"SNAKE-OIL SECURITY CLAIMS" THE SYSTEMATIC MISREPRESENTATION OF PRODUCT SECURITY IN THE E-COMMERCE ARENA

John R. Michener, Ph.D.[*]
Steven D. Mohan, D.CS.[**]
James B. Astrachan, J.D., LL.M.[***]
David R. Hale, J.D.[****]

Cite as: John R. Michener, Steven D. Mohan,
James B. Astrachan and David R. Hale,
"Snake-Oil Security Claims" The Systematic Misrepresentation of Product Security in the E-Commerce Arena
9 MICH. TELECOMM. TECH. L. REV. 211 (2003),
available at http://www.mttlr.org/volnine/Michener.pdf

[*] Dr. John R. Michener is Chief Scientist and Vice President for Business Development of BBX Technologies. During his career, Dr. Michener has served as Chief Technology Officer of Enterprises Solutions, Security Architect at Novell, and Chief Technology Officer and Vice President for Engineering at Wave Systems Corp. Dr. Michener has published extensively and holds numerous related patents. *Dr. Michener may be contacted at jrmichener@ieee.org.*

[**] Dr. Steve Mohan is the former Area Chairman of the College of Information Systems & Technology at the Southern Colorado Campus of the University of Phoenix. He previously served as Senior Systems Architect at WorldCom. Dr. Mohan, a retired senior U.S. Air Force Officer, is an expert in implementation and integration of advanced technology. He has published numerous papers and presentations on the subject. *Dr. Mohan may be contacted at stevekathymohan@netzero.net.*

[***] James B. Astrachan is a principal of the law firm of Astrachan Gunst & Thomas, P.C. He is an adjunct professor at the University of Maryland Law School where he teaches Trademark Law and Unfair Competition and at the University of Baltimore Law School where he teaches Mass Media and Intellectual Property Law. *Mr. Astrachan may be contacted at jastrachan@agtalawyers.com.*

[****] David R. Hale is Associate Counsel for Intellectual Property and Privacy at Ameritrade Holding Corporation and an adjunct professor at the University of Baltimore School of Law. Mr. Hale is a former associate at Astrachan Gunst & Thomas, P.C. *Mr. Hale may be contacted at dhale@ameritrade.com.*

The modern commercial systems and software industry in the United States have grown up in a snake-oil[1] *salesman's paradise. The largest sector of this industry*[2] *by far is composed of standard commercial systems that are marketed to provide specified functionality (e.g. Internet web server, firewall, router, etc.). Such products are generally provided with a blanket disclaimer stating that the purchaser must evaluate the suitability of the product for use, and that the user assumes all liability for product behavior. In general, users cannot evaluate and cannot be expected to evaluate the security claims of a product.*[3] *The ability to analyze security claims is important*

1. Because snakes do not exude oil, the term snake-oil has come to mean any preparation that has no real medicinal value and yet is fraudulently sold by traveling medicine shows as a cure for many ills.

2. Other sectors have radically different liability environments; vendors of life and mission critical systems and software (medical devices, aircraft controls, etc.) are well aware of their responsibilities and liabilities. Vendors/integrators of large custom systems, developed to specific specifications created by knowledgeable specialists (the development contract would clearly assign liability between the parties), also know their responsibilities and liability.

3. In the vast majority of cases, the vendor's processes and behavior have made it impossible for anyone to evaluate the product's security. Indeed, it is generally true that the vendors lack of appropriate development processes and controls have made it impossible for the vendors themselves to have a good understanding of the true security characteristics of their products.

In 1997, the United States Government formed the National Information Assurance Partnership (NIAP), *available at* http://niap.nist.gov, out of concern for the growing national security vulnerability resulting from the industry's poor practices. NIAP promotes "the development of technically sound security requirements for IT products and systems and appropriate metrics for evaluating those products and systems. The long-term goal of the NIAP is to help increase the level of trust consumers have in their information systems and networks through the use of cost-effective security testing, evaluation and assessment." *See* http://niap.nist.gov/niap/index.html. The National Security Agency, through the Information Assurance Technical Forum (IATF), *available at* http://www.iatf.net, is now trying to provide a framework to assess critical infrastructure components (critical infrastructure components are functional security blocks, such as firewalls, enterprise authentication and authorization servers, guards, intrusion prevention systems, etc. They may involve stand-alone blocks, software modules that are loaded onto protected systems, etc.) and is planning on providing guidelines on how to use these components. While NIAP/IATF have little impact in the

because a consumer may place unwarranted trust in the security abilities of a web server (or other computer device) to perform its stated purpose, thereby putting his own organization at risk, as well as third parties (consumers, business partners, etc.). All but the largest and most capable organizations lack the resources or expertise to evaluate the security claims of a product. More importantly, no reasonable and knowledgeable person would expect them to be able to do so. The normal legal presumptions of approximate equality of bargaining power and comparable sophistication in evaluating benefits and risks are grievously unjust in the context of software security.[4] *In these transactions, it is far wiser to view the general purchaser, even if that purchaser is a sizable corporation, as an ignorant consumer.*

Hence, often purchasers accept what appear to be either implied merchantability claims of the vendor or claims of salespersons' made outside of the context of a written document. These claims frequently have little, if any, basis in fact. These standard commercial systems form the bulk of the critical infrastructure of existing Internet functionality and e-commerce systems. Often, these systems are not trustworthy, yet the use of these systems by misinformed purchasers created massive vulnerability for both purchasers and third parties (including a substantial fraction of both U.S. and international citizens). The frequent disclosure of individual credit card information from supposedly secure commercial systems illustrates an aspect of this vulnerability and raises serious questions concerning the merchantability of these systems. While it is impossible to avoid all risks, they can be reduced to a very small fraction of their current level. Vendors have willfully taken approaches and used processes that do not allow assurance of appropriate security properties, while simultaneously and recklessly misrepresenting the security properties of their products to their customers.

commercial industry at this time, their evaluations and guidelines are likely to provide a reasonable basis for the requests of users and the efforts of integrators in the future.

4. 3 MARY JANE FORAN, WILLISTON ON SALES § 20-36 (5th ed. 1996) [hereinafter WILLISTON].

INTRODUCTION

The authors provide an overview of important issues concerning security-relevant functionality, including critical functionality for security product merchantability. The authors also review certain critical issues concerning vendor development and management processes that are necessary for a vendor to have any understanding of the security properties of his product. Unless a vendor takes appropriate actions concerning these and related issues, the vendor has little, if any, basis for claiming secure products. Thus, there is a high likelihood that representations concerning security and the basic merchantability of security-dependent functionality are misleading, at best.

Two analogies may illuminate the security concerns. First, implementations of electronic signatures are also vulnerable to security concerns and the true trustworthiness of the implementation may be far lower than generally believed. Electronic signatures require "digesting" of the transaction/message/authorization to be signed followed by "signing" with a private key. In most implementations it appears possible for an attacker to substitute or compromise an authorization prior to signing. In many cases, the signing key itself can be compromised, allowing wholesale electronic forgery. Second, systems with inadequate security properties are like tobacco products. Tobacco products cause damage primarily to its users and those in the smoker's vicinity and secondarily, to all who are dependent upon the tobacco user's health. Systems with inadequate security properties have even greater damage potential than tobacco because a broader scope of society may be affected by inadequate e-commerce security. Innocent parties, be they the end user, an organization, or a business, are placed at a substantial risk of fraud by deficient products sold by willfully ignorant producers. Attacks against organizational databases all too often compromise numerous credit card numbers and information, while attacks against user PCs can compromise credit card information as well as user names and passwords used by on-line systems for banking, stock transactions, and health information.

Part I of this article provides a legal review of risk management, and associated policy issues. Following this introduction, Part II discusses the existing legal structures. Part III discusses the technology and associated security issues (to provide the reader with a reasonable understanding of the technical and commercial context) and a short discussion of security issues concerning e-commerce servers. Finally, Part IV provides a) a general legal approach, b) a possible role for the U.S. Government, c) potential solutions, and d) the conclusion.

