [HeinOnline](#)

Mon May 27 17:00:05 2019

-- Your use of this HeinOnline PDF indicates your
acceptance of HeinOnline's Terms and Conditions
of the license agreement available at
<https://heinonline.org/HOL/License>

-- The search text of this PDF is generated from
uncorrected OCR text.

-- To obtain permission to use this article beyond the scope
of your HeinOnline license, please use:

[Copyright Information](#)



Use QR Code reader to send PDF
to your smartphone or tablet device

PUTTING BLAME WHERE BLAME IS DUE: SOFTWARE MANUFACTURER AND CUSTOMER LIABILITY FOR SECURITY-RELATED SOFTWARE FAILURE

*Kevin R. Pinkney**

TABLE OF CONTENTS

I. Introduction	45
II. Nature of the Threat of Security-Related Disasters	47
A. <i>The Scale of Security-Related Software Failure</i>	48
III. Common Methods of Causing Security-Related Software Failure	51
A. <i>Password Detection</i>	51
B. <i>The Specific Case: Buffer Overflows</i>	53
C. <i>Example of Buffer Overflow: rlogin</i>	54
D. <i>The General Case: Exploits Arising from Trusting Untrustworthy User Input</i>	55
E. <i>Automated and Semi-Automated Attacks</i>	57
F. <i>Attacks to Deny Service</i>	58
G. <i>Distributed Denial of Service Attacks</i>	59
H. <i>Attackers Have Different Skill Levels and Motives</i>	60
I. <i>The Life Cycle of Vulnerability</i>	60
IV. Ineffective Approaches to Preventing Security- Related Software Failure	62
A. <i>Government Approach Has Been Ineffective</i>	62
B. <i>Market Response Has Been Inefficient</i>	65
V. A Deterrence Model Built on Tort	69
A. <i>Software Manufacturer Liability: The Pre-Patch Case</i>	69
B. <i>Software Manufacturer and Customer Liability: The Post-Patched Case</i>	73

* B.S., 1997, Brigham Young University; J.D., 2002, Harvard Law School. The author is currently serving as a clerk for a judge on the Fifth Circuit Court of Appeals and is a member of the Virginia bar.

C. <i>Assigning Liability Only to One Party</i>	74
D. <i>Strict Liability with Contributory Negligence</i>	78
E. <i>Proportional Liability</i>	81
VI. Conclusion	82

Software launches missiles, flies jets and beams X-rays at people. More pervasively, if with less drama, it processes sales, replenishes inventories and performs countless other tasks on which businesses depend. When software fails, dependency can mean disaster.

—Douglas E. Phillips¹

I. INTRODUCTION

Disaster caused by software failure has increased commensurate to our dependency on computers. With tens of thousands of computers depending on identical software, there is potential for software failure to cause the same disasters on multiple machines. The problem is compounded when computers are networked, especially when an intruder intentionally causes software failure. Gone are the days when hacker intrusions were isolated and caused small-scale harm.² Today, security-related software failure can be replicated; hackers can mass-produce disaster.³

The potential disaster has received a high profile, if not a satisfying solution. Captains of industry focus on “trustworthy” software,⁴ while military commanders warn of an “electronic Pearl Harbor”⁵ and scientists detail the danger.⁶ The government and

¹ Douglas E. Phillips, *When Software Fails: Emerging Standards of Vendor Liability Under the Uniform Commercial Code*, 50 BUS. LAW. 151, 151 (1994).

² Such traditional intrusions still occur, when software is mis-configured, when ‘insiders’ exceed authorization, and when passwords are cracked. See discussion *infra* Part II.A. The effect of a security breach, while disruptive of a specific computer, had limited impact on the overall economy. See discussion *infra* Part II.A.

³ Viruses and worms can affect hundreds of thousands of host computers. See *Frequently Asked Questions about the Melissa Virus*, CERT/Coordination Center, at http://www.cert.org/tech_tips/Melissa_FAQ.html (last updated May 24, 1999) [hereinafter *Melissa FAQ*]. Melissa, a virus prominent in 1999, infected more than 100,000 computers in one weekend. *Id.*

⁴ See Jonathan L. Zittrain, *Taming the Consumer's Computer*, N.Y. TIMES, Mar. 11, 2002, at A21 (stating that Bill Gates declared this focus in an internal memorandum to all Microsoft employees in January 2002).

⁵ Lieutenant Colonel Joginder S. Dhillon & Lieutenant Colonel Robert I. Smith, *Defensive Information Operations and Domestic Law: Limitations on Government Investigative Techniques*, 50 A.F. L. REV. 135, 144 (2001) (noting that the Department of Defense suggests a small group of novice computer users could execute an electronic Pearl Harbor within three months of work).

⁶ COMPUTER SCI. AND TELECOMM. BD., NAT'L RESEARCH COUNCIL, CYBERSECURITY TODAY AND TOMORROW: PAY NOW OR PAY LATER 2–3 (2002), available at <http://www.nap.edu/openbook/0309083125/html/R1.html> (last visited Oct. 15, 2002) [hereinafter COMPUTER SCI. BD.].

market participants have taken some steps to reduce the risk created by hackers, but the response has been insufficient.⁷

The risk of security-related software failure can be attributed not only to hackers, but also to software manufacturers and software consumers.⁸ Nevertheless, the government's exclusive response has been an attempt to deter hackers.⁹ This approach is ineffective because many intrusions are possible purely because of sloppy software design.¹⁰ Many more are the result of lax system administration.¹¹ Rather than increasing accountability of manufacturers or consumers, the government has on at least two occasions shielded manufacturers from liability for harm arising from software failure.¹²

Market participant response to security-related software failure has also been inefficient.¹³ Software customers misjudge the risk of intrusion and under-consume security.¹⁴ Software manufacturers rush to market with products full of foreseeable vulnerabilities. Due to the market power possessed by some manufacturers,¹⁵ software manufacturers directly affect how much hacker risk enters the system.¹⁶ Software manufacturers are the least cost avoiders for many types of hack-prevention, yet they shoulder almost none of the harm that results from hacking.

A number of legal doctrines have constrained the market as well, at least historically. Several doctrines under the UCC coalesced to prevent effective contracting over liability for software

⁷ See discussion *infra* Part IV (stating that the government has actively tried to deter hacking, but has not been effective because of missed opportunities, imperfect information, and market power possessed by software manufacturers).

⁸ See discussion *infra* Part IV.A.

⁹ See discussion *infra* Part IV.A.

¹⁰ See discussion *infra* Part IV.A (discussing, among other items, the use of software defaults and the placement of "back doors" by software manufacturers).

¹¹ See *infra* Part III.A (describing how hackers use passwords to infiltrate systems and how users and administrators can work to prevent this).

¹² See *infra* Part IV.A.

¹³ See *infra* Part IV.B.

¹⁴ See *infra* Part IV.B (noting that under-consumption occurs because good security is a positive externality).

¹⁵ See, e.g., *United States v. Microsoft Corp.*, 253 F.3d 34, 45-46 (D.C. Cir. 2001) (finding that Microsoft is a monopoly in the market for desktop operating systems).

¹⁶ See, e.g., Don Clark, *Digital says Microsoft Program isn't Secure*, WALL ST. J., Feb. 14, 2002, at B6 (discussing how software manufacturer's attempt to decrease security risk resulted in an increase in security risk), available at 2002 WL-WSJ 3385946; see also *infra* Part IV. B.

failure. Furthermore, the common law disallowed recovery for the economic losses typical of software failure suits.¹⁷

Due to the magnitude of disaster caused by security-related software failure, courts will increasingly handle suits against software manufacturers for harms resulting from failure. Understanding the common design flaws that lead to security-related software failure is essential when considering such things as what configuration of liability rules a court might adopt and whether a software manufacturer exercised reasonable or due care. Such an understanding suggests: 1) that software manufacturers should be liable, perhaps strictly liable, for design flaws underlying security-related software failure and; 2) that software manufacturers should have a defense against liability when software customers delay installation of security patches.

The discussion that follows is divided into several sections. Part II details the possible scale of a network disaster, the popular techniques for causing security-related software failure, the composition of the hacker community, and the life cycle of an exploitation used to cause software failure.¹⁸ Part III explains how both the government and the market have failed to fully address security-related software failure.¹⁹ Part IV analyzes several configurations of liability rules that could lead to efficiently preventing security-related software failure.²⁰ Finally, Part V concludes that security and market efficiency will increase as software manufacturers are held liable for damage caused when the software they offer fails to prevent intrusions.²¹

II. NATURE OF THE THREAT OF SECURITY-RELATED DISASTERS

Proposing software manufacturer and consumer liability for security-related software failure under tort law requires that "security-related software failure" be defined. To begin with, "[s]oftware reliability is defined as 'the probability of failure-free operation of a computer program for a specified time' in a given environment."²² On the other side of the coin, "software 'failure,'

¹⁷ See *infra* Part V (discussing the applicability of tort law to software failure).

¹⁸ See *infra* notes 22–46 and accompanying text.

¹⁹ See *infra* notes 47–132 and accompanying text.

²⁰ See *infra* notes 133–86 and accompanying text.

²¹ See *infra* notes 187–284 and accompanying text.

²² Phillips, *supra* note 1, at 155.

as used in defining reliability means that a computer program "in its functioning has not met user requirements in some way."²³

"Security-related" refers to the subset of software failures that allows an intruder to access "a computer without authorization or exceeding authorized access."²⁴ "Security-related software failure" refers to all situations where the functioning of a computer program has not met all user requirements relating to access of either information or control.²⁵

In order to understand how tort liability might deter security-related software failure, it is important to consider the possible scale of a network disaster, the prevailing techniques for causing security-related software failure, the composition of the hacker community, and the life cycle of an exploitation used to cause software failure.

A. *The Scale of Security-Related Software Failure*

Computer systems and networks are vulnerable to traditional harms, such as equipment failure, human error, natural disaster, and physical attack.²⁶ Often these threats are from outside an organization—concerns over the physical safety of computer networks during a nuclear war justified research that eventually resulted in the Internet.²⁷ Threats also originate within organizations, from both past and present disgruntled employees.²⁸

Computers and networks are additionally vulnerable to exotic threats not applicable to other business equipment. When a corporate or home network attaches to the Internet it becomes close neighbors with over a billion people.²⁹ The nosier neighbors are

²³ *Id.*

²⁴ 18 U.S.C. § 1030(a) (2000).

²⁵ Lawrence R. Rogers, *rlogin(1): The Untold Story*, CERT/COORDINATION CENTER, Nov. 1998, at 3–4 (demonstrating how control can be usurped through the rlogin program, where one measure of control is whether an intruder can cause the "execution of arbitrary code," or code the content of which is the full discretion of the intruder), at <http://www.cert.org/archive/pdf/98tr017.pdf> (last visited Oct. 16, 2002).

²⁶ See COMPUTER SCI. BD., *supra* note 6, at 3–5 (noting for example, lighting surges destroying power supplies or accidental human programming errors).

²⁷ Dhillon & Smith, *supra* note 5, at 138.

²⁸ Eric J. Sinrod & William P. Reilly, *Cyber-Crimes: A Practical Approach to the Application of Federal Computer Crime Laws*, 16 SANTA CLARA COMPUTER & HIGH TECH. L.J. 177, 184 (2000). In one survey "fifty-five percent of survey respondents reported malicious activity by insiders." *Id.* at 185.

²⁹ See Dhillon & Smith, *supra* note 5, at 138. Estimates regarding Internet usage tend to differ. One estimate reported in March of 2000 puts the number of users at 304 million people. Mary M. Calkins, Note, *They Shoot Trojan Horses*,

saboteurs, foreign spies, criminal organizations, law enforcement agencies, competitors, juvenile deviants, activists, and the curious.³⁰ Such neighbors create the risk of accidental and intentional harm.³¹

There are many anecdotes of the harm. When telecom giant MCI was infiltrated by hackers, fifty thousand credit card numbers were stolen and used to transact "\$50 million in charges and purchases."³² Citibank lost millions of dollars and some prestige as the result of a hacker attack.³³ Revlon allegedly lost \$20 million when a disabled computer system prevented products from shipping.³⁴ The telephone local loop in Worcester, Massachusetts was disabled, killing 911 services and potentially preventing the local air traffic control tower from landing aircraft.³⁵ Furthermore, intruders at the Air Force's Laboratory in Rome, New York went undetected for five days.³⁶ During this time the intruders accessed critical information and used the Lab's computer system as a launch pad for infiltrating the computers of NASA's Goddard Space Flight Center.³⁷ The space shuttle Atlantis was tempora-

Don't They? An Economic Analysis of Anti-Hacking Regulatory Models, 89 GEO. L.J. 171, 172 (2000).

³⁰ See Dhillon & Smith, *supra* note 5, at 157; see also Calkins, *supra* note 29, at 176 (discussing the variety of hacking groups, in contrast to the hacker stereotype).

³¹ See Calkins, *supra* note 29, at 186-87 (detailing the various costs associated with hackers acting with benign intent).

³² David L. Gripman, Comment, *The Doors are Locked but the Thieves and Vandals are Still Getting in: A Proposal in Tort to Alleviate Corporate America's Cyber-Crime Problem*, 16 J. MARSHALL J. COMPUTER & INFO. L. 167, 169-70 (1997).

³³ See *id.* at 169 n.13 (noting that the FBI arrested the Russian hackers responsible for stealing approximately \$12 million, and were able to recover most of the lost funds).

³⁴ Timothy P. Heaton, *Electronic Self-Help Software Repossession: A Proposal to Protect Small Software Development Companies*, 6 B.U. J. SCI. & TECH. L. 8, 8 (2000) (discussing Revlon, Inc. v. Logisticon, Inc. and the circumstances involved with Logisticon intentionally disabling Revlon's software after Revlon's failure to fully pay its bill).

³⁵ Dhillon & Smith, *supra* note 5, at 141-42. The teenage intruder tested various commands, apparently unaware the compromised system was involved with telephone service. Calkins, *supra* note 29, at 185 n.79.

³⁶ Dhillon & Smith, *supra* note 5, at 142-43.

³⁷ *Id.* Ironically, "[t]he Rome Laboratory is the Air Force's lead research facility for all information technology issues." *Id.* at 142. The Air Force spent an estimated \$500,000 responding to the intrusion, with the damage to national security unknown. *Id.* at 143.

rily prevented from docking with the International Space Station when NASA was attacked in 1997.³⁸

The scale of vulnerability has grown far beyond isolated intrusions. Recent attacks against high profile websites have involved unauthorized use of thousands of computers.³⁹ There have been attempts to estimate the overall threat of harm from software failure allowing security breaches.⁴⁰ Various surveys suggest that between half and three-quarters of networks have experienced security breaches,⁴¹ with the rate of infiltration increasing over time.⁴² The FBI has estimated that computer viruses that are related to electronic crimes cost approximately \$10 billion per year.⁴³

Authorized intrusion attempts, used to test security, also indicate widespread vulnerability. In 1999, the agency responsible for security evaluations of the United States Department of Defense attempted to gain unauthorized access on 38,000 occasions and succeeded two-thirds of the time.⁴⁴ A separate military exercise, designed to test the government's response to a coordinated cyber assault, showed severe security inadequacies.⁴⁵ The special

³⁸ See Gregory L. Vistica & Evan Thomas, *The Secret Hacker Wars*, NEWSWEEK, June 1, 1998, at 60 (noting that a hacker used the Internet to overload an uplink in NASA's reportedly vulnerable system), available at 1998 WL 9578708.

³⁹ Matt Richtel & Sara Robinson, *Several Web Sites Attacked Following Assault on Yahoo*, N.Y. TIMES, Feb. 9, 2000, at A1 (discussing the attack on yahoo.com in the spring of 2000).

⁴⁰ See *infra* Part III.B (discussing the problems associated with making these estimations).

⁴¹ Calkins, *supra* note 29, at 172-73 (citing various surveys measuring the level of hacker activity).

⁴² Michael Lee et al., Comment, *Electronic Commerce, Hackers, and the Search for Legitimacy: A Regulatory Proposal*, 14 BERKELEY TECH. L.J. 839, 845 (1999) (noting that the federal government believes the number of hacker attacks increases by seventy percent each year).

⁴³ Dhillon & Smith, *supra* note 5, at 139. Others estimate that "corporations spent about \$12.3 billion to clean up damage from computer viruses in 2001." *U.S. Cyber Security Weakening*, WIRED NEWS, Jan. 8, 2002, at <http://www.landfield.com/isn/mail-archive/2002/Jan/0051.html> (last visited Nov. 1, 2002).

⁴⁴ Dhillon & Smith, *supra* note 5, at 140.

⁴⁵ See *id.* at 143-44. Exercise "Eligible Receiver" had an "enemy" team comprising forty Department of Defense employees with "a working knowledge of information technology," and no computer experts. *Id.* at 144. The team was allowed three months to plan an attack, with their weapons "limited [to] hacker tools available on the Internet." *Id.* The team succeeded in shutting down air traffic control systems, oil refineries, and power grids. *Id.* They also succeeded in disrupting the DOD's logistic system, having the ability to change orders for missiles to orders for headlamps, and the capability of disrupting train traffic. *Id.*

master in a lawsuit over administration of funds, held in trust for Indian tribes, hacked into the computers at the Bureau of Indian Affairs and set up multiple false trust accounts.⁴⁶

III. COMMON METHODS OF CAUSING SECURITY-RELATED SOFTWARE FAILURE

Hackers use a handful of common strategies to gain unauthorized access to computers and networks.⁴⁷ Almost all the strategies arise in the first instance because programmers ignore known security pitfalls and system administrators fail to quickly patch software having known vulnerabilities.⁴⁸

A. Password Detection

Since most computers and networks use password systems to authenticate users, discovering the name and password of an authorized user is the most immediate way to gain unauthorized access to a computer.⁴⁹ This approach is particularly easy for insiders and ex-employees,⁵⁰ two groups that are considered significant sources of unauthorized access.⁵¹ Non-insider intruders using this approach require two things: a connection to the target and time.⁵² Establishing a connection is often trivial because many computers are attached to the Internet, and stand-alone

⁴⁶ John Markoff, *'Hackers' Find No Bars to Indian Trust Files*, N.Y. TIMES, Feb. 26, 2002, at A1. The special master reported the successful hack to the judge and the Bureau before any checks were sent to the addressee of the false accounts. *Id.*

⁴⁷ See Lee, *supra* note 42, at 847.

⁴⁸ See Kevin J. Houle & George M. Weaver, *Trends in Denial of Service Attack Technology*, CERT/COORDINATION CENTER, Oct. 2001, at 19 (stating that network resources are limited and susceptible to attacks and contain vulnerabilities that fail to be patched in a timely manner), at http://www.cert.org/archive/pdf/DoS_trends.pdf (last visited Nov. 1, 2002).

⁴⁹ See Gripman, *supra* note 32, at 184–85 n.106 (explaining how people choose passwords that are easy to remember, thus making them vulnerable to being cracked).

⁵⁰ Lee, *supra* note 42, at 848.

⁵¹ See Gripman, *supra* note 32, at 189; see also Keith J. Epstein & Bill Tancer, *Enforcement of Use Limitations by Internet Service Providers: "How to Stop that Hacker, Cracker, Spammer, Spoofer, Flamer, Bomber"*, 19 HASTINGS COMM. & ENT. L.J. 661, 681 (1997) (asserting that "[m]ost hacking is the product of system administrators and users who carelessly choose passwords, who leave passwords lying around where they can be found, and/or who forget or are too lazy to change passwords").

⁵² Cf. Gripman, *supra* note 32, at 185 (inferring that hackers succeed because a password's security value decreases with time and hackers' programs try every word in the dictionary with proper variants).

networks are vulnerable if there is even one unauthorized modem installed.⁵³

Once a connection is established, a determined intruder might attempt to crack the password by trying every word in the dictionary.⁵⁴ Such "brute-force" attacks regularly succeed,⁵⁵ but they can be slowed by password systems requiring non-alphanumeric characters.⁵⁶

Would-be intruders also use "social engineering" to solicit passwords from users. For example, users have been tricked into supplying passwords to strangers pretending to be legitimate system engineers.⁵⁷ A final approach to compromising passwords is simply to try default passwords. Hardware often ships, and software often installs, with default users and passwords that are not required to be changed immediately.⁵⁸

An intruder masquerading as a legitimate user remains subject to any access restrictions faced by the user. But intruders with initially limited access can investigate the target system and often glean enough information to elevate themselves to system administrator status.⁵⁹

Software customers are often in full control of specifying rules for password systems, such as minimum password length,

⁵³ *Id.* at 169–70, 178. "War dialing" refers to the systematic dialing of phone numbers believed to be in the range of the intended target. *Id.* at 168 n.6. Hackers generally operate at night, when staff is absent, to avoid detection and suspicion. *See, e.g., id.* at 168 n.6.

⁵⁴ *Id.* at 185.

⁵⁵ Lee, *supra* note 42, at 851.

⁵⁶ *See* Gripman, *supra* note 32, at 185 n.108 (providing examples of difficult to crack, yet easy to remember, passwords; such as the use of randomly selected dictionary pass phrases, with arbitrary numbers of symbols).

⁵⁷ *Cyberterrorism Hype*, INFORMATION WARFARE SITE, Oct. 21, 1999 (describing how a social engineering attack may proceed), at <http://www.iwar.org.uk/cyberterror/resources/janes/jir0525.htm> (last visited Oct. 19, 2002). Initially, the attacker calls an employee and says,

"Hi, I'm Cheryl. I'm new in IT support. I'm having trouble with the modem bank. Can you check the modem to make sure it's turned on? Also, can I have the number to make sure I'm using the right one?" Of course, being a diligent and helpful worker, the recipient of such a call is only too happy to help.

Id.

⁵⁸ *See, e.g.,* Jason Rafail, *Vulnerability Note VU#557136: Cayman Gateways Ship with Null Administrative and User Level Passwords*, CERT/COORDINATION CENTER, at <http://www.kb.cert.org/vuls/id/557136> (last updated Aug. 27, 2001); Cory Cohen & Jeffrey P. Lanza, *Vulnerability Note VU#212088: Alcatel ADSL Modems Contain a Null Default Password*, CERT/COORDINATION CENTER, at <http://www.kb.cert.org/vuls/id/212088> (last updated Apr. 11, 2001).

⁵⁹ *See* Rafail, *supra* note 58.

mandatory inclusion of non-alphanumeric characters, and frequent password changes.⁶⁰ Customers can also respond to concerns that passwords are antiquated mainframe security in an Internet world by adding additional authentication systems.⁶¹ Nevertheless, improved technology will not immunize against spurious authentication, in part because whatever the software customers depend on may still allow intruders to impersonate legitimate users.⁶²

B. The Specific Case: Buffer Overflows

Buffer overflows are one of several recurring software design flaws and are responsible for a significant portion of the reports received by Internet security groups such as the Computer Emergency Response Team (CERT) at Carnegie Mellon University.⁶³ Exploits of buffer overflow vulnerabilities can allow intruders unrestricted access to computer and network resources.⁶⁴

Many computer programs copy user input directly to the "buffer," a temporary storage area, without testing it for content or length. When the user input is longer than the buffer, it can overwrite the very instructions the computer is currently executing since these instructions are often adjacent to the buffer in the computer's memory.⁶⁵ Sophisticated hackers determine the

⁶⁰ See Gripman, *supra* note 32, at 185 (suggesting that corporations should require users to create new passwords frequently and to use passwords not easy to guess but still easy to remember).

⁶¹ Dhillon & Smith, *supra* note 5, at 139; see also Gripman, *supra* note 32, at 187–89 (setting forth possible solutions to this problem through the use of encryption technology); Lee, *supra* note 42, at 859–61 (providing an examination of the development of contemporary password systems).

⁶² COMPUTER SCI. BD., *supra* note 6, at 12; see also Gripman, *supra* note 32, at 188–89 (discussing the pitfalls of encryption technology).