I. HISTORICAL ALLOCATION OF RISK BY THE LAW

Risk allocation is a standard function of business contracts and even the most straightforward contracts contain risk allocation. For example, a simple contract for the purchase of a single good or service includes an inherent allocation of risk. First, the purchaser assumes that the risk that the good is defective or the good will not fulfill the purchaser's intended need. Second, the seller assumes the risk that the purchaser will breach.

Like all contracts, contracts for computer security systems, expressed or implied, are governed by the law of contracts as applicable in each state. In some cases, the Uniform Commercial Code (U.C.C.) may create terms of the contract, such as warranties of fitness for a particular use. An initial step in any contracting arrangement is to determine whether an enforceable contract exists, and if so, what is the applicable law. Even though the parties have an expressed understanding, a contract may not be enforceable due, for example, to the application of the statute of frauds or statute of limitations.

II. CURRENT LEGAL STRUCTURES

Two major factors influence the law that will eventually be applied to disputes regarding computer-related contracts. Not only do traditional concerns of jurisdiction have an effect, but the choice of which of the several applicable laws to apply also plays a significant role.

Jurisdiction governs the location where a suit may be brought and this determination of which jurisdiction's laws apply may be complicated. The jurisdictional question becomes even more complicated when one or more parties are international. Due to jurisdiction's complex nature, it will only be briefly discussed.[5]

5. Many recent articles on jurisdiction focus on the Internet, but many of the concepts discussed in the context of the Internet apply generally to information technology transactions. This is because IT contracts are negotiated and agreed upon remotely, with parties potentially in separate states or even countries never actually meeting face to face. Articles discussing jurisdiction related to electronic transactions nearly universally acknowledge the difficulty in determining controlling law. See, e.g. Michael A. Geist, *Is There a There There? Toward Greater Certainty for Internet Jurisdiction*, 16 BERKELEY TECH. L.J. 1345 (2001) (describing different tests courts have used to determine jurisdiction in electronic commerce cases); Warren B. Chik, U.S. Jurisdictional Rules of Adjudication Over Business Conducted via the Internet—Guidelines and a Checklist for the E-Commerce Merchant, 10 TUL. J. INT'L & COMP. L. 243 (2002) (describing methods of obtaining jurisdiction in the e-commerce context); Paul Schiff Berman, *The Globalization of Jurisdiction*, 151 U. PA. L. REV. 311 (2002) (discussing the theoretical underpinnings for legal jurisdiction).

As set forth below, factors for determining jurisdiction include the residence of the parties, the predominance of "goods" versus "services" provided under the contract, the existence of express or implied warranties, the jurisdiction in which an action might be brought, and which jurisdiction's laws apply.

States are contemplating, and have passed, laws that prohibit forum shopping for issues involving electronic transactions. These laws, characterized by proponents as "bomb-shelters" will limit a party's ability to contractually choose a jurisdiction.[6] For example, a California-based software seller may no longer be able to haul a Connecticut customer into a California court to litigate software issues, despite contractual language to that effect. While generally aimed at preventing the application of the Uniform Computer Information Transactions Act (U.C.I.T.A.), these provisions limit the ability of a nationwide entity to consolidate product disputes in one court to increase efficiency.

The choice between applying either federal or state law (and if applying state law, which state law) must also be considered. Further, state courts can and do interpret federal law in varying ways. Thus, federal legislation can have different meanings and effects in different states.

The laws of the various states vary widely in even the basic framework through which computer-related contracts are evaluated. For example, different courts and legislatures have examined computer services and computer software contracts under common law,[7] under Article 2 of the Uniform Commercial Code (U.C.C.),[8] and, most

6. It appears that as of the date of this writing, only West Virginia has enacted the "bomb-shelter" provision into law, although it is also being considered by other states. W. Va. Code Ann. § 55-8-15 (Michie Supp. 2002). Iowa considered and ultimately defeated the provision. Uniform Electronic Transactions Act, 2000 Iowa Legis. Serv. Ch. 1189 (West).

7. *See, e.g.* RRX Indus., Inc. v. Lab-Con, Inc., 772 F.2d 543, 546 (9th Cir. 1985) (noting that whether a software transaction was to be treated as a U.C.C. controlled goods contract or common law controlled services contract is determined on a case by case basis under California law).

8. The following cites analyze and apply Article 2 to computer transactions that involve laws governing the sale of goods. Douglas E. Phillips, *When Software Fails: Emerging Standards of Vendor Liability Under the Uniform Commercial Code*, 50 BUS. LAW. 151, 157 (1994) (describing the emergent use of the U.C.C. in software transactions); *see also* Advent Sys., Ltd. v. Unisys Corp., 925 F.2d 670 (3d Cir. 1991) (applying the U.C.C. under Pennsylvania law); Rocky Mountain Micro Sys., Inc. v. Public Safety Sys., Inc., 989 F. Supp. 1352 (D.Colo. 1998) (applying the U.C.C. under Colorado law); Architechtronics, Inc. v. Control Sys., Inc., 935 F. Supp. 425 (S.D.N.Y. 1996) (applying the U.C.C. under New York law); Chatlos Sys., Inc. v. Nat'l Cash Register Corp., 479 F. Supp. 738 (D.N.J. 1979) (applying the U.C.C. under New Jersey law); Neilson Bus. Equip. Ctr., Inc. v. Monteleone, 524 A.2d 1172 (Del. 1987) (applying the U.C.C. under Delaware law); Richard Haney Ford, Inc. v. Ford Dealer Computer Serv., 461 S.E.2d 282 (Ga. Ct. App. 1995) (applying the U.C.C. under Georgia law); Sys. Design and Mgmt. Info., Inc. v. Kansas City Post Office Employees Credit Union, 788 P.2d 878 (Kan. Ct. App. 1990) (applying the U.C.C. under Kansas law); USM

recently, under the ostensibly[9] *sui generis* framework set forth in the U.C.I.T.A.[10] Additional laws may also apply. For instance, the Magnussen-Moss Act[11] governs aspects of both warranties in consumer transactions and, in the case of international transactions, international multilateral or bilateral trade agreements.[12] It imposes limitations on the form and content of written warranties for consumer goods.[13] Magnussen-Moss requires that a seller, depending on the protections of the warranty,[14] designate its warranty as "full" or "limited."[15] In particular, a full warranty must require the seller to repair or replace a defective product without charge,[16] and any disclaimer of consequential damages must be conspicuous.[17] It appears, however, that Magnussen-Moss may only cover the medium upon which software is delivered and not the software itself.[18] Nonetheless, the same reasoning that allowed courts to interpret software contracts under the U.C.C. may allow the applications of Magnussen-Moss to software in the

---

Corp. v. Arthur D. Little Sys., Inc., 546 N.E.2d 888 (Mass. App. Ct. 1989) (applying the U.C.C. under Massachusetts law); Delorise Brown, M.D., Inc. v. Allio, 620 N.E.2d 1020 (Ohio Ct. App. 1993) (applying the U.C.C. under Ohio law); M.A. Mortenson Co., Inc. v. Timberline Software Corp., 998 P.2d 305 (Wash. 2000) (applying the U.C.C. under Washington law); Micro-Managers, Inc. v. Gregory, 434 N.W.2d 97 (Wis. Ct. App. 1988). *But see RRX Indus., supra* note 7 (noting that software may be considered good under the U.C.C. or services controlled by the common law).

9. *See* Leo L. Clarke, *Performance Risk, Form Contracts and UCITA*, 7 MICH. TELECOMM. & TECH. L. REV. 1, 4 (2001). Professor Clarke takes issue with the drafters of U.C.I.T.A. on this point; Clarke criticizes U.C.I.T.A. for not taking a significant step away from the structures and for its assumptions of common law and U.C.C. predecessor statutes.

10. MD. CODE ANN., COM. LAW. II § 22-102(a)(11) & 103(a) (2002); *See* Va. Code Ann. § 59.1-501.3 (Michie 2001).

11. 15 U.S.C. §§ 2301–2312 (1994).

12. International transactions pose an emerging problem in computer transactions both in and out of the security fields. High bandwidth global computer networks allow for the world-wide efficient distribution of software. For example, a software developer in Jakarta could easily sell a "turn-key" credit card processing software package to a company in Iowa with only electronic means. The supplier in Jakarta can take the order through the Internet, process the Iowa firm's credit card payment through international credit card networks, and "ship" the processing software to the Iowa company via the Internet—all without either party sending a single piece of paper or even a CD containing the program.

13. 15 U.S.C. § 2301 (1994); *see also* Essex Ins. Co. v. Blount, Inc., 72 F.Supp.2d 722 (E.D.Tex. 1999) (refusing to apply Magnussen-Moss to equipment designed for commercial use).

14. *See id.* at § 2304.

15. 15 U.S.C. § 2303 (1994).

16. *Id.* at § 2304(a)(1).

17. *Id.* at § 2304(a)(3).

18. *Id.* at § 2301(1) (defining the subject matter of the statute as "tangible personal property"). The authors know of no cases that comment on whether software is considered a tangible item.

appropriate situation.[19] In addition to the state and federal laws, international laws and bilateral treaties may apply.[20]

Although not a key factor of this article, understanding the complicated and sometimes conflicting laws that are applied in such disputes aids in understanding the potential complexities of bringing a security-related claim.

A. *Warranties*

Under the U.C.C., which governs contracts for software in most states,[21] there are two general categories of warranties: express and implied.[22] It is critically important in transactions related to electronic security products that the responsibilities of the seller include ensuring the proper functionality of the product. The traditional framework to address such responsibilities is the warranties contained in the contract along with the remedies provided for their breach. As discussed below, these mechanisms fail in the context of electronic security products.

1. Express Warranties

Express warranties are created when the seller represents that the product has particular features or qualities and where those qualities are a basis for the purchase.[23] In practice, such warranties are usually effectively disclaimed in a blanket manner in most form contracts in order to eliminate the possibility a seller might be held responsible for the claims it makes about a product in promotional materials.[24] This is because many contracts specifically exclude any meaningful warranty and are drafted to include an integration clause that recites that the written contract is the final and complete statement of the agreement between its parties, limiting any express warranties to those found explicitly within the contract itself. That way, if a salesperson made promises that are not included in the contract, the contract shuts off any

19. *See, e.g.,* PHILLIPS, *supra* note 8.

20. *See, e.g.*, U.N. Convention of Contracts for the Sale of Goods, Final Act, April 11, 1980, U.N. Doc. A/Conf. 97/18, Annex 1 (1980).

21. *See* Phillips, *supra* note 8.

22. *See* U.C.C. § 2-313 (express warranties); U.C.C. §§ 2-314 and 315 (implied warranties) (2000).

23. *See* U.C.C. § 2-313 (2000). U.C.I.T.A.'s language related to express warranties tracks the U.C.C.'s closely. *Compare* U.C.C. § 2-313 *with* U.C.I.T.A. § 402 (2000).

24. *See* Dowty Communications, Inc. v. Novatel Computer Sys. Corp., 817 F.Supp. 581, 584 (D.Md. 1992) (citing U.C.C. § 2-316.1, Commercial Law Article, Annotated Code of Maryland, in support of the proposition that "express warranties can be negated or limited to those stated in the contract by express language to that effect").

reliance by the buyer on representations made in promotional materials or by sales representatives.[25]

2. Implied Warranties

The U.C.C. (and U.C.I.T.A.) provide three relevant implied warranties. The first, and least important in this context, is the implied warranty of title which provides that the seller has all the rights necessary to sell the product or transfer ownership.[26] Second, the U.C.C. provides for an implied warranty of merchantability.[27] This states that the product sold is suitable for the ordinary purposes for which the product is used. Third, where the seller has reason to know of a particular use the buyer intends for the product, and the seller is aware the buyer is depending on the seller's knowledge in selecting the product to fulfill the buyer's need, the implied warranty for fitness for a particular purpose arises.[28] U.C.I.T.A. adds, as a subset of this warranty, an implied warranty that all the components of a system shall function together as an integrated system.[29]

Implied warranties may be disclaimed in contracts by the seller, at least in non-consumer transactions.[30] Most contracts, especially form contracts, are artfully drafted by sellers and contain disclaimers. Under the U.C.C., disclaimers of a warranty of merchantability must mention merchantability and must be conspicuous.[31] Disclaimers of fitness for particular purpose must be in writing and be conspicuous.[32] U.C.I.T.A. sets forth similar requirements.[33] Under the U.C.C. and U.C.I.T.A.,

25. *Id. See also* 3 WILLISTON, *supra* note 4, at § 20-8 at 78–79; *Compare* Grooch v. E.I. Dupont de Ne Mours & Co., 40 F.Supp. 2d 863 (W.D. Ky. 1999) (holding disclaimer in bold and all capitals on back of bottle to be effective for express and implied warranties), *and* Minnesota Forest Products, Inc. v. Ligna Machinery, Inc., 17 F.Supp. 2d 892 (D. Minn. 1998) (constructing inconsistencies between express representations and attempted disclaimer of express warranties in favor of purchaser in absence of integration clause).

26. U.C.C. § 2-312 (2000); U.C.I.T.A. § 401 (2000).

27. *See* U.C.C. § 2-314 (2000); U.C.I.T.A. § 404 (2000).

28. *See* U.C.C. § 2-315 (2000); U.C.I.T.A. § 405 (2000).

29. U.C.I.T.A. § 405(c) (2000).

30. *See* U.C.C. § 2-316(2) (2000) (allowing disclaimer of warranties); U.C.I.T.A. § 406(b) (2000); *but see* MD STAT. ANN. COM. LAW I 2-316.1 (prohibiting disclaimer in consumer transactions).

31. *See* U.C.C. § 2-316(2) (2000).

32. *See id.*

33. U.C.I.T.A. § 406(b)(1). U.C.I.T.A.'s sections differ slightly in that they refer to existence in "a record," but these differences stem only from U.C.I.T.A.'s construct of a writing (in U.C.I.T.A., generally referred to as a record) such that it includes writings that are only embodied electronically.

however, in at least one jurisdiction, such disclaimers are ineffective in the case of a consumer transaction.[34]

A seller will almost always attempt to disclaim implied warranties when entering a contract. But, such disclaimers are not the only mechanism by which a seller may avoid the financial consequences of the product's failure. A seller may also limit the buyer's remedies.

B. *Remedies*

Even where the parties recognize a warranty, they may agree to limit remedies if the warranty is breached.[35] The contract may also limit the amount of damages available and/or limit the availability of consequential damages, such as the loss caused by the destruction of a business.[36]

Thus, a contract may provide that the only remedy available to a buyer in the event the product fails is the repair or replacement of the defective good.[37] This provision will likely be binding (at least in the absence of negligence[38]) unless, and only unless, the remedy made available "fails in its essential purpose."[39] This failure of essential purpose requires more than simply the recognition that another remedy would be better suited to make the injured party whole or that the injured party would prefer a different remedy.[40] Rather, the injured party must have *no* remedy, or the remedy must be so insignificant as to be essentially worthless.[41] For example, a repair and replace remedy would fail if the defective product could not in fact be repaired or replaced in a manner that would eliminate the defect and retain the essential function of the product.

If the essential function is not preserved, the remedies otherwise available under the U.C.C. or U.C.I.T.A. apply.[42] These include offsetting damages against the price, actual damages cased by the breach,

34. *See* MD State Ann., Comm; § 2-316.1, which provides, "The provisions of § 2-316 (allowing exclusion or modification of warranties) do not apply to sales of consumer goods, as defined by § 9-109, services, or both." *See* Maryland Uniform Computer Information Transactions Act (M.U.C.I.T.A.) § 406(i)(1) for a parallel provision.

35. U.C.C. § 2-719 (2000); U.C.I.T.A. § 804 (2000).

36. U.C.C. § 2-719 (2000); U.C.I.T.A. § 804 (2000).

37. 3 WILLISTON, *supra* note 4, at § 20-33 at 291–2.

38. Williston contends that in the event of negligence, all damages limitations are thrown out as a matter of public policy. *Id.* at § 20-36 at 307 (advising that "[a] public policy argument should be maintained that the code can never be used to limit a recovery for an injury produced by negligence").

39. *Id.* § 20-35 at 296.

40. *Id.* at 297.