⁶³ See Rogers, *supra* note 25, at 1, 3 (explaining how coding defects allow subroutine linkage information to be overwritten, which allows control to be transferred to a random address in a computer's memory system); see also *infra* notes 84–87 and accompanying text (discussing other recurring design flaws, such as unexpected operators and cross-site scripting).

⁶⁴ See, e.g., Larry Rogers, *Buffer Overflows—What are They and What Can I Do About Them?*, CERT/COORDINATION CENTER, at http://www.cert.org/homeusers/buffer_overflow.html (last updated Dec. 3, 2001) [hereinafter *Buffer Overflow*]. A buffer overflow has been analogized with "trying to put ten pounds of sugar into a container that only holds five pounds." *Id.* When the creator of a computer program does not check the size of the destination area or the buffer, and inputs data larger than the buffer, a spillover results. *Id.* However, the sugar is distinguishable from the buffer overflow because unlike sugar, which can be cleaned up and restores the area to its original condition, the excess data in the buffer overwrites the previous content in the computer's memory space. *Id.*

⁶⁵ See *id.*

length user input needs to be in order to overflow the buffer and to supply input exceeding that length. The input contains a short computer program to do something useful (from the standpoint of the hacker). The overflowing input typically replaces the instruction the computer will next execute with an instruction that transfers control to the short program introduced into the buffer.⁶⁶ The content of the short program is completely within the discretion of the hacker.⁶⁷ When the hacker supplies the input to a computer running the vulnerable software, the computer executes the arbitrary code, which often opens communication to the hacker and allows permissions equal to the most trusted system administrator.⁶⁸

C. Example of Buffer Overflow: *rlogin*

"Many UNIX systems provide the *rlogin* program."⁶⁹ The *rlogin* program allows a user to access the UNIX system from a remote terminal,⁷⁰ such as a computer attached to the Internet. It provides information about the remote terminal, including a terminal type description.⁷¹ Furthermore, "[t]his functionality allows terminal-aware programs such as full-screen text editors to operate properly across the computer-to-computer connection created with *rlogin*."⁷²

But many versions of the *rlogin* program contain a design flaw.⁷³ When *rlogin* receives the terminal type description, the program copies the description into the buffer without first checking to see if the buffer is large enough to accommodate the terminal type description.⁷⁴ If the description is larger than the buffer, the buffer "overflows."⁷⁵ On some computers, the buffer is adjacent to critical information, including a link indicating where con-

⁶⁶ See *id.*; see also Rogers, *supra* note 25, at 3.

⁶⁷ See *Buffer Overflow*, *supra* note 64.

⁶⁸ See Rogers, *supra* note 25, at 4–5 (providing a full explanation of the *rlogin* vulnerability, including an example of a defective code segment).

⁶⁹ *Id.* at 3.

⁷⁰ *Id.* (quoting Brian Kantor, *BSD Rlogin*, *Request for Comment 1282*, available at <http://rfc.sunsite.dk/rfc/rfc1282.html>) (last visited Nov. 2, 2002).

⁷¹ Rogers, *supra* note 25, at 3 (explaining that the terminal type description is stored in the TERM environment variable).

⁷² *Id.*

⁷³ *Id.*

⁷⁴ See Rogers, *supra* note 25, at 3.

⁷⁵ See *Buffer Overflow*, *supra* note 64 (explaining that a data overflow requires the computer owner or system administrator's attention to clean up the mess).

trol should be transferred once users exit *rlogin*.⁷⁶ This means that part of the terminal type description can overwrite the link information and pass control to an arbitrary area of the computer system's memory.⁷⁷

By engineering the value of the terminal type description, a careful intruder can simultaneously fill the buffer with instructions, overflow the buffer, and re-write the link to point to the instructions in the buffer.⁷⁸ When the user exits *rlogin*, the computer follows the (now re-written) link, and executes the instructions inserted by the intruder. The nature of the instructions is completely under the control of the intruder.⁷⁹

In addition, the computer runs the new instructions with "root" privileges, typically the level of the most trusted system administrator.⁸⁰ The *rlogin* program on the target machine runs as root because it must have access to open a "port" connection to the remote terminal.⁸¹ The *rlogin* program never sheds its root privileges, so the surreptitiously inserted instructions are executed in root mode.⁸²

D. The General Case: Exploits Arising from Trusting Untrustworthy User Input

Hackers exploit software when it fails to verify user input, or more generally, when it accepts user input as trustworthy without testing to see whether the input is in fact.⁸³ This is a design flaw underlying buffer overflows. It is also the flaw that provides

⁷⁶ See Rogers, *supra* note 25, at 3 (specifying that the buffer is adjacent to a variable local to the main subroutine, and the subroutine linkage information can be overwritten with data from the TERM environment variable).

⁷⁷ *Id.*

⁷⁸ See *id.* at 4 (indicating this procedure is referred to as stack smashing).

⁷⁹ *Id.*

⁸⁰ *Id.* at 16, 23 (describing how *rlogin* is a poor authentication scheme).

⁸¹ *Id.*

⁸² Rogers, *supra* note 25, at 16–17 (stating that "[t]he goal we are trying to achieve here is to use extraordinary privileges only where needed and then give them up when they are no longer required").

⁸³ See *id.* at 14–15 (suggesting that in order to make untrustworthy data trustworthy the following steps should be taken: "1. Identify the boundaries in a program. 2. Identify data that cross boundaries in that program [, and] 3. Examine that data for correct form, substituting meaningful and predictable defaults for nonconforming data").

opportunities for “cross-site” scripting attacks,⁸⁴ unexpected oper-

⁸⁴ Shawn Hernan, *Vulnerability Note VU#672683: Apache Tomcat Vulnerable to Cross-Site Scripting via Passing of User Input Directly to Default Error Page*, CERT/ COORDINATION CENTER (explaining that cross-site scripting attacks are a web-browser example of vulnerabilities that arise because software products do not verify user-supplied data), at <http://www.kb.cert.org/vuls/id/672683> (last updated Aug. 17, 2001). For example, cross-site scripting attacks inject malicious scripts (typically written in JavaScript or VBScript) into a web page. *Id.* To launch the attack, an intruder causes a legitimate web server to send a page that contains a malicious script of HTML code of the intruder’s choosing to a victim’s web browser. *Id.*

The malicious script runs with the privileges of a legitimate script originating from the legitimate web server. Several server applications are vulnerable to such an [sic] technique via various default error pages. For example, A [sic] valid URL request might be

<http://www.example.com/FILENAME.html>

However, if the requested document “FILENAME.html” did not exist, the web site could return an error message such as:

<HTML>

404 page does not exist: FILENAME.html

....

</HTML>

Notice that “FILENAME.html” is a string that was inputted by a user and is included in the page returned by the web site straight through to the client’s browser.

If a malicious web site existed that offered a link to example.com that looked something like this <A HREF=“<http://www.example.com/><script%20SRC=‘<http://www.malicioussite.com/evilscrip.js>’></script>“> Click Here

The “FILENAME.html” submitted to example.com is

<script SRC=‘<http://www.malicioussite.com/evilscrip.js>’></script>

example.com then uses its ordinary routines to generate an error page to you that reads: <HTML>404 page not found: <script SRC=‘<http://www.malicious site.com/evilscrip.js>’></script> . . . </HTML>

In effect, malicioussite.com has managed to “inject” a JavaScript program of their choosing into the page returned to the user by example.com. The JavaScript runs as if it originated at example.com, and can therefore process events in that document, but it can also communicate with malicioussite.com by virtue of having come from there. Thus, this vulnerability could be used to “sniff” sensitive data from within the web page, including passwords, credit card numbers, and any arbitrary information the user inputs. There are a variety of variants to this problem.

The ultimate fix to this problem involves recoding a very large number of web sites so that they properly filter and validate the input they receive and properly encode or filter the output before returning it to the user or acting upon it. This process is a very large undertaking.

Id.

ator⁸⁵ attacks,⁸⁶ and manipulations of MIME types.⁸⁷

E. Automated and Semi-Automated Attacks

Once a hacker devises a method to trigger software failure in one application running in a given environment, the method often works on the software running in the same environment on many different computers. One way hackers leverage common hardware, software, or operating combinations, is to automate the hacking process by writing a "virus" or "worm" to exploit the vulnerability.

Many computer worms are completely automated attacks; they contain code both to detect and exploit vulnerabilities.⁸⁸ Such worms can replicate themselves onto newly exploited computers that become launch pads for more exploitation.⁸⁹ Computer viruses are considered semi-automatic because some user interaction is required to trigger the hacker's arbitrary code.⁹⁰

Both worms and viruses rely on the security-related software failures previously discussed. For example, the Melissa virus manipulated the MIME type of the e-mail attachment⁹¹ and the

⁸⁵ Arithmetic equations, or "expressions," contain numbers and operators. In the familiar expression 1+1, "+" is the operator. Other arithmetic operators include -, *, ÷. Computer languages usually include these familiar arithmetic operators. But they also include operators suited for computer environments, such as for directing flows of input and output.

Many computer programs running on UNIX systems accept expressions as input. For a user typing his or her age, "49+1" or "5*10" will be interpreted by the computer as though the user had typed "50". Using the computer-specific operators can produce unexpected results and transfer program control to a malicious user.

⁸⁶ *Vulnerability Notes by Metric*, CERT/COORDINATION CENTER, at <http://www.kb.cert.org/vuls/bymetric> (last visited Nov 2, 2002).

⁸⁷ MIME is the protocol that allows identification of e-mail attachments. A *Quick Guide to Email Security*, ZZEE: SIMPLE WEB SOLUTIONS, at http://www.zzee.com/email_security (last visited Nov. 2, 2002). The Melissa virus exploited the way Outlook processed MIME types: the attachment claimed to be an image file (in jpeg format) in order for Outlook to attempt to open it. See *Melissa FAQ*, *supra* note 3. But Outlook, during the attempt to open it, would pass it to the command interpreter because the attachment was an executable program. See *id.*

⁸⁸ See Houle & Weaver, *supra* note 48, at 10–12 (tracing the development of automated attacks and describing various worms).

⁸⁹ *Id.*

⁹⁰ See *id.* (stating that a familiar trigger is the opening of an email attachment).

⁹¹ See *supra* note 87 and accompanying text.

Code Red worm relied on, among other things, buffer overflow vulnerability.⁹²

F. Attacks to Deny Service

Computers and networks are also vulnerable to attackers seeking to disrupt rather than to intrude. Disrupters seek to prevent or impair legitimate uses of resources and commonly target bandwidth, processing power, and storage capacity.⁹³ Denial-of-service (DoS) attacks have been analogized to “sending a hundred people into a store who overwhelm the sales staff, rendering them unable to respond to legitimate customers. Eventually, the store becomes so crowded that a line forms outside, where the ‘bogus’ customers and real customers queue up, denying access to legitimate customers.”⁹⁴

Denial-of-service attacks exploit features of the protocols that govern how computers communicate over the Internet. When computers attempt to communicate with each other, their communication is governed by a communication protocol. Under one of these, the Transmission Control Protocol/Internet Protocol, two computers begin to communicate when the “client” seeks to establish a connection to the “server” by sending a “SYN flag.”⁹⁵ The server responds, opens a channel of communication to the client, and waits for further communication.⁹⁶

In a DoS attack, a hacker, using one computer with a fixed address, floods one server with packets containing the SYN flag without any intention of using the resulting channel to communicate further.⁹⁷ Since there is a limit to the number of SYN requests to which a server can respond, inundating a server with insincere SYN packets prevents the server from responding to requests made from legitimate users.⁹⁸

The obvious response to DoS attacks was to screen incoming packets to ensure no single source emitted too many requests.⁹⁹

⁹² See discussion *supra* Part III.B.

⁹³ Houle & Weaver, *supra* note 48, at 1.

⁹⁴ Sinrod & Reilly, *supra* note 28, at 190 n.60.

⁹⁵ *Id.* at 190–91. The term “SYN” stands for “synchronized/start.” *Id.* at 190 n.61.

⁹⁶ *Id.* at 190.