41. *Id.*

42. U.C.C. § 2-719(2) and comment 1 (2000); U.C.I.T.A. § 804(b) (2000).

and incidental and consequential damages.[43] But, a careful seller will contractually limit these damages in a non-consumer transaction,[44] so long as such limitation is not unconscionable.[45]

A damage limitation, however, is unlikely to be considered unconscionable in a commercial, non-consumer context, because courts treat the two parties as though the two commercial contracting parties have roughly equal bargaining powers and levels of sophistication.[46] This assumption,[47] is in fact misplaced because the buyer does not and cannot have access to the information required. For instance, the buyer does not have the information necessary to assess the risk of consequential damages that may arise from a security product's failure.[48]

Courts, then, should be required to examine and evaluate the relative strength of the customer's bargaining power both on the basis of its sophistication and its reasonable and ready access to critical information withheld by the vendor.

III. HISTORICAL ENTRY OF COMPUTERS INTO BUSINESS SYSTEMS

The personal computer has evolved over the past 30 years from a hobbyist's plaything to a critical cornerstone of modern business, commerce, and everyday life. According to a recent Harris Interactive survey, two-thirds of United States adults consistently access the Internet.[49] In the same time period, systems running UNIX developed from time-sharing systems in engineering environments into engineering workstations and, then, into high performance servers. Both workstations and servers are critical components of the Internet and related business systems. The personal computer and its associated software, in combination with UNIX and its associated software, have grown to dominate the computing and the data processing industry

43. U.C.C. § 2-711 to 719 (2000).

44. 3 WILLISTON, *supra* note 4, at § 20-33 at 290; Aquascene, Inc. v. Noritsu Amer. Corp., 831 F.Supp. 602, 604 (M.D. Tenn. 1993).

45. U.C.C. § 2-719(3) (2000); U.C.I.T.A. § 804(c) (2000).

46. 3 WILLISTON, *supra* note 4, at 303, 307.

47. Courts have raised this assumption to the status of a rebuttable presumption which requires a tougher evidentiary standard. *See* 3 WILLISTON, *supra* note 4, at § 20-35; Siemens Credit Corp. v. Marvik Colour, Inc., 859 F.Supp. 686, 695 (S.D.N.Y. 1994); Am. Dredging Co. v. Plaza Petroleum, Inc., 797 F.Supp. 1335, 1339 (E.D.N.Y. 1992).

48. *See supra* Section III; *see also* Clarke, *supra* note 9, at 54.

49. *Harris Interactive Poll # 18, April 17, 2002*, *at* http://www.harrisinteractive.com/harris_poll/index.asp?PID=295 (last visited Mar. 21, 2003).

(e.g. Microsoft Windows / NT / XP families on the client and server side; and, Sun's Solaris, HP's HPUX, IBM's AIX, BSD, and Linux on the UNIX side).[50]

While more attention has been paid to system safeguards in the server environment, that level of attention, and its resulting capabilities and integrity, have not matched the needs of such a critical component of the infrastructure. The Internet is a descendent of the ARPANET, a research effort of DARPA (Defense Advanced Research Projects Agency, an agency of the U.S. Government Department of Defense in the 1970's and 1980's) and the academic community.[51] The original Internet protocols[52] were not designed for security or integrity; they were designed for simplicity and robustness in the event of random communication node failure. The protocol's ability in the face of deliberate electronic attack to preserve its security and integrity, let alone robustness, is deeply flawed.[53] The basic IP protocols were designed to enable data packets to be efficiently routed from sender to receiver in the face of unreliable communications networks, but they were not designed to deal with security compromises. They do not, in general, provide any defense against active address manipulation/forgery, do not detect malicious servers and routers, and do not provide any level of integrity assurance. The implementation of the IP naming and lookup system is too vulnerable to directed attacks against the top nodes.[54]

Building trustworthy business processes upon the foundations of untrustworthy PC-based clients, vulnerable servers, and a weak Internet infrastructure is dauntingly difficult and requires great care and knowledge. While it is possible to provide reasonable security, all too often the commercial/state-of-the-art (as opposed to academic) level of computer equipment and software implementation leaves the user quite

50. See Gartner Dataquest or the U.S. Business Reporter http://www.activemedia-guide.com/pc_mrkt.htm for the exact breakout of the ever-shifting market share wars. Essentially, Compaq™, Dell™, Gateway™, Hewlett Packard™, and IBM™ dominate personal computer sales. IBM™, Hewlett Packard™, Sun Microsystems™, Compaq™, and Siemens™ dominate the midrange server market while IBM™ dominates the mainframe space. Microsoft™ desktop operating systems (Windows, etc.) and software such as Microsoft Office™ retain 96% of the office suite software market.

51. Barry M. Leiner, et al., *A Brief History of the Internet and Related Networks, at* http://www.isoc.org/internet/history/cerf.shtml (last updated Nov. 18, 2001).

52. The Internet Protocols are a set of rules or languages that govern how computers communicate. They were designed to automatically work around damaged communications links or failed systems so that the networks would be robust when faced with extensive physical damage. They were not designed to be robust against directed corruption of the platforms that run the Internet protocols.

53. Peter G. Neumann, *Practical Architectures for Survivable Systems and Networks*, 63–66 (2000), *at* http://www.csl.sri.com/~neumann/arl-one.pdf (last visited Mar. 21, 2003).

54. *See id.*; *See also* Bruce Schneier, Secrets and Lies, Digital Security in a Networked World 186 (2000).

vulnerable to theft, corruption, misuse, or destruction of confidential information.[55] Indeed, many alleged security measures actually provide no resistance whatsoever to a knowledgeable attacker. As an example, if a hacker penetrates a system and inserts a Trojan horse (a program that purports to have one function but has hidden back doors or compromise-enabling functionality) to "take over" a given system, the hacker can utilize the systems' encryption functionality (supposedly a strong security protection) and other security measures such as internally held passwords to "authenticate" itself to other parts of the internal or external system, thus helping to corrupt other systems or transactions.

In truth, such security measures serve only to provide a mistaken belief in the security of the systems. Vendors, who do or should know better, are careful to enforce the user's illusion because it increases the sale of their products and services.[56] The typical hype concerning "secure servers" or "secure browsers with 128-bit encryption" is familiar. The user, usually lacking the knowledge to assess their vulnerability to attack, typically accepts the assurances of the vendor or integrator. The vendors usually protect themselves from liability behind blanket disclaimers, which typically make the user assume all liability for failures, even though the user does not have the necessary expertise or information to evaluate the vendor's claims. System failures frequently result in losses by third parties, such as banks or credit card holders whose accounts were compromised.

The vendors of many e-commerce systems are the electronic equivalent to builders of safe deposit vaults whose back wall (behind the deposit boxes) is foil-coated drywall (rather than the requisite concrete and steel), thereby easily permitting a knowledgeable thief (with very modest and available tools) to enter and empty all the safe deposit boxes. These products (including consumer e-wallets, or even more so, merchant e-commerce systems that capture and maintain consumer financial information such as credit card numbers in the same system that interfaces to the user) do not provide security against rudimentary, let alone reasonably sophisticated, attacks. While both parties are unaware of their respective vulnerabilities, the purchaser of the vault has

55. SCHNEIER, *supra* note 54, at 156 (discussing Trojan horses).

56. Vendors have little incentive to conduct reasonable, let alone comprehensive, security reviews of their products. They want to believe that all commercial products are equivalently vulnerable and that the smoke and mirrors of their marketing staff's are reality. As long as they face no risks or costs from such behavior, they have no incentive to change their deceptive ways. *See* SCHNEIER, *supra* note 54, at 335–338.

not assumed responsibility, whereas the e-commerce user has inequitably assumed liability via product disclaimers.

It is the position of the authors that such products a) should not be viewed as merchantable because of the security deficiencies, and b) vendors are willfully and recklessly committing a tacit fraud upon their purchasers. The manufacturers of the "vaults" should be liable for both their consumers' losses and the losses of third parties who place mistaken credence in the assurances of product safety.

A. *Software Development*

The complexity of software grows with the volume of *active* code and the number of functions in the program.[57] Initially, programs that contained hundreds of lines of active code and a small number of functions were considered large. The general classification of "large" has since changed to include programs containing hundreds of thousands of lines, thousands of internal functions, and is now further changing to include programs containing millions of lines of code and even more functions.[58] The software development community continually demonstrates the immaturity of software through the rapidity with which it adopts, changes, and abandons tools, processes, and languages, and its response to the marketing and spin of self-interested vendors and true believers in search of the "magic bullet."

Large-scale software (even if defined as only containing a few tens of thousands of lines of active code) has never been fault-free, and it is unlikely that such a desirable situation will ever occur. Industry

57. Counting the sheer numbers of lines of code was the first (and probably the weakest) method of determining the complexity of software. Though it is intuitively reasonable to claim the more lines of code, the more complex the software, many of the lines are generally "no op" lines which represent information/documentation about the code functionality, but perform no operation. A modern system for establishing the complexity of software, and hence the probable cost, was established by Capers Jones and is called function or feature point analysis. This method is dependent upon identifying five elements in the code: input, outputs, inquiries, internal stores of data, and external references to data. *See* The David Consulting Group, *Estimating Software Earlier and More Accurately*, *at* http://www.davidconsultinggroup.com/articles/pbestart.htm (last visited Mar. 29, 2003) for an in-depth discussion of the system.

58. Software program size has grown in direct proportion to a) the availability of hardware direct and virtual memory, and b) the increasing number of functions the software is supposed to perform. In addition to the five basic elements identified, *supra* text accompanying note 57, function/feature points now include: data communications, distributed data processing, performance, heavily used configurations, transfer rates, on-line data entry, end-user efficiency, on-line updates, complex processing, reusability, installation ease, operational ease, multiple sites, and technologies that facilitate change. By 1998, major U.S. Department of Defense software systems contained about 300,000 function/feature points which was roughly equivalent to 27 million lines of code. The code growth phenomenon has continued. *See* Caper Jones, *Sizing Up Software*, SCIENTIFIC AMERICAN, Dec. 1998, at 104.

representatives and professionals believe that a) there is no known way of writing large scale software without defects,[59] b) logical operation of most software systems are not adequately documented,[60] and c) it is unreasonable to expect companies to build and maintain documented, fault-free systems and applications.[61] *They are indeed correct!* Unfortunately, we have to live with software defects in operating systems and applications that we use. While the existence of software defects is an abysmal fact of life, the number and severity of software defects is something that the software manufacturers can substantially control, were they motivated to do so. If software for life safety systems (e.g. those that control aircraft, automotive braking systems, hospital monitoring systems, etc.) could not be produced with few or no software defects, then perhaps the standard commercial vendors could be excused from an evidentially unattainable goal. However, the mission critical software developers do, indeed, attain the few or no defects goal.[62] Designing software to minimize software defects is a known, but all too infrequently used, process. Similarly, since the beginning of the PC days, software vendors knew that coding

59. Many of these systems are so large (exceeding several hundred million lines of code) that no single individual understands either the whole system or the relationship of a given part to the whole system. Since the bulk of these systems interface to a large number of other equally complex systems, often in complicated, tightly coupled ways, comprehensive testing of the system is neither feasible nor undertaken.

60. This problem is as common as dirt for several reasons: a) programmers want to write code, not document code, b) if the code isn't documented, it provides a certain level of "job security" since the original programmer will likely be called upon to fix any error or add any functionality to the software, and c) if the development of the code is suffering from normal schedule compression, then management will forgo code documentation and, instead, have the programmers continue to just write functional code. *See* Software Program Managers Network, *SPMN Software Development Bulletin #3*, Dec. 31, 1998, at 7, *at* http://www.spmn.com/lessons.html (last visited Mar. 29, 2003) [hereinafter SPMN].

61. Building and maintaining large-scale systems is often quite counterintuitive. *See* Chris Verhoef, The Realities of Large Software Portfolios (Feb. 24, 2000) (unpublished manuscript, on file with MTTLR), *available at* http://www.cs.vu.nl/~x/lsp/lsp.html. One measure of maintainability is the so-called ripple effect—defined as the number of separate code areas that have to be changed to effectuate a single modification to the original code. One study found that each modification leads to about 300 other modifications. *See* G.M. WEINBERG, QUALITY SOFTWARE MANAGEMENT: VOLUME 1 SYSTEMS THINKING (Dorset House, 1992). This leads to a combinatorial explosion, *see infra* note 89.

62. 90% of the software errors can be found by inspections before the first test case is run. Inspections, if done correctly, are hard work, requiring demanding attention to software structure, each individual detail in the software itself, and a significant amount of time. It is a hard task to find the people who can and will do the inspections. Unfortunately, most companies don't do many inspections, and some do none at all. Robert L. Glass, *Inspections—Some Surprising Findings*, COMMUNICATIONS OF THE ACM, April 1999, at 17.

guidelines minimize the impact of software defects.[63] Unfortunately, these guidelines are rarely followed.

The classic model of the Software Development Life Cycle (SDLC) illustrates the steps to create functionally viable software. First, system requirements are obtained from the customer. Second, a functional baseline/system design is created. Third, software requirements are developed. Fourth, a preliminary software design is built. Fifth, a detailed software design is generated. Sixth, the actual coding, divided into software modules (or units), is completed. At this point, individual units are usually tested for stand-alone functionality. Seventh, the individual units are then integrated. Since different software developers typically utilize their preferred strategies in developing an individual software unit, the disparate units can prove non-functional when mated with the rest of the overall program, requiring redevelopment of a given unit.[64] Eighth, once the entire program is operational, it is then tested as a system for functionality. Ninth, the customer begins to use the software. Finally, as the customer utilizes the software, defects are identified and corrective actions are taken to eliminate the defects and/or improve the functionality of the code.

This model, typically called the "waterfall" model, requires a thorough understanding of the product requirements by both the customer and the system developer. Unfortunately, commercial software vendors frequently do not understand the customer's business model and requirements,[65] and customers may not know what their business model and requirements will be once they implement new technology.[66] In many cases, customers cannot even adequately explain how they currently do business. Computer system analysts, upon investigating and documenting the customer's business processes, are often faced with a customer who sincerely disbelieves what has been found. The sheer complexity of some modern business processes effectively defies

63. *See* Jerome H. Saltzer and Michael D. Schroeder, *The Protection of Information in Computer Systems*, PROCEEDINGS OF THE IEEE 1278, 1282 (1975) (discussing the importance of implementing guidelines in the development of computer security systems).

64. A Survey of Major Software Design Methodologies, *at* http://userpages.umbc.edu/~khoo/survey2.html [hereinafter Methodologies Survey].

65. Vendors commonly send in both marketing and technical teams to ascertain a given business' needs in order to match those needs to the product and/or services that the vendor sells. Modern business models are sufficiently complex that the vendor will offer (as a set of credentials) a history of where they have successfully sold products and services in a given industry, implying that that all of the trial and error in supporting a given business model has now been eliminated.

66. Any change, be it managerial or technological, ordinarily manifests unintended consequences. Fixing a given piece in the "supply chain" can cause numerous problems elsewhere, much like being a teenager and putting a really impressive engine in the hot rod, only to discover that the rest of the drive train must now be upgraded.

analysis. By the time a set of processes is analyzed, requirements generated, and the desired software developed, utilizing the SDLC waterfall model, the business (and hence the software) requirements have drastically changed.[67] In such an environment, a rapid prototyping process is frequently used, and high-level tools are used to assemble basic system functionality so that the customer can interact with it and determine what their requirements and criteria will be. In theory, once these requirements and criteria are determined, the development process should return to the beginning and do a full development, aided by a better understanding of the business process.[68]

Since the explosive growth of the Internet, pressure from shareholders and management to produce products on "Internet Time" have increased to an incredible level.[69] Although the bubble has burst relative to the "dot.com" Internet explosion, the time to market pressures have only marginally relaxed. Frequently, the up-front time (meaning SDLC steps one through five) to design the system structure before the developers start coding has been massively truncated.[70] Rapid prototyping tools are used to assemble a product prototype, which is repeatedly modified until the customer and/or the sales and marketing department approve. The prototype is then turned over to the development teams for fleshing out, cleaning up, and optimization. Indeed, one school of developers employ "Extreme Development", the rapid spiral development where incremental modifications and extensions are made to the project code base as new requirements are added, old requirements are modified, and earlier errors are found and fixed.[71] It thus jumbles the SDLC steps to decrease production time. It is Dr. Mohan's opinion that there is no reason to believe that the success of Extreme Development practices, in dealing with very large or mission critical development

67. *See* Diane Wilson, Thyra Rauch, & Joeann Paige, *Prototyping in the Software Development Cycle, at* http://www.firelily.com/opinions/cycle.html (last visited Mar. 20, 2003).