⁹⁷ *Denial of Service Attacks*, CERT/COORDINATION CENTER, at para. 3 (noting the DoS attacks are generally focused against network connections with the goal of preventing host or network communications), at http://www.cert.org/tech_tips/denial_of_service.html (last visited Nov. 2, 2002).

⁹⁸ *Id.*

⁹⁹ See *id.* at para. 4.

Attackers modified their attack styles to mask the source of the SYN requests.¹⁰⁰ This elicited a new counter-measure, as there continues to be an arms race with single source DoS attacks.¹⁰¹

G. *Distributed Denial of Service Attacks*

Distributed Denial of Service (DDoS) attacks occur when the tools used for a single source DoS are distributed to multiple computers and these computers, in concert, flood the target with requests.¹⁰² The earliest DDoS attacks were launched after hackers manually broke into well-placed intermediate computers (using vulnerabilities such as buffer overflows), planted DoS attack tools, and subsequently ordered the intermediate computers to attack.¹⁰³ Over time, hackers automated the process of compromising intermediate machines by injecting DoS attack scripts into worms.¹⁰⁴ These worms exploit well-known vulnerabilities, replicate themselves, and are programmed to launch a coordinated attack without interaction from the worm's creator.¹⁰⁵

Websites with enormous resources are not immune to DDoS attacks.¹⁰⁶ In February of 2000, the mainstream media reported on the "DDoS attacks that targeted several high-profile websites."¹⁰⁷ DDoS attacks were historically launched against UNIX systems, both professionally and end-user administered.¹⁰⁸ However, the end-users of Windows-based systems and Internet routers are being targeted more and more.¹⁰⁹ Vulnerabilities are so plentiful that rival hackers vie for the most attractive "territory" of computers and routers.¹¹⁰

¹⁰⁰ See Houle & Weaver, *supra* note 48, at 3 (discussing the ease with which a SYN packet's origin can be obscured); Sinrod & Reilly, *supra* note 28, at (noting the challenge to law enforcement when SYN packet origin is falsified).

¹⁰¹ See *Denial of Service Attacks*, *supra* note 97, at para. 4.

¹⁰² Sinrod & Reilly, *supra* note 28, at 194.

¹⁰³ Houle & Weaver, *supra* note 48, at 12 (noting that systems were tested for network connectivity, traffic, and available bandwidth before being used as agents in these DDoS attacks).

¹⁰⁴ See *id.* at 10–12.

¹⁰⁵ See *id.* at 10–11 (describing how the Code Red and Morris Worms scanned for vulnerable systems, exploited this vulnerability, and copied itself and the DoS attack tools to the new system).

¹⁰⁶ See *id.* at 1–2.

¹⁰⁷ *Id.* at 5; see, e.g., Matt Richtel, *Yahoo Blames a Hacker Attack for a Lengthy Service Failure*, N.Y. TIMES, Feb. 8, 2000, at C11; Richtel & Robinson, *supra* note 37, at A1.

¹⁰⁸ See Houle & Weaver, *supra* note 48, at 12.

¹⁰⁹ See *id.*

¹¹⁰ See *id.* at 14.

H. Attackers Have Different Skill Levels and Motives

As the above examples suggest, some attacks require sophistication and intelligence, and some attacks require neither. There are obvious obstacles to quantifying the relative proportion of skilled to unskilled hackers. Nevertheless, some observers believe there is a vanguard of "elite" hackers representing approximately one percent of total hackers.¹¹¹ Hackers also have varied motives. Initially, many in the hacker community "viewed themselves as 'learners and explorers who want[ed] to help rather than cause damage.'" ¹¹² Hackers also had a code of ethics.¹¹³

As the Internet expanded, the hacker community used newsgroups and websites to share information.¹¹⁴ The public nature of the Internet allowed the hacking community to grow and absorb members with less benign motives.¹¹⁵ It is not clear what proportion of present-day hackers maintains the old ethical code.¹¹⁶

I. The Life Cycle of Vulnerability

The life cycle of vulnerability reflects the varying skill levels within the hacker community. CERT divides the life cycle into six phases roughly corresponding to the decreasing skill level needed to exploit the vulnerability.¹¹⁷ Phase one consists entirely of activity by elite hackers and comprises the time span when vul-

¹¹¹ See *Cyberterrorism Hype*, *supra* note 57 (noting that, Bruce Sterling, author of *THE HACKER CRACKDOWN: LAW AND DISORDER ON THE ELECTRONIC FRONTIER*, estimates that anywhere from 250 to 1,000 of a total 100,000 hackers can be skilled enough to be designated elite); Sinrod & Reilly, *supra* note 28, at 182-83.

¹¹² Lee, *supra* note 42, at 865.

¹¹³ *Id.* at 865-66. "The . . . 'hacker ethic' included the following principles: 1) ['Access to computers-and anything which might teach you something about how the way the world works—should be unlimited and total.' 2) 'All information should be free.' 3) 'Thou Salt Not Destroy.'" *Id.*

¹¹⁴ See *id.* at 867 (noting that hackers, at one point, congregated around private Bulletin Board Systems).

¹¹⁵ See Calkins, *supra* note 29, at 176. These new hackers "include disgruntled employees, foreign spies, fraud perpetrators, political activists, conventional criminals, and juveniles with little computer knowledge." *Id.*

¹¹⁶ See Lee, *supra* note 42, at 867 (noting that some commentators doubt if the hacker ethic was ever very widespread, even though the existence of this code is undisputed).

¹¹⁷ CERT/COORDINATION CENTER OVERVIEW, INCIDENT AND VULNERABILITY TRENDS, at illus. 25 (2002), at <http://www.cert.org/present/cert-overview-trends/module-2.pdf> (last visited Nov. 2, 2002) [hereinafter INCIDENT AND VULNERABILITY TRENDS].

nerabilities are discovered.¹¹⁸ This phase begins even before software is released to the public.¹¹⁹

During phase two, elite hackers develop crude tools to exploit the vulnerability and distribute those tools to other sophisticated hackers.¹²⁰ Internet security groups, such as CERT, may detect the new exploitation and issue public advisories as early as phase two.¹²¹

Phase three is ushered in when less-sophisticated hackers begin using the crude exploit tools.¹²² The fourth phase is reached with the introduction of software tools that automate detection and exploitation of the vulnerability.¹²³

The fifth phase displays the widespread use by novice hackers of the tools developed in the fourth phase.¹²⁴ Software manufacturers usually produce patches for the vulnerability by the beginning of the fifth phase.¹²⁵ Generally, the fifth phase is just underway when Internet security groups issue the first advisories regarding the vulnerability.¹²⁶ This phase can last much longer than the first four phases combined.¹²⁷ It lingers because software customers are slow to install software patches and because software manufacturers may decline to fix vulnerabilities in light of an impending release of a new software version.¹²⁸ Finally, phase six begins the decline in the exploitation of the vulnerability.¹²⁹

¹¹⁸ *Id.*

¹¹⁹ See Cheryl S. Massingale & A. Faye Borthick, *Risk Allocation for Computer System Security Breaches: Potential Liability for Providers of Computer Services*, 12 W. NEW ENG. L. REV. 167, 171-72 (1990). Software programmers may intentionally maintain backdoor entry into systems. *Id.* at 188. They may also hide code in a compiler, so that even a fellow developer verifying source code will not find the code. *Id.* at 193 n.143. There was some concern that developers, hired to avert problems associated with Y2K, left behind unwanted trapdoors. See Frank J. Cilluffo et al., *Bad Guys and Good Stuff: When and Where Will the Cyber Threats Converge?*, 12 DEPAUL BUS. L.J. 131, 164 (2000).

¹²⁰ See INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 25 (illustrating phase two of the vulnerability exploit cycle).

¹²¹ *See id.*

¹²² *See id.*

¹²³ *See id.*

¹²⁴ *See id.*

¹²⁵ See INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 25.

¹²⁶ *See id.*

¹²⁷ *See id.* (illustrating the relation between the amount of time vulnerability exists during phase five as compared to phases one through four).

¹²⁸ Houle & Weaver, *supra* note 48, at 19.

¹²⁹ INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 25.

The entire vulnerability and exploit life cycle typically lasts two to three years.¹³⁰ Completion of the cycle does not imply that a system has become less susceptible to compromise; computers are often vulnerable to multiple exploits at a given moment, each in a different phase of the cycle. A computer system or network faces a constant stream of vulnerabilities.¹³¹ For some styles of attack, there has been "a significant decrease in the time window from when a vulnerability is discovered to when it is widely exploited."¹³²

IV. INEFFECTIVE APPROACHES TO PREVENTING SECURITY-RELATED SOFTWARE FAILURE

Neither the government nor the market has effectively reduced or limited the potential harms from security-related software failure. The government has actively attempted to deter hacking, but has foreclosed opportunities to hold software manufacturers liable for programming blunders.¹³³ The market has been ineffective because of imperfect information about intrusions and software manufacturers' market power.¹³⁴ Further, legal doctrines impaired private contracting regarding liability and limited loss recoveries in tort.¹³⁵

A. *Government Approach Has Been Ineffective*

The federal government's first awareness of hacking has been attributed to the classic 1983 hacker movie *War Games*.¹³⁶ Congress responded in 1984 with an attempt to deter hackers by creating the Counterfeit Access Device and Computer Fraud and

¹³⁰ Houle & Weaver, *supra* note 48, at 19 (noting that this period is still the norm "despite security community and vendor efforts to raise awareness of serious security issues").

¹³¹ INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 26 (displaying the overlap of the various exploitation cycles of vulnerabilities a computer system or network may experience).

¹³² Houle & Weaver, *supra* note 48, at 10.

¹³³ See generally Calkins, *supra* note 29, at 179–85 (discussing the government's attempts to deter hackers and criminalize unauthorized computer use).

¹³⁴ See *infra* Part IV.B (discussing the vulnerability of software to security-related attacks).

¹³⁵ See *infra* notes 182–86 and accompanying text (noting the ineffectiveness of suing a software manufacturer).

¹³⁶ See Calkins, *supra* note 29, at 175 (noting the premise of this movie involves "a teenage computer whiz . . . [who] hacks into a North American Aerospace Defense Command (NORAD) computer . . . and, thinking he is playing a computer game, nearly starts a global thermonuclear war").

Abuse Act.¹³⁷ This act criminalized unauthorized computer use.¹³⁸ This law then became the Computer Fraud and Abuse Act (CFAA) in 1986,¹³⁹ and has since been amended.¹⁴⁰ The CFAA prohibits unauthorized access to computers "used in interstate or foreign commerce or communication,"¹⁴¹ which in effect means every computer on the Internet.¹⁴² Currently, all fifty states also have computer crime laws.¹⁴³

Attempting to deter hackers does not work. In the first instance, not all intruders are detected;¹⁴⁴ when they are, few intruders are reported to law enforcement agencies.¹⁴⁵ Victims, particularly businesses, do not want to risk bad publicity or copy-cat attacks.¹⁴⁶ They also wish to avoid providing their competitors with possible marketing strategies. Further, firms believe notifying law enforcement will compromise confidentiality.¹⁴⁷ Moreover, going after hackers is useless because they are usually judgment-proof.¹⁴⁸ Even when law enforcement agencies are noti-

¹³⁷ *Id.* at 179.

¹³⁸ Pub. L. No. 98-473, § 2102(a), 98 Stat. 1837, 2190-91 (1984) (criminalizing hacker activity by adding fraud and related activity in connection with computers).

¹³⁹ 18 U.S.C. § 1030 (2000).

¹⁴⁰ See Calkins, *supra* note 29, at 179. In 1994 and 1996, the *mens rea* was successively lowered to "ensure that a wide class of hackers (including foreign hackers) would fall under the statute." *Id.* In 2001, the Act was again amended to shield software manufacturers from liability for defective code. See *infra* notes 149-53 and accompanying text.

¹⁴¹ 18 U.S.C. § 1030(e)(2)(B).

¹⁴² Calkins, *supra* note 29, at 180; see also Lee, *supra* note 42, at 869 (noting that a "protected computer," under this statute, is any computer with Internet access).

¹⁴³ Calkins, *supra* note 29, at 184.