68. *Id.*

69. Everybody believed that if they were not first in the Internet arena, then they would lose. This reasoning permitted many individuals to convince investors, stockholders, bankers, managers, programmers, marketers, etc. that they better take the plunge now—later would be too late. This resulted in the "dot.com bubble" and all of its sad outcomes (*see Venture Capital Goes Back to the Basics, at* http://news.com.com/2009-1017-887703.html?tag=cd_mh (last visited Mar. 20, 2003).

70. Worse, the time is not only truncated, the required planning simply isn't done. *See* SPMN, *supra* note 60, at 9.

71. The idea of extreme programming was born out of a) the need to quickly communicate ideas and approaches in writing code for a given software project, and b) to see that the (small) teams were continuously productive at a sustainable pace. *See* Ron Jeffries, *What is Extreme Programming?* (Nov. 8, 2001), *at* http://www.Xprogramming.com/xpmag/whatisxp.htm (describing in depth the values and virtues of this design methodology) (last visited Mar. 16, 2003).

projects, will equal that of earlier practices. This is especially true for projects that have real security or survivability requirements (Dr. Michener strongly concurs on this statement).[72]

Unfortunately, neither the SDLC waterfall model nor the rapid prototyping model will ever functionally succeed if development time is cut by more than 25% of a reasonable schedule (reasonable being based upon the documented history of how much time it takes to typically build an equivalent program).[73] Further, even when the schedules are uncompressed, the success rate (defined as slightly less than 200% of budget and 200% of schedule) of large software projects has been dismal, typically around 9%.[74] The other 91% of the projects are either abandoned outright or renamed and started over.[75]

Ignoring this data, development schedules have been compressed well beyond the 25% point[76], design and code reviews have been reduced or eliminated[77] (effectively eliminating the opportunity to discover and correct both design and coding errors), testers have been turned into developers[78] (thus reducing further the possibility of finding and correcting the more subtle design and coding failures), and test time has been minimized in the rush to market.[79] Obviously then, large numbers of software defects and design oversights are shipped to customers. Certain software organizations are famous for the shipment of functionally defective (bug laden) code.[80] The software organization euphemistically refers to these bugs as "features" of the system.[81] This

72. Extreme programming relies upon the fact that two or three individuals are assigned to develop a set of functions or features of the code. Though this might provide "rapid prototyping" of a system, it is difficult to ascertain exactly who is responsible for which function. Further pressure to clarify the situation or ascertain which coder is responsible typically only exacerbates the problem.

73. BARRY W. BOEHM, SOFTWARE ENGINEERING ECONOMICS 467 (1981).

74. J. C. Whitmarsh, *Letter From The Editor*, CIO, Nov. 15, 1995, at 10.

75. *Id.*

76. There is a remarkably consistent cube-root relationship for the most effective schedule for a single-increment, industrial grade software development project. Barry Boehm, *Industrial Software Metrics Top 10 List*, Number 2, IEEE SOFTWARE, Sept. 1987, at 84.

77. Glass, *supra* note 62, at 17.

78. Often such actions are referred to as a software "death march" – large amounts of time and available bodies are spent on generating functional code. Meanwhile, no effort is put towards testing. *See* SPMN, *supra* note 60, at 20.

79. *See id.*

80. Windows 1.0 was considered "buggy", crude and slow. *See* Marry Bellis, *Part 2: Getting the Bugs Out of Microsoft Windows, at* http://inventors.about.com/library/weekly/aa080499a.htm (last visited Mar. 16, 2003). Other operating system vendors argue against Microsoft™ because of the Windows' historical and current lack of reliability. *See also* John Kirch, *Microsoft Windows NT Server 4.0 versus UNIX, at* http://kirch.net/unix-nt/ (last updated Aug. 7, 1999).

81. Complex systems of dynamically interacting components often behave in ways that their designers did not intend. These systems will display emergent behavior—behaviors not

is done to blunt criticism by implying that the particular defect was deliberately designed into the system. Ironically, the software organization is actually correct! By not making the effort to clean up the code, the original design (or lack thereof) and its defects, are retained.[82] Though the users are often painfully aware of this fact, there is little that they can do, other than to report the functional defects as they are found.

This process also directly applies to security defects. By not appropriately considering security issues in the architecture and design, the ensuing security oversights and weaknesses are designed in.[83] Customers suffer losses, quite often large losses, due to these software defects and design oversights. The competitive nature of the industry places massive roadblocks in the path of responsible software manufacturers. Manufacturers lose if a product is shipped late, but are not held responsible for damages for shipping defectively designed and implemented products. As any experienced software developer can attest, any schedule and cost constraint can be met, where quality is sacrificed.[84]

B. *Security Must be Designed in From the Start*

Executives, managers, and software developers in the software and systems industry want to believe that security is similar to other system functionalities—a module you snap into an existing system to make it secure by adding appropriate security functions. Knowledgeable security professionals know that this is not true.[85] Experienced security

intentionally designed into the system but that emerge from unanticipated interactions among components. *See* National Research Council, Making IT Better 113 (2000), *available at* http://www.nap.edu/html/making_IT_better/ch3.html (last visited Apr. 11, 2003).

82. By not making the effort to clean up any errors, all of the original errors remain. *See* Glass, *supra* note 62, at 17.

83. The same is truer for issues of system survivability, which is a harder issue to deal with than security! Survivability is the ability of the system to continue functioning, with reduced performance, even when faced with a wide variety of failures including, but not limited to: failures of communications networks; security attacks on one, many, or all available systems; and, denial of service attacks to prevent system access, power failure, natural disasters, and physical attacks against system facilities. Some of these failures may be maliciously managed; malicious actions differ from random and statistical types of errors found in noisy environments because they are deliberate attempts to attack the weakest component of the system to compromise the entire system. Motivations for such attacks can range from revenge to sheer greed. Issues of survivability will not be further discussed here.

84. If no one cares about the quality of the software, then any insignificant effort is successful. SPMN, *supra* note 60, at 15 (pointing out that software projects rarely have qualitative, let alone quantitative, quality goals).

85. *See* SCHNEIER, *supra* note 54, at 127–130.

professionals know that their advice will likely be ignored.[86] It is impossible to add security to an existing system by adding a security module on top of it;[87] security must be appropriately considered and implemented from the beginning when system functionality is partitioned, appropriate modules and sub-systems are designed, appropriate routines are developed, and internal and the external system interfaces are specified, designed and tested.[88] Security is difficult because it is necessary to provide assurance that the security functionality is exactly what it is specified to be, and *nothing more*. The *nothing more* is the hard part.[89]

86. *See* Alan Chmura & Richard Vireday, *Managing Software Development*, 6 COMPUTER BITS 12, December 1996, *available at* http://www.computerbits.com/archive/1996/1200/softdev1.html; *and see* SPMN, *supra* note 60. Software program managers are rarely rewarded for surfacing key risks (e.g. security, safety, schedule, etc.). Not only are such risks ignored, no funding or schedule reserves are set aside to deal with the risks when (not if) they surface. Instead, the software products are pushed out the door with the motto that "any problems found will be fixed during the next release" (which is obviously someone else's job). Unfortunately, security fixes cannot be added in afterward. *See supra* note 85.

87. A cryptographic module can be added to a system, but this does not add security. It does not deal with a category of attacks that take the data once it has been decrypted, compromise the key, compromise the user password, compromise the cryptographic module itself, insert unauthorized users, etc. *See* NEUMANN, *supra* note 53, at 72–74; *see also* Paul H. Merrill, NOT the Orange Book §§ 1–2 (1992).