¹⁴⁴ See, e.g., Calkins, *supra* note 29, at 183 (estimating fewer than ten percent of intrusions are detected). However, skepticism is always in order when one claims to know the percent of things that are reported, because it implies one knows the number of things unreported. Some entities that conduct infiltration tests have released the success rates of such tests. See Dhillon & Smith, *supra* note 5, at 140-41. For example, the Department of Defense detected only four percent of successful friendly intrusions. See *id.*

¹⁴⁵ See Dhillon & Smith, *supra* note 5, at 141 (noting that of the few detected intrusions only twenty-seven percent of those were reported to the Defense Information System Agency (DISA), the agency responsible for computer security).

¹⁴⁶ See Calkins, *supra* note 29, at 183.

¹⁴⁷ See Stevan D. Mitchell & Elizabeth A. Banker, *Private Intrusion Response*, 11 HARV. J.L. & TECH. 699, 715 (1998).

¹⁴⁸ See Lee, *supra* note 42, at 875.

fied, success is not guaranteed,¹⁴⁹ and claims are seldom pursued.¹⁵⁰ While a law that is rarely enforced¹⁵¹ can deter, if the punishment is sufficiently large, it is clear from empirical evidence that hordes of hackers are not currently discouraged.¹⁵²

While hackers play an obvious role in security-related software failure, the most prevalent methods of attack are the result of careless programming practices by software manufacturers and sloppy system administration by software consumers. Hackers thrive on the negligence of the other two groups. Yet the government has twice foreclosed opportunities to hold software manufacturers liable for programming blunders. In one sense, deterring hackers is irrelevant to preventing software failure.¹⁵³ We usually avoid outbreaks of infectious disease by vaccination, not by a direct assault on the pathogen.

Congress missed an opportunity to hold software manufacturers liable when it amended the CFAA in 2001. The CFAA provides for a civil action against whomever "knowingly causes the transmission of a program, information, code, or command, and as a result of such conduct, intentionally causes damage without authorization to a protected computer."¹⁵⁴

¹⁴⁹ See Dhillon & Smith, *supra* note 5, at 160 (discussing the enforcement limitations the CFAA places on the government). The CFAA's prohibition on unauthorized access attaches to the government in the first instance. *Id.* While the CFAA has a carve-out for law enforcement, it does not explicitly exempt national security agencies seeking to "hack back." *Id.*

¹⁵⁰ Mitchell & Banker, *supra* note 147, at 703.

¹⁵¹ See, e.g., *United States v. Morris*, 928 F.2d 504 (2d Cir. 1991) (demonstrating that the CFAA has caused some cases to be prosecuted, such as that of Robert Morris, the author of the appropriately named, Morris worm).

¹⁵² *Id.*; see also discussion *supra* Part II.A. CERT reported 52,658 incidents, and 2,437 vulnerabilities in 2001. *CERT/CC Statistics 1988-2001*, CERT/COORDINATION CENTER, at <http://www.cert.org/stats> (last updated Oct. 4, 2002) [hereinafter *CERT/CC Statistics*].

¹⁵³ See Lawrence Lessig, *The Constitution of Code: Limitations on Choice-Based Critiques of Cyberspace Regulation*, 5 COMM'LAW CONSP'CTUS 181, 184 (1997). Lessig holds the belief that hackers are irrelevant to the broader development of the Internet.

We live life *subject to* the code [in cyberspace], as we live life subject to nature. Just as we do not choose whether to see through a wall or not, we don't choose whether to enter America Online without giving our password. Superman might choose whether to see through a wall; and hackers might be able to choose whether to enter AOL with a password. But we are neither supermen nor hackers (if such a distinction exists). We live life subject to the constraints of code; however (and by whomever) these constraints have been set.

Id.

¹⁵⁴ 18 U.S.C. § 1030(a)(5)(A) (2000).

Several judicial decisions expanded the definition of "transmission" to cover market transactions. In *Shaw v. Toshiba American Information Systems, Inc.*,¹⁵⁵ the court held that the sale of floppy disk controllers containing faulty code by a computer vendor was a knowing transmission for purposes of the CFAA.¹⁵⁶ Other cases have found CFAA violations where software and hardware vendors surreptitiously placed "time bombs" on customer machines to disable the machines at a point in the future if the customers failed to pay rent or royalties.¹⁵⁷

In apparent response to these decisions, Congress amended the CFAA, which now includes the language: "No action may be brought . . . for the negligent design or manufacture of computer hardware, computer software, or firmware."¹⁵⁸ Admittedly, Congress seems to have intended the CFAA to curb computer fraud and abuse resulting from unauthorized computer use.¹⁵⁹ Removing manufacturer liability is faithful to the original purpose. But the unfortunate result was to foreclose an opportunity for software manufacturers to be held liable for defective programming.

The most prominent missed opportunity to hold software manufacturers accountable for the business risk they create is the limitation of liability in suits over Y2K software failures.¹⁶⁰ Missing both of these opportunities is unfortunate because exposing software manufacturers to tort claims will reduce the number of intrusions and the potential harm arising from security-related software failure.

B. Market Response Has Been Inefficient

Neither software consumers nor software manufacturers have responded efficiently to the dangers from security-related software failure. The behavior of software consumers, departing from an efficient level of software failure prevention, is empiri-

¹⁵⁵ 91 F. Supp. 2d 926 (E.D. Tex 1999).

¹⁵⁶ *Id.* at 935-36.

¹⁵⁷ See *North Texas Preventive Imaging, L.L.C. v. Eisenberg*, No. CV 96-71, 1996 U.S. Dist. LEXIS 19990, at *6 (C.D. Cal. Aug. 19, 1996); *Gomar Mfg. Co. v. Novelli*, C.A. No. 96-4000, 1998 U.S. Dist. LEXIS 23458, at **35-36 (D.N.J. Mar. 24, 1998).

¹⁵⁸ 18 U.S.C. § 1030(g) (2000).

¹⁵⁹ See S. REP. No. 99-432, at 2-3 (1986), *reprinted in* 1986 U.S.C.C.A.N. 2479, 2480 (relating examples of unauthorized computer use that motivated passage of the CFAA).

¹⁶⁰ See Year 2000 Computer Date Change Act, 15 U.S.C. §§ 6601-17 (2000).

cally evident from the less than timely installation of security patches¹⁶¹ during the final two phases of a vulnerability's life cycle.¹⁶² Most telling though, is CERT's finding that over ninety-five percent of intrusions use known vulnerabilities¹⁶³ for which counter-measures are available.¹⁶⁴

Companies make software failure prevention decisions based on "their own needs or the desires of their customer base."¹⁶⁵ Some companies value security and employ counter-measures such as firewalls, intruder detection systems, and authentication techniques beyond passwords.¹⁶⁶ Others choose to spend less on security, perhaps "because they have little valuable data to protect."¹⁶⁷ While this is rational, it is not efficient. Low-security sites impose a cost on high-security sites by increasing the risk of accidents from some types of attacks, such as DDoS.¹⁶⁸ Low-security sites deploy less security than they would if they bore the full cost of their activities.¹⁶⁹ Under-consumption of security occurs for the symmetrical reason that any spending on security benefits more groups than the organization doing the spending.¹⁷⁰

¹⁶¹ Houle & Weaver, *supra* note 48, at 19.

¹⁶² The economic theory of the time value of money assumes that money gained today has more value than the possibility of gaining money tomorrow. *Time Value of Money*, ECONEDLINK, (explaining how investing money earlier pays off in the longrun), at <http://www.econedlink.org/lessons/index.cfm?lesson=em37> (last visited Nov. 3, 2002). Using this theory one can hypothesize that the benefit of a patch installed today is worth more than the benefit of a patch installed tomorrow, and we would expect to see early adoption of patches subject only to the cost of being aware of a patch, which is the cost of reading an email.

¹⁶³ INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 16.

¹⁶⁴ Countermeasures range from rigorous password administration routines to installing security patches provided by software manufacturers. See *id.* at illus. 17. One counter argument is that the "market" for security accident prevention is actually working well—that the cost of security-related software failure, given its probability of occurrence, is less than the cost of installing a security patch. See Michael P. Dierks, *Computer Network Abuse*, 6 HARV. J.L. & TECH. 307, 309 (1993) (arguing that legislative schemes focusing on computer abuse are unnecessary because "[t]he invisible hand of the market correctly resolves the problem of computer abuse"). The externalities and imperfect information discussed *infra* undercut this argument. See *infra* Part IV.B.

¹⁶⁵ Calkins, *supra* note 29, at 216.

¹⁶⁶ See COMPUTER SCI. BD., *supra* note 6, at 12–14 (discussing the use of Cryptographic authentication as an important aspect of security techniques).

¹⁶⁷ Calkins, *supra* note 29, at 216.

¹⁶⁸ See *supra* Part III.F.

¹⁶⁹ See Calkins, *supra* note 29, at 217 (explaining a low-security website's disincentive to invest heavily in security).

¹⁷⁰ See *id.* (illustrating why organizations with "deep pockets" must invest in security as opposed to those without "deep pockets").

Left alone, the market gives insufficient incentives to consume security-related resources.

Software consumers also fail to prevent security-related software failure because of imperfect information.¹⁷¹ Some customers misjudge the threat for the same reasons hackers are not deterred—intrusions are for the most part largely undetected and unreported.¹⁷² Others exhibit an “it can’t happen to me” mentality.¹⁷³ Finally, the shortage of accurate information regarding intrusions is exacerbated by a shortage of competent computer security specialists—an immediate increase in the demand for security specialists would outstrip supply.¹⁷⁴

The behavior of software manufacturers also influences the rate of security-related software failure. On the most general level, software is the source of *all* security problems.¹⁷⁵ The instant software is written to perform some function it is susceptible to compromise.¹⁷⁶ On a more practical level, preventable security-related software failure occurs with alarming frequency.¹⁷⁷ Software is rushed to market and shipped with default configura-

¹⁷¹ See Houle & Weaver, *supra* note 48, at 19 (reasoning that because consumers continue to use vulnerable software, CERT will continue to issue vulnerability notes to raise awareness on serious-security problems).

¹⁷² See Mitchell & Banker, *supra* note 147, at 708 (cautioning against empirical evidence of intrusions). Approximately “less than one in ten successful computer intrusions [are] detected.” *Id.*

¹⁷³ See Sandra Swanson & George V. Hulme, *Insuring Against Cybercrime Gets Tougher—Insurers set Stricter Policies and Change more for Cyberinsurance after Sept. 11*, INFORMATIONWEEK, Jan. 7, 2002, at 24 (illustrating how some companies think they are invincible to cybercrime), available at LEXIS, Hr Library, Infowk File. Swanson quotes Sean Magee, VP of IS for Lanier Worldwide Inc., stating that, “I doubt we’re on anyone’s hacking radar screen.” *Id.*

¹⁷⁴ See Mitchell & Banker, *supra* note 147, at 716–17 (discussing potential decreased interest in computer security as a profession, if professional licenses were required).

¹⁷⁵ See Lessig, *supra* note 153, at 184 (discussing how law and code regulate cyberspace).

¹⁷⁶ See Massingale & Borthnick, *supra* note 119, at 173 n.26 (stating that “perfectly” secure systems keep out legitimate users and are of no use to their owners).

¹⁷⁷ See, e.g., Rogers, *supra* note 25, at 1. CERT assigns security metrics to vulnerabilities. See generally *CERT/CC Vulnerabilities Notes by Metric*, CERT/COORDINATION CENTER, at <http://www.kb.cert.org/vuls/bymetric> (last visited Nov. 3, 2002). Of the top thirty exploitations, as of January 2002, twelve were buffer overflow flaws, six were cross-site scripting, and three were other failures to inspect user input. *Id.*

tions that disable security features.¹⁷⁸ Such software is replete with foreseeable vulnerabilities (e.g., buffer overflows, cross-site scripting, or unexpected operator attacks) because it trusts user input without testing to see whether the input is trustworthy.¹⁷⁹ Even after many years and thousands of examples, software manufacturers still offer programs destined for security-related software failure.¹⁸⁰ And, of course, market efficiency is impaired because many software manufacturers have market power.¹⁸¹

Historically, a number of legal doctrines have also constrained the software market. Several doctrines under the Uniform Commercial Code (UCC) have coalesced to effectively prevent contracting over liability for failures.¹⁸² For example, the UCC favors maintaining a "bargain-in-fact,"¹⁸³ yet it is often cheaper for a software manufacturer to refund license money than to cure a defect.¹⁸⁴ Finally, courts have been reluctant to grant economic damages in tort.¹⁸⁵ Considering that a large proportion of damage from security-related software failure is economic, victims have been unable to be made whole through tort.¹⁸⁶

¹⁷⁸ See Massingale & Borthnick, *supra* note 119, at 188 (stating that vendors often configure systems with default configurations, disabling available security features).