88. *See* Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, (December 1985). In Part 1, Critieria, colloquially referred to as the "Orange Book," the specifications of the various classes require that this functionality be engineered in. Governments first faced the security implications of computers and the vulnerabilities they presented. The U.S. Government funded large research programs to understand the issues and develop solutions. This governmental work has provided the theoretical basis for the development and understanding of secure systems since that time. In the 1980's, this governmental work resulted in the U.S. Government Trusted Computer System Evaluation Criteria (TCSEC) and moved in the 1990's to the Common Criteria, an internationally accepted security evaluation basis. B1 and B2 are TCSEC categories for computer systems. Unlike the "commercial grades" of C1 and C2, the B categories require the support of "Mandatory Access Control" (MAC) labels on data, memory, communications channels, devices, etc. MAC functionality is necessary to confine the damage from malicious or defective software. *See* ROSS ANDERSON, SECURITY ENGINEERING: A GUIDE TO BUILDING DEPENDABLE DISTRIBUTED SYSTEMS 527 (2001) (providing an overview of the criteria).

89. Attempting to prove that any arbitrary state of the hardware and software system will NOT result in a given form of unwanted behavior is prohibitively difficult because increasing the number of entities that can be combined creates a huge number of possible combinations. For example, arbitrarily combining three lines of code into possible coalitions results in five coalitions. Among five lines it becomes 52 coalitions, among 10 lines it is 115,975 coalitions, and among 20 lines it is 51,724,156,235,572 coalitions, ad nausem. Today's software runs to millions of lines of code that can be arbitrarily multiplied against many different states of the hardware. Simply stated, *no one* performs comprehensive testing to prove a lack of unwanted behavior—they cannot afford the time or the expense. The discussion in MERRILL, *supra* note 87, at Section 1, clearly shows functional partitioning and security engineering being required at the onset of system and procedure development. *See also* MICHAEL HOWARD & DAVID LEBLANC, WRITING SECURE CODE 20 (2002).

The development and implementation of any new technology ordinarily suffers from unintended consequences. Although it is generally easier to verify the presence of specified functionality, it is usually very difficult to provide assurance that the system does not have unspecified behavior and is intrinsically safe. As previously discussed, assurance of proper behavior in complex systems must be designed in from the beginning. Since security professionals are rarely provided this opportunity, they are typically reduced to provide security-relevant modules and patch-ups on already designed and implemented products. It is not uncommon for corporate security professionals to discover that their organization has been shipping a product with unacceptable security properties for a substantial period.[90] Developers and development managers view security experts as roadblocks to adding new features for consumers and to their ability to timely ship a product which meets their development milestones. Consequentially, it is very common for development teams and product managers to hide security relevant products and feature enhancements from the critical eyes of the corporate security expert.[91]

C. *The Security Problem and the Failure of Existing Practices*

The problems of design oversights and software defects are probably most serious for areas of software, system security, system integrity, and survivability because these areas are not implemented by functional add-in modules and are most sensitive to errors of composition, omission, and undocumented features.[92] The following examples of problems were excerpted from the Carnegie Mellon Computer Emergency Response Team [CERT] public Internet listings.[93]

---

90. *See* Chmura, *supra* note 86 (providing the Network File System (NFS) as an example of a security deficient product that is an active security hazard to any organization deploying it. Unfortunately, it is a commonly supported "standard"). *See also* ANDERSEN, *supra* note 88, at 370.

91. For the reasons noted in Chmura, supra note 86, the security guru asks very hard questions and poses development scenarios that no one (besides those truly concerned with security) wants to hear. They would just as soon not invite this "troublemaker" to any of the management reporting meetings and, if they can get away with it, never do. The technical authors personal experiences and hearsay with other security professionals strongly supports this observation. *See* Tolga Acar & John R. Michener, *Risks in Features vs. Assurance*, COMMUNICATIONS OF THE ACM, Aug. 2002, at 112, *available at* www.csl.sri.com/users/neumann/insiderisks.html#146.

92. NEUMAN, *supra* note 53, at 49–62.

93. These listings are so popular with attackers looking for new attack points that announcements of new vulnerabilities are delayed to allow the development and deployment of security patches before the public announcement of the problem. To view the CERT vulnerabilities and advisories in 2002, see http://www.cert.org (last visited Apr. 15, 2003).

- It is not recommended to install more than one copy of Windows NT on the same computer, however if you must, we recommend that the second copy of Windows NT has no users except for the local Administrator and that a strong password be set on this account. There are cases where ACLs created by one of the copies are not protected when another copy is active.[94]
- When installing Windows NT do not copy the entire root directory and a few other files from one computer to another as each NT installation receives a unique system ID which makes its accounts and group ID's also unique. Making such copies may compromise the entire network's security.
- Check for ROLLBACK.EXE on the hard disc and if present remove it. ROLLBACK can destroy critical system information including the registry, user account information. To recover from the ROLLBACK.EXE damage the entire system has to be restored from the backup tape, if one is available. Note that Microsoft inadvertently distributed ROLLBACK.EXE with some Windows NT 4.0 releases.

Inter-version change logs, which detail system improvements between version releases, demonstrate that Open Source development[95] is not inherently secure. Typically, each release contains many security bug fixes.

The unfashionable field of system security[96] has been badly handled by the bulk of the commercial community.[97] On the other hand, the Government commissioned, at great expense, processes, approaches,

94. ACLs refer to mechanisms and policies that restrict access to computer resources. An access control list (ACL), for example, specifies what operations different users can perform on specific files and directories.

95. Open-source development is the development of operating systems, applications, and modules where the "source code" for the project is visible to all. Anyone who wants to can go through the code line-by-line, module-by-module looking for errors and flaws. Despite the openness of the Linux source, Linux has long been known for its security vulnerabilities. Looking at the change logs between versions of Linux shows a continual patching of security holes.

96. The authors note that the term "assurance" is largely unknown in the commercial environment in the U.S. While a basic foundation in the government security arena, the areas of the foundations of trust have been neglected in the commercial world. You can add a new, "improved" cryptographic module, you can add a "smart card" to a system, but you cannot add a trust module to an existing commercial product. It must be designed in from the beginning.

97. *See* SCHNEIER, *supra* note 54, at 120–134, 151–175.

and technologies to provide for assurance of security properties. The commercial market has generally chosen to ignore these approaches as being justified for government applications only.[98] This willful, and perhaps reckless, ignorance was and is a great error. Some of the approaches developed by the Government are among the best means of dealing with very serious problems in the electronic commerce arena.[99] Perhaps the greatest security problem is the problem of malicious software. Standard operating systems have no defenses against malicious software. Systems with Mandatory Access Control functionality can be configured to limit the damage that can be done by malicious software and provide substantial defenses against the unauthorized introduction of malicious software.

The current development paradigm consists of composing functional blocks together until a system with the desired properties is assembled.[100] Unfortunately, as previously discussed, this approach rarely[101] works in the security arena because the systems we build and develop are layered, modularized, and packetized. Yet, the layering model might be used as a solution to the problem of software functional complexity. The model suggests dividing the software functionality into *layers*, each of which solves part of the functionality problem. This simplifies the design of the code because software for each layer depends only on the software services provided by lower layers. When software is layered, the functionality within a given unit or module of code is arranged so similar functions are grouped together, with some groups coming before (a "higher" layer) and some groups coming after (a "lower" layer). Typically, the most basic functionality resides in the lowest layer.[102]

All commercial operating systems (OS) have two layers: the operating system or kernel, and the user layer. Theoretically, there is a checks and balances system. For instance, the operating system is

98. The primary exception here appears to be Sun Microsystems's government division. It is trying to market its "Trusted Solaris 8," an evaluated product, to the commercial marketplace. Similarly, Hewlett- Packard is trying to market its "Virtual Vault," a product descended from an evaluated project, to the commercial marketplace.

99. The government's classic multi-level secure approaches have their problems and must be applied with care. Integrity management is more broadly applicable than security management in the commercial world, but both are invaluable tools. *See* ANDERSON, *supra* note 88, at 157.