¹⁷⁹ Rogers, *supra* note 25, at 14 (emphasizing that "data must be considered *untrustworthy* and be made *trustworthy* before being used," something programmers frequently fail to recognize).

¹⁸⁰ See Houle & Weaver, *supra* note 48, at 19 (recognizing that many systems, both new and old, are still vulnerable).

¹⁸¹ See *United States v. Microsoft*, 253 F.3d 34, 51 (D.C. Cir. 2001) (holding that Microsoft has the ability to affect the market's total output in that its market power provides them "the ability to cut back the market's total output and so raise price").

¹⁸² See Daniel T. Perlman, *Who Pays the Price of Computer Software Failure?*, 24 RUTGERS COMPUTER & TECH. L.J. 383, 393 (1998) (stating that integration clauses, warranty disclaimers and limitations of remedy clauses usually protect software sellers from liability).

¹⁸³ See *id.* at 389 (addressing the UCC's approach to bargained for agreements).

¹⁸⁴ See Donald R. Ballman, *Commentary, Software Tort: Evaluating Software Harm by Duty of Function and Form*, 3 CONN. INS. L.J. 417, 419 (1997) (stating that the only liability faced by software manufacturer for a defective product is replacement of the product or payments "not to exceed the original licensing fees (sale price)").

¹⁸⁵ See, e.g., *Hou-Tex, Inc. v. Landmark Graphics*, 26 S.W.3d 103, 106-07 (Tex. App. 2000) (finding that the economic loss rule precludes any duty in tort "and damages are not recoverable unless they are accompanied by actual physical harm to persons or their property").

¹⁸⁶ Computer transactions are considered "the sale of goods" and therefore have been guided by the UCC that trumps negligence claims. Perlman, *supra* note 182, at 388. This further emphasizes the fact that negligence claims are

V. A DETERRENCE MODEL BUILT ON TORT

It is time to consider laying more blame for security-related software failure where it is due—on software manufacturers and consumers.¹⁸⁷ This is true because both groups contribute significantly to the probability of security-related software failure, as they are the lowest cost avoiders for some of the risk, and because hacker-based deterrence by itself is not effective.¹⁸⁸ While it is clearly desirable to place liability on the party primarily responsible for intrusions, the hacker, placing back-up liability can lead to a second-best solution when the primary party is judgment-proof.¹⁸⁹ The analysis of deterrence models directed at software manufacturers and consumers is broken into two cases distinguished by whether a vulnerability has been patched.

A. *Software Manufacturer Liability: The Pre-Patch Case*

In the first case, software manufacturers sell software containing a design flaw and customers install and configure the software according to instructions provided by the manufacturer.¹⁹⁰ A hacker identifies and exploits a design flaw, which in turn damages a victim.¹⁹¹ All this occurs before the software company patches the software.¹⁹² While this case does not represent the majority of intrusions,¹⁹³ almost any vulnerability goes through a phase when only a few elite hackers exploit it on a limited number of machines and the software manufacturer has yet to create a

usually not applied and are not proper when economic losses are involved. *Id.* at 388–89.

¹⁸⁷ See Epstein & Tancer, *supra* note 51, at 664 (“ISPs and users could find themselves subject to direct, contributory, or vicarious criminal or civil liability. . .”).

¹⁸⁸ See generally WILLIAM M. LANDES & RICHARD A. POSNER, *THE ECONOMIC STRUCTURE OF TORT LAW* 196 (1987) (explaining optimal care and the interest of tortfeasors to avoid liability).

¹⁸⁹ See *id.* at 200 (illustrating that sometimes when the best solution is not available, because a party is judgment proof, it is more efficient to obtain the second best solution).

¹⁹⁰ See Houle & Weaver, *supra* note 48, at 19 (explaining that vendors still create technology that contains “exploitable security vulnerabilities”).

¹⁹¹ See Ballman, *supra* note 184, at 422–23. The most immediate victims are the commercial and private software consumers. *Id.* Other victims may include parties with whom the software consumer has a market relationship, including employees and clients. *Id.* at 423 (noting that both private and commercial parties are injured by malfunctioning software).

¹⁹² See Houle & Weaver, *supra* note 48, at 19 (raising concerns that computer resources are still susceptible to attacks even when computers are patched in a less than timely manner).

¹⁹³ See, e.g., INCIDENT AND VULNERABILITY TRENDS, *supra* note 117, at illus. 17.

patch.¹⁹⁴ In cases such as these, holding software manufacturers liable encourages them to prevent design flaws that lead to security-related software failure.¹⁹⁵

One approach is to hold software manufacturers strictly liable. This would entail holding them liable for all harm caused as a result of a design flaw allowing a security-related software failure. There are several economic reasons for courts to impose strict liability. First, software manufacturers facing strict liability will efficiently adjust every aspect of their behavior.¹⁹⁶ Software manufacturers influence the total risk level by choosing how much care to take in programming software and how much software to sell.¹⁹⁷ When software manufacturers are strictly liable they bear all the costs of accidents.¹⁹⁸ They will take due care¹⁹⁹ to investigate potential software failure when preventing the failures is less costly than paying for the resulting harm.²⁰⁰

Software manufacturers will also adjust how much software they sell.²⁰¹ Since producers face the costs of accidents, they will most likely adjust the market price to equal production costs plus accident costs.²⁰² The higher market price results in fewer units sold.²⁰³ Strict liability influences software manufacturers to

¹⁹⁴ See *id.* at illus. 25 (illustrating this phenomena by way of a bell curve).

¹⁹⁵ Cf. Ballman, *supra* note 184, at 427 (noting that current law allows software manufacturers to limit their liability simply by stating the limitations in a standard licensing agreement accompanying their products).

¹⁹⁶ See Steven Shavell, *Strict Liability Versus Negligence*, 9 J. LEGAL STUD. 1, 2-4 (1980) (explaining that strict liability generally demands that sellers take appropriate actions to guarantee efficiency).

¹⁹⁷ See *id.* (reasoning that the amount of care taken by the seller and the amount produced will determine the seller's risk level).

¹⁹⁸ See *id.* at 2-3 (reinforcing that strict liability carries with it a burden of liability for all costs). Strict liability ignores "[c]ausal or other reasons for limiting the scope of liability." *Id.* at 3 n.4.

¹⁹⁹ See LANDES & POSNER, *supra* note 188, at 58-62 (discussing and exemplifying an analysis of optimal or due care and the actions a risk neutral person should take in order to minimize the social costs of accidents).

²⁰⁰ See *generally id.* at 58-60 (explaining through graphs due care and its resulting consequences in regards to expected damages).

²⁰¹ See, e.g., *id.* at 275 (stating the results of strict liability and the resulting behavior to achieve efficiency).

²⁰² See Shavell, *supra* note 196, at 3 (emphasizing that "customers will face the 'socially correct' price for the product").

²⁰³ See *id.* at 3 (reiterating that producers will most likely recover the costs of accidents by increasing the price of the product, resulting in lower sales of that good).

behave efficiently for both quantity produced and level of care taken.²⁰⁴

The legal reason to use strict liability might arise from the common law doctrine regarding abnormally hazardous activities.²⁰⁵ Abnormally hazardous activities are identified by the:

(a) existence of a high degree of risk of some harm to the person, land or chattels of others; (b) likelihood that the harm that results from it will be great; (c) inability to eliminate the risk by the exercise of reasonable care; (d) extent to which the activity is not a matter of common usage; (e) inappropriateness of the activity to the place where it is carried on; and (f) extent to which its value to the community is outweighed by its dangerous attributes.²⁰⁶

Manufacturing software in general and software providing network functions in particular, is arguably abnormally hazardous.²⁰⁷ There is a high degree of risk that computer and network vulnerabilities will be exploited—such exploitation is routine (factor a).²⁰⁸ There is a high likelihood the harm that results will be great (factor b).²⁰⁹ There is general agreement over the inability to eliminate the risk by exercising reasonable care—it is difficult to write bug-free or hack-free code (factor c).²¹⁰ The fact that careless programming in some areas, such as network protocols and applications, creates much more risk of security-related software

²⁰⁴ See *id.* at 4; LANDES & POSNER, *supra* note 188, at 66–69. Due care always considers *how* parties conduct themselves (level of care), it rarely considers *whether* they conduct themselves (level of activity). Optimal deterrence requires sellers and customers to adjust both levels. See *id.* at 67–68. It is true that pointing strict liability at an injurer does not necessarily encourage victims to adjust their activity level efficiently. *Id.* at 69. When victims are customers of injurers, as will often be the case with software manufacturers, modifying either the injurer's or the victim's activity level adjusts the other's (what is produced is purchased). *Id.* at 275. But when victims are strangers to injurers, strict liability directed at injurers with a defense of contributory negligence may or may not be more efficient than a negligence rule. *Id.* at 69.

²⁰⁵ See LANDES & POSNER, *supra* note 188, at 111 (stating that under common law many areas of strict liability are classified as hazardous activities, for example the keeping of wild animals).

²⁰⁶ RESTATEMENT (SECOND) OF TORTS § 520 (1977).

²⁰⁷ See Rogers, *supra* note 25, at 13–19 (explaining that function of software and its manufacturing carry hazardous possibilities).

²⁰⁸ See CERT/CC Statistics, *supra* note 152 (reporting that there were 52,658 incidents and 2,437 vulnerabilities reported in 2001).

²⁰⁹ See, e.g., Dhillon & Smith, *supra* note 5, at 144–45 (stating that highly classified information such as war plans and detailed maps to nuclear sites were infiltrated).

²¹⁰ See Lee, *supra* note 42, at 878 (stating that any regulation would only have a temporary impact and that “no code-based solution is likely to regulate hackers effectively”).

failure than careless programming elsewhere demonstrates that factor e, an essentially spatial requirement, can be applied to cyberspace.²¹¹ On the other hand, writing software to perform networking functions has become common place (factor d); and the benefits of software are likely outweighed by its dangerousness (factor f). While the argument that the benefits of software innovation outweigh its costs has prevailed at the policy-making level,²¹² a court's determination to the contrary is not thereby foreclosed.

Even if courts do not find strict liability, software manufacturers should be liable for the harm that results when they fail to take due care preventing design flaws that allow security-related software failure.²¹³ Proposing a due care standard requires that the standard be defined. One possible definition is apparent from the previous discussion of common hacking attacks: does the software trust user input without testing the input for trustworthiness.²¹⁴ In practice, this test would operate as strict liability when hackers use known exploitations, but would shield software manufacturers whenever hackers devise a truly innovative method to cause software failure.²¹⁵

The worst choice, to find no liability, is the status quo.²¹⁶ Software manufacturers are eager to reduce time to market. Some manufacturers who otherwise might take due care preventing security-related software failure are forced under a no liability

²¹¹ Viruses and worms are just the software incarnations of a broader group of potential harms arising when hostile human creates devise the ability to self-propagate. Other examples are genetically altered biological agents and robotic nano-technology devices. See discussion *supra* Part III.D.

²¹² 15 U.S.C. § 6601 (2000) (stating the benefits of software innovation and the possible disruptions from computer failure).

²¹³ See COMPUTER SCI. BD., *supra* note 6, at 14 (stating that a possible option would include exposing software and system vendors to increased liability for system breaches).

²¹⁴ See discussion *supra* Part III.D.

²¹⁵ See *OpenBSD Security*, OPENBSD (describing a new and particularly exotic hack that, although a patch was issued before the vulnerability was ever actually exploited by a hacker, the design flaw allowing the hack went back for fifteen years), at <http://www.openbsd.com/security.html> (last updated Mar. 19, 2002).

²¹⁶ Design flaws that prevent software from performing core functionality rarely give rise to liability beyond returning the purchase price; design flaws that create security vulnerabilities have historically presented even less liability. This is because software manufacturers are alert compared to their user base which requires months or years to install security patches. But, software manufacturers lag behind the hacker vanguard. See *supra* notes 165–70 and accompanying text.

rule to lower their standard of care in order to compete with manufacturers not taking due care.²¹⁷ The no liability legal regime punishes those companies who would like to take due care.²¹⁸ Imposing liability will allow software manufacturers to take due care.²¹⁹

Another reason software manufacturers should be liable is that they are the lowest cost avoiders. This is true, if for no other reason than that they have the best information about their software product. With the noteworthy exception of open source software, software manufacturers jealously guard most source code.²²⁰ This guarded attitude prevents software consumers and third parties from estimating the likely numbers of intrusions a software product will allow over its lifetime. This leaves producers with the best ability to estimate the total number of intrusions likely to occur and include the cost of this risk in the market price.

Holding software manufacturers liable will not bankrupt the industry. Insurance is available for losses stemming from denials of service and viruses.²²¹ Software manufacturers can spread the cost of liability over the entire user base.

B. Software Manufacturer and Customer Liability: The Post-Patched Case

In the second case of security-related software failure, a software manufacturer sells software containing a design flaw. The customer correctly installs and configures the software. A hacker identifies the design flaw and exploits it on a few systems. Then the software manufacturer becomes aware of the flaw, patches the software, and alerts its customers. The customer fails to install the patch and a hacker exploits the flaw on the customer's computer system, thereby harming a victim.²²² Initial proposals to prevent security-related software failure focused

²¹⁷ See Shavell, *supra* note 196, at 5 (explaining that if all manufactures do not act with due care, those that do will be forced out of the market because of increased costs and decreased sales).

²¹⁸ See LANDES & POSNER, *supra* note 188, at 62 (explaining the economic impact of no liability as it affects the incentives of the parties).

²¹⁹ See *id.* at 63 (describing the economic impact of strict liability).

²²⁰ See David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241, 254 (2001) (stating that open source software is premised on free transparent distribution rather than exclusion).

²²¹ See Swanson & Hulme, *supra* note 173 (stating that supplemental insurance is available for viruses, security breaches, and DoS attacks).

²²² See discussion *supra* Part II.A (discussing potential network disasters and popular techniques for causing network failure).

almost exclusively on this second case²²³ for good reason: system mis-configuration, weak password systems, and delay in installing security patches are all within the control of software customers. While exclusive liability is appropriate for harms resulting from mis-configuration and password problems, the need to install security patches arises in the first instance because of the software manufacturer's actions.

Tort law assigns a number of familiar doctrines to the liability of multiple tortfeasors: place all liability on one party or the other, hold one party strictly liable with a defense of contributory negligence, adopt proportional liability, or allow joint and several liability.²²⁴ Each doctrine might play a role in efficiently deterring security-related software failure.

C. Assigning Liability Only to One Party

Currently the law regarding security-related software failure assigns liability to one party.²²⁵ Liability exists only with software consumers, regardless of whether the software manufacturer could have eliminated the underlying design flaw.²²⁶ When security-related software failure harms a software consumer, the consumer pays the full cost of the harm.²²⁷ This is essentially strict liability.

When failures harm third parties, courts are increasingly likely to review software consumer behavior for negligence. One group of software consumers, Internet Service Providers (ISPs), are likely to be liable when they "knew or should have known" of either a subscriber's hacking activities or of the security patches that could have reduced system vulnerability but went unin-

²²³ See Gripman, *supra* note 32, at 178–82. But see Calkins, *supra* note 29, at 214–17. Calkins acknowledges the attractiveness of liability on "middlemen" such as Internet Service Providers (ISPs), but rejects the idea as being inefficient. Calkins, *supra* note 29, at 216. Calkins does not explain why network externalities should not be included in calculations of efficiency. See *id.*

²²⁴ See LANDES & POSNER, *supra* note 188, at 213–15 (defining joint tortfeasor from an economic standpoint).

²²⁵ See Calkins, *supra* note 29, at 188.

²²⁶ See *Hou-Tex, Inc. v. Landmark Graphics*, 26 S.W.3d 103, 105–07 (Tex. App. 2000) (finding the fact that the software consumer suffering only economic damages precludes any duty and therefore recovery in tort by the software manufacturer).

²²⁷ *Id.* at 107.

stalled.²²⁸ Courts have found ISPs contributorily negligent for defamation,²²⁹ copyright infringement,²³⁰ and will likely do so in other areas, such as patent infringement.²³¹

Holding ISPs contributorily negligent for a hacker's harm is similar. Applying the "should have known" standard to software consumers is straightforward. The question becomes, whether the software customer failed to install a patch that was available at the time the security-related software failure occurred. Holding software consumers directly negligent however, requires four elements: (1) a duty to use reasonable care; (2) a breach of that duty; (3) a reasonably close causal connection between the conduct and resulting injury; and (4) an actual loss or damage.²³²

Any claim of negligence must define reasonable care. Historically, there was no generally accepted industry-wide standard of reasonable care for software customers.²³³ This was in part due to a perceived potential "arms race"—that as hackers penetrated any given standard of care, a more burdensome standard of care would be required.²³⁴ This concern is well placed for some security

²²⁸ Calkins, *supra* note 29, at 218 (stating that ISP's are generally not liable for misuse of their systems unless they know or should have known the misuse was occurring).

²²⁹ See, e.g., *Stratton Oakmont v. Prodigy*, 1995 WL 323710, *4 (N.Y. Sup. Ct. May 24, 1995), *reh'g denied*, 1995 WL 805178 (N.Y. Sup. Ct. 1995) (finding that an ISP that held itself out to the public and its members as controlling the content of its computer bulletin boards was a publisher for purposes of a libel suit). *But see* *Cubby Inc. v. CompuServe Inc.*, 776 F. Supp. 135, 137, 144 (S.D.N.Y. 1991) (holding that an ISP with no more editorial control over a publication than a public library, book store, or newsstand is not a "publisher" for purposes of a libel suit).

²³⁰ See *Sega Enters. Ltd. v. Maphia*, 948 F. Supp. 923, 939 (N.D. Cal. 1996) (granting summary judgment on manufacturer's trademark infringement claim). *But see* *Religious Tech v. Netcom*, 907 F. Supp. 1361, 1381 (N.D. Cal. 1995).

²³¹ See Keith E. Witek, *Software Patent Infringement on the Internet and on Modern Computer Systems—Who is Liable for Damages?*, 14 SANTA CLARA COMPUTER & HIGH TECH. L.J. 303, 381–82 (1998) (stating that the theory of "vicarious liability and inducement can be used to find culpable and/or volitional Internet-enabling entities while offering justified protection to passive Internet-enabling entities").

²³² RESTATEMENT (SECOND) OF TORTS § 281 (1977).

²³³ Gripman, *supra* note 32, at 183 n.95. It should be noted, however, that this analysis is based on facts pre-dating the Internet explosion. See Calkins, *supra* note 29, at 216–17; Ballman, *supra* note 184, at 460–61 (suggesting the duty of care standard for software should mirror the duty of care of the profession that the software expert system represents; accounting software must satisfy an accountant's duty of care).

²³⁴ See Calkins, *supra* note 29, at 216–17 (emphasizing that hackers would penetrate any given level of reasonable care, such that the standard would continually increase and the cost of the standard would also increase).

issues. An evolution in the way users are authenticated, from current technology (passwords)²³⁵ to future technology (retinal scans), would likely evoke hacker innovations.²³⁶ Despite this concern, a standard of care is evolving.²³⁷ For the cases where a patch exists for a known vulnerability, reasonable care should require keeping security patches up-to-date.²³⁸

Any claim of negligence must also define proximate cause.²³⁹ For many attacks, hackers loop through numerous computers, some of which are located in foreign jurisdictions, prior to assault.²⁴⁰ This looping is usually possible only because of lax security on numerous intermediate machines.²⁴¹ The question is whether the intermediate computers should share liability. From a deterrence perspective, only the target of the attack need be held liable; upstream computers are irrelevant.²⁴² Each computer in

²³⁵ See Lee, *supra* note 42, at 859–61 (providing a history of how current password technology is the result of a technological arms race between systems administrators and hackers).

²³⁶ Encryption technology is a prime example. The minimum key length considered secure keeps increasing as computer power increases. See William A. Hodkowski, Comment, *The Future of Internet Security: How New Technologies Will Shape the Internet and Affect the Law*, 13 SANTA CLARA COMPUTER & HIGH TECH. L.J. 217, 227–28 (1997) (comparing 56 bit key “weak” encryption with 112 bit key “strong” encryption). Quantum computing may render current encryption systems ineffective. *Hacked, Cracked, Broken, or Otherwise Compromised Encryption Technologies*, ASERITECH.COM, at <http://www.asiartech.com/threats/threat.htm> (last visited Nov. 4, 2000).

²³⁷ See generally *The Twenty Most Critical Internet Security Vulnerabilities*, SANS INSTITUTE, May 2, 2002 (updating the computer vulnerability list from ten to twenty, thus helping to trace the exploitation of security flaws via the Internet), at <http://www.sans.org/top20.htm> (last visited Nov. 4, 2002) [hereinafter *Internet Security Vulnerabilities*]. Vulnerabilities discussed include accounts with no passwords or weak passwords, large numbers of open ports and bind weaknesses. See *id.*; see generally *UNIX Security Checklist v2.0*, CERT/COORDINATION CENTER (discussing the steps to improve the UNIX operating system security), at http://www.cert.org/tech_tips/usc20_full.html (last visited Nov. 4, 2002).

²³⁸ See *Internet Security Vulnerabilities*, *supra* note 237, at para. G1.4–G1.5 (noting that in order to protect against these system vulnerabilities, “[r]emove unnecessary software, turnoff unneeded services, and close extraneous ports”).

²³⁹ KENNETH S. ABRAHAM, *THE FORMS AND FUNCTIONS OF TORT LAW* 117 (1997).

²⁴⁰ See Sinrod & Reilly, *supra* note 28, at 197–98; see also Calkins, *supra* note 29, at 214.

²⁴¹ Calkins, *supra* note 29, at 214 (illustrating a variety of ways a hacker can exploit “middleman” i.e., a computer or computer system).

²⁴² Upstream computers however, are not irrelevant to liability for denial of service attacks, where the penultimate computers serve as launching pads. See *supra* Part V.B (discussing liability rules that could prevent security-related software failure).

an attack chain is as likely to be the target of a future attack.²⁴³ Placing all of the liability on the actual site of the security-related software failure gives each computer owner equal incentive to take care.²⁴⁴

Any negligence claim must also establish damages.²⁴⁵ While there has historically been a bar against recovery for economic damages in negligence suits,²⁴⁶ the trend in modern courts allows recovery for economic harm.²⁴⁷ Congress similarly recognizes the need for economic damages in cases involving hacking—the CFAA explicitly allows victims to recover economic damage.²⁴⁸

A plaintiff claiming negligence is susceptible to a countercharge of voluntary assumption of risk; users connect to the Internet at their own risk.²⁴⁹ The response to this countercharge is that Internet users do not voluntarily assume risk because they do not accurately perceive it.²⁵⁰ Even if plaintiffs have a contract stating otherwise, they should not be prevented from a remedy under a theory of assumption of risk,²⁵¹ because “[h]igh information costs relative to the benefits of the information may defeat voluntary contracting.”²⁵²

²⁴³ Note, *Addressing the New Hazards of the High Technology Workplace*, 104 HARV. L. REV. 1898, 1899 (1991) (suggesting that as computers are progressively more linked together locally, nationally and internationally, they are more susceptible to hacker attacks).

²⁴⁴ This test would apply differently in cases of DoS and DDoS attacks. See *supra* Part V (discussing liability rules that could prevent security related software failure).

²⁴⁵ ABRAHAM, *supra* note 239, at 7.

²⁴⁶ See Gripman, *supra* note 32, at 176; see also Calkins, *supra* note 29, at 209.

²⁴⁷ Gripman, *supra* note 32, at 177 (recognizing the fundamental unfairness of the economic loss rule and discussing the courts’ trend towards allowing recovery for purely economic losses).

²⁴⁸ See 18 U.S.C. § 1030(g) (2000); see also Calkins, *supra* note 29, at 209. States permitting recovery for hacking victims provide a range of damage measures. See, e.g., VT. STAT. ANN. tit. 13, § 4106 (2001).

²⁴⁹ ABRAHAM, *supra* note 239, at 157 (stating that some courts have found that a plaintiff’s conscious, non-negligent assumption of the risk cannot be used by a defendant as a defense).

²⁵⁰ See *supra* Part IV (explaining how the government and the market have failed to address security related software failure causing consumers to perceive an altered assumption of risk); see also Shavell, *supra* note 196, at 5–6 (“when the knowledge is not perfect, there is generally scope for the use of liability, and the relative performance of liability rules depends on the precise nature of the imperfection in knowledge”).

²⁵¹ See Shavell, *supra* note 196, at 6.

²⁵² LANDES & POSNER, *supra* note 188, at 282.

Holding software consumers to a negligence standard will not by itself lead to an efficient outcome.²⁵³ The software manufacturer and the consumer are joint injurers,²⁵⁴ for example, if the customer failed to install a patch that would have prevented harm, but the patch would not have been needed had the design flaw been repaired by the manufacturer prior to sale. Imposing negligence on customers allows software manufacturers to avoid accident costs entirely.²⁵⁵ Consequently, manufacturers have no incentive to take any care preventing vulnerabilities.²⁵⁶ At best, customers liable for negligence might indirectly cause software manufacturers to adjust their level of care.²⁵⁷ If installing patches were costly, customers would prefer software expected to require few patches, assuming the customers seek to avoid liability for negligence. On the other hand, if patching is easy and inexpensive, customers will not indirectly influence manufacturer care level because they are indifferent.

Of course, all liability could also be pointed at software manufacturers, regardless of the behavior of software consumers—the proposal in the “pre-patch” case, to hold software manufacturers liable (either strictly or in negligence), is just such a regime.²⁵⁸

D. *Strict Liability with Contributory Negligence*

Since software sellers are in a market relationship with their customers, holding one party strictly liable with a defense of contributory negligence can lead to efficient deterrence.²⁵⁹ Strict lia-

²⁵³ See generally *id.* at 190–201 (noting that “it is sometimes efficient to impose back-up liability to achieve a second-best solution in circumstances where the best solution is not feasible”). Increased law enforcement spending might influence hacker deterrence, subject to the reservations stated earlier. See discussion *supra* Part IV.

²⁵⁴ See Landes & Posner, *supra* note 188, at 191 (explaining how a joint tort occurs successively rather than simultaneously, where “one tortfeasor aggravates an injury inflicted by the other . . .”).

²⁵⁵ See Shavell, *supra* note 196, at 4 (stating that if customers perceive the risks, they also bear accident losses).

²⁵⁶ *Id.* at 5.

²⁵⁷ See, e.g., *OpenBSD Project Goals*, OPENBSD (demonstrating at least one example of an operating system manufacturer who has made security a central marketing issue by listing the Company’s goal to be the number one most secure operating system), at <http://www.openbsd.com/goals.html> (last visited Nov. 4, 2002).

²⁵⁸ See *supra* notes 190–94 and accompanying text.

²⁵⁹ See LANDES & POSNER, *supra* note 188, at 274–75 (noting that what is produced is purchased). But *c.f. supra* note 204 and accompanying text (explaining why this theory may not be true if the victim is a stranger).

bility with a defense of contributory negligence can be applied two ways, either of which can lead to efficient outcomes.²⁶⁰

The first way is to hold software manufacturers strictly liable for all damages, but allow them a defense, when customers fail to exercise due care, such as failing to patch software.²⁶¹ In this way victims would collect damages from the software manufacturer or the customer, as appropriate. Software manufacturers facing strict liability will adjust their level of care and their level of activity.²⁶² As previously stated, they bear accident costs. For example, software manufacturers will remove design flaws allowing security-related software failure when doing so is less costly than compensating victims. These manufacturers will recover expected accident costs through higher market prices for software.²⁶³ Due to increased prices, customers will also take due care to avoid liability and they will adjust their activity level in accordance with that of software manufacturers.²⁶⁴

The second way to orient liability is a mirror of the first: hold software consumers strictly liable with a defense of contributory negligence against software manufacturers.²⁶⁵ The effect of this second orientation is almost identical to holding software manufacturers to a level of due care. In both cases, manufacturers have incentives to take due care-by avoiding well-known design flaws and by patching known vulnerabilities-while customers must shoulder the costs of unexpected failures.²⁶⁶ The only practical difference between these two rule configurations is that when third parties are harmed by unexpected failures, strict liability for software consumers with the defense provides a remedy, whereas software manufacturer negligence does not.²⁶⁷

²⁶⁰ See LANDES & POSNER, *supra* note 188, at 275-78 (describing how different allocations of strict liability with contributory negligence lead to efficiency where parties participate in perfect market).

²⁶¹ See *id.* at 278 (explaining that this defense is one possibility, in the case of strict liability, for the manufacturer to monitor the level of care being displayed by the consumer).

²⁶² See *id.* (suggesting that sellers who want to induce consumers to take greater care in exchange for a lower price could have to also monitor the consumer's care).

²⁶³ See Shavell, *supra* note 196, at 4 (indicating that market price will equal production costs plus accident costs).

²⁶⁴ *Id.* at 6.

²⁶⁵ Shavell, *supra* note 196, at 6-7.

²⁶⁶ See discussion *supra* Part V.B.

²⁶⁷ See LANDES & POSNER, *supra* note 188, at 79 (explaining that strict liability with a defense of contributory negligence shifts the accident losses to the injurers even when both parties use due care).

In choosing between the two ways to orient liability between customers and software manufacturers, it is necessary to determine which has the best information.²⁶⁸ To understand why, consider the role that information plays on customers' behavior.²⁶⁹ When customers are strictly liable, they seek to buy products with the lowest full price, which is market price plus expected accident costs.²⁷⁰ If customers know how much care software manufacturers are taking, they are able to correctly price software.²⁷¹ They will also require software manufacturers to take due care.²⁷² But when software consumers have trouble accurately perceiving risk²⁷³ (i.e., reverse engineering software is illegal)²⁷⁴ they are unable to accurately impute the cost of accidents into the full price of software ownership.

Software manufacturers, on the other hand, have several ways of measuring whether software consumers are taking due care. They can monitor the number of downloads for security patches, or the number of calls to the call center, and thereby estimate how much care is being taken by the customer base and how much risk remains.

Strict liability with a defense for contributory negligence does have a proximate causation element.²⁷⁵ This is appropriate for many automatic and semi-automatic attacks. Even though a virus or worm infects multiple upstream computers, only the target of the attack (and software manufacturer) should be liable, for the same reason we ignored upstream computers in a looped attack.²⁷⁶ When viruses and worms must infect a computer in order to cause harm, software consumers cause their own harm, in the case of patched software, or software manufacturers are the

²⁶⁸ LANDES & POSNER, *supra* note 188, at 286–87 (stating that for the law to be efficient it should place liability on the party who has the information or who can obtain it at the lowest cost).

²⁶⁹ *Id.* at 287.

²⁷⁰ Shavell, *supra* note 196, at 4–5.

²⁷¹ See, e.g., *id.* at 3 (illustrating how “socially correct” prices are formulated for taxi drivers under strict liability).

²⁷² *Id.*

²⁷³ See discussion *supra* Part IV.B.

²⁷⁴ See, e.g., 17 U.S.C. § 1201(f) (Supp. 2002) (prohibiting reverse engineering of copyrighted materials under the Digital Millennium Copyright Act).

²⁷⁵ See ABRAHAM, *supra* note 239, at 173 (stating that under a system of strict liability engaging in the activity must not only be the cause in fact, but also the proximate cause of the injury to the plaintiff).

²⁷⁶ See discussion *supra* Part V.B (suggesting that those who “knew or should have known” about system vulnerabilities are liable).

cause of all the harm.²⁷⁷ Strict liability with a defense of contributory negligence will optimally prevent such attacks.²⁷⁸

E. Proportional Liability

Some viruses and worms cause harm to computers they have not infected.²⁷⁹ The foremost examples are worms that distribute attack tools to vulnerable machines in order to launch a DDoS attack against an otherwise immune machine.²⁸⁰ Each intermediate machine involved in the attack shares some of the responsibility for the harm caused by the attack.²⁸¹ Owners of infected computers can be liable up to the proportion that their computer contributed to the overall attack.²⁸²

Again, the same question arises over the liability of upstream computers in DDoS attacks that occur with looped computers. Computers infected early on by a worm provide a platform for the worm to infect other computers,²⁸³ yet some of these machines might be inoculated prior to the date a DDoS attack is launched.²⁸⁴ From a deterrence perspective, inoculated com-

²⁷⁷ See Robin A. Brooks, *Detering the Spread of Viruses Online: Can Tort Law Tighten the 'Net'?*, 17 REV. LITIG. 343, 369 (discussing mutual negligence between a software manufacturer's failure to warn a user of a virus and a consumer failing to utilize proper virus scanning mechanisms).

²⁷⁸ LANDES & POSNER, *supra* note 188, at 80 (reiterating that a contributory negligent defense against strict liability creates a level of due care both to manufactures and consumers).

²⁷⁹ *Virus Information, How Viruses are Contracted*, COMPUTERHOPE.COM (noting that viruses can be contracted to other computers via floppy disks, the Internet, emails and attachments), at <http://www.computerhope.com/vlist.htm#02> (last visited Nov 4, 2002).

²⁸⁰ See Sinrod & Reilly, *supra* note 28, at 194 (explaining that hackers use multiple computers to attack the computer from various "launch points").

²⁸¹ *Id.* (stating that hackers place a "daemon" on a third-party computer and then distribute the source of attacks across a larger pool of third-party computers making it harder for the target server to block attacks).

²⁸² One difficulty with this approach arises because hackers can mask the source of a denial of service attack by a process of "IP spoofing." *Hacker Threat - IP Spoofing Attacks*, UK SECURITY ONLINE (defining IP spoofing to be when a hacker forges the source address information so that it appears the source is coming from somewhere else), at <http://www.uksecurityonline.com/threat/ips spoof.php> (last visited Nov. 4, 2002).

²⁸³ See generally Houle & Weaver, *supra* note 48, at 12 (discussing how some worms are "self-propagating" and use random target selection to target vulnerable systems).

²⁸⁴ Jon Swartz, *Code Red Worms into 150,000 Computers*, USA TODAY, Aug. 2, 2001, at B3 (stating that this worm may not have spread as quickly due to businesses inoculating their computers with security software), available at 2001 WL 5468277. But see D. Ian Hopper, *Officials Warn of Internet Threat*, AP

puters should be exonerated. With each inoculation, owners of the remaining computers bear additional incentive to prevent their machines from being used in the DDoS attack. This is desirable because it would, in the extreme, prevent the attack entirely.

VI. CONCLUSION

The shocking sloppiness of software manufactures and tardiness of software consumers to patch software susceptible to security-related failure explains the enormous security shortfall found in computers and networks attached to the Internet. The tort system offers a direct way to optimally deter software manufacturers and customers from failing to take due care in avoiding security vulnerabilities. A strict liability standard for security breaches arising before software is patched encourages software manufacturers to take due care, to internalize the cost of accidents, and to raise prices to efficient levels.

The risk of security-related software failure can also be reduced significantly by holding software manufacturers strictly liable, with a defense of contributory negligence against customers, when patches are available to protect vulnerable software, but the patches have gone uninstalled. Applying these theories effectively makes it possible to reduce the likelihood of a large-scale disaster due to security-related software failure.

ONLINE, July 29, 2001 (illustrating officials' frustrations with the worm attacks even when software inoculation is available), *available at* 2001 WL 25488283.