100. Methodologies Survey, *supra* note 64.

101. Such composition requires a knowledgeable and experienced security architect working with functional blocks of known properties and behaviors. Functional blocks (including low integrity operating systems) with unknown behaviors CANNOT be used in the security critical portions of the system.

102. Methodologies Survey, *supra* note 64.

protected from modification by code or processes running in user space. Systems designed to be secure have three or more layers, with a minimized security process functioning below the operating system, to enforce the security policy that protects the OS and users from rogue users and processes.[103] Network protocols often possess seven or more layers, while functional application modules can consist of many more.

Hence, one software layer can be changed and the impact of the change assessed, without affecting the entire module. Software layers within a module can be tested independently and can be replaced wholesale within a software stack.[104] Here lies the fundamental security problem. In commercial operating systems, experience has shown holes that enable Trojan horses, viruses and worms to execute code that replaces the correct functionality of a given layer of software with the corrupted functionality of the inserted code.[105] As long as the interfaces between the layers above and below the inserted rogue code accept the input and output of the replacement layer, the software will continue to function, albeit in a corrupted manner, without knowing the difference. An analogy may be useful. An attacker substituting a module in a system is equivalent to the mafia replacing a government agent with his or her own man. The impostor can delete true information, insert false information, leak valuable information, etc. All the module has to do is have the same input and output interfaces, which are typically standardized and frequently published.

Unless explicit mechanisms are built into each layer of each module; the receiving layer implicitly trusts the data presented. Commercial software generally possesses no such security mechanisms to alert it to untrustworthy data.[106] For instance, systems are dependent

103. Typically this security process is called the "reference monitor." "Reference monitors" are needed in highly trusted systems and in the old Trusted Computer Security evaluations (orange book), levels B2 and higher. B1 systems, which are the lowest level system to be "trusted," do not require the implementation of a reference monitor.

104. Methodologies Survey, *supra* note 64.

105. The four major problems that modern commercial operating systems suffer from include: a) that the system does not authenticate itself to the use, b) there is improper handling of passwords, c) there is improper implementation of the operating system, and d) there is a parameter passing by reference, any of which could easily be corrupted by a modestly skilled hacker. Computer Security Flaws, *available at* http://www.cs.ucd.ie/staff/tahar/home/courses/4thyear/chapter4/img005.htm (last visited Mar. 19, 2003).

106. The only tool occasionally deployed in commercial systems is code signing. This is primarily used to provide a minimal basis of trust for mobile or downloaded code and is the security mechanism for Java applets, Microsoft's ActiveX technology, CORBA, etc. Code signing is not used much to protect the integrity of applications or operating systems, although it was integral to Novell's cryptographic infrastructure. Unfortunately, code signing does not deal with the issue of configuration management or of revocation of rights of modules that have been discovered to contain errors. Solutions for these issues are known. John R.

upon driver software to communicate with or utilize a third party device, such as a high performance disc array or a graphics system. Attackers can distribute an "improved" driver with compromised code. If attackers get an administrator to install one of these drivers, which are installed in the operating system kernel, they can control the system.[107] Such human-engineering-assisted compromises are easier to do than is generally recognized.[108]

This type of security problem is further extended when the software is modularized. Instead of moving "up" and "down" within a given module of software, the data moves "horizontally" among modules until the last module is reached. Thus, rogue data can be inserted either at the receiving interface for a module or at the transmitting interface for the next module. All of this takes place within a given software application, or at the interfaces between the application and the operating system (or system traffic cop). Since the operating system is also constructed using the layered software approach, there is nothing preventing rogue code from executing the same kind of replacement scenario, thus taking over the layer, described for the internal and external module interfaces.

Worse, the same problem can occur with data packets sent across a network. Unless session security and integrity protocols are strongly enforced[109] (something that is rarely done due to the computational

---

Michener & Tolga Acar, *Managing System and Active Content Integrity*, IEEE COMPUTER, Jul. 2000, at 108.

107. One of the technical authors attended the Defcon conference in 2000. Hackers discussed such attacks, both in presentations as well as in person.

108. Drivers are upgraded all the time, and administrators know that a routine part of their job is upgrading device drivers. Examples of such drivers are printer drivers which are necessary to provide printing, video drivers for display, network drivers for communications, and the like. Unfortunately, even legitimate drivers may have security holes that can be taken advantage of. Trojan drivers supplied by an attacker will be worse. If the administrator is not scrupulous concerning the source and integrity of all drivers, the systems are easily compromisable from this approach.

109. IP is the standard network protocol used by the Internet. The most common transmission protocol for general use on IP is Transmission Control Protocol (TCP); hence, the acronym TCP/IP. If I wish to set up a communications session between my system and another system, our two computers must negotiate the session between them. If I want to protect the session against alteration/hijacking by an attacker, I instruct my system to require integrity checksums on all packets that are exchanged. Since the attacker cannot forge the checksums, our systems can detect alteration or insertion of packets. If I want to protect the session against observation, I instruct the system to require cryptographic security, and I may instruct my system on what constitutes acceptable cryptographic algorithms. In this case, the two systems must negotiate together to establish a session key, which is used to protect the communications. Both integrity and security capabilities are supported by the IP Security standard, IPSec (which is not universally supported). If systems do not have IPSec available to them and security is still desired, cryptographic protection is added at the application level—SSL, HTTPS, etc.

burden that security and enforcement places upon the systems), packets can be deleted and new packets inserted in their place. The fundamental flaw of having wholly replaceable links in the chain is replicated throughout the entire set of end user, corporate, and Internet systems. Hence, the security of a system is no stronger than the weakest link, layer, interface, or network in the chain.[110] A competent attacker can deliberately compromise the weakest link, layer, interface, or network and use this breach to exploit the rest of the system.

Security attacks are targeted against the weakest links in a system and any and all compromises are ruthlessly exploited. This fact separates security engineering from the more common engineering for noise immunity and traffic handling,[111] which are typically dominated by statistical properties. The vulnerability of systems to such exploits is well illustrated by the problem of cryptographic protection of communications and data. Cryptographic protection prevents unauthorized users from reading or modifying protected data because the unauthorized user does not have the "key" that is used to encrypt or decrypt the data. The attacker can solve the code to determine the key, but this can be an exceptionally expensive and time consuming operation. Stealing the keys is typically far easier.[112] Indeed, one of the major uses of malicious software is the acquisition of user names, passwords, and key phrases; allowing the attacker unauthorized use of the protected system functionality. If the system does not provide adequate protection for cryptographic keys and their access / activation functionality, the cryptographic protection provides the illusion of security, not the actuality. Unfortunately, the user is likely to have an entirely mistaken belief in their level of security.

Software and systems to facilitate electronic commerce are in exceptional demand and are subject to all the pressures for speedy shipment mentioned above, and then some. Sadly, customers generally lack understanding of the issues, have unreasonable expectations, and look for miracle cures (typically some form of add-on module that will secure their existing product without affecting existing product

110. SCHNEIER, *supra* note 54, at xii.

111. A deliberate and targeted traffic loading of a particular site is a Denial of Service attack. Such an attack is not a security attack. Such attacks demonstrate that systems designed for statistical behavior fail under deliberate misuse.

112. The passwords and key phrases that people can easily remember and use are too short and predictable to make strong keys. The long binary numbers that constitute public keys are totally beyond human memory and data entry error rates. Hence, keys are typically stored in system in encrypted form, with a weaker pass-phrase being used to unlock the real key. If the encrypted key can be extracted and taken off-line for pass-phrase attack, it is likely to fail. Even easier for the attacker is a keystroke capture attack, which enables the attacker to extract the pass-phrase and then to recover the user's private key.

interfaces or functionality) where none exist.[113] Unfortunately, e-commerce requires substantial system security, discussed *infra* at Part D, that *cannot* be met by adding functional modules after the fact, due to the layering, modularity, and network interface vulnerabilities. Ergo, many of the commercial products in the e-commerce marketplace are flawed by design.

Electronic Signatures

A document need not be signed to be admitted into evidence if it is authenticated. An electronic signature is one way to authenticate a document. A signature is defined as "any symbol executed or adopted by a party with a present intention to authenticate a writing."[114] The movement of many human activities to electronic media, including e-commerce, requires establishing trustworthy electronic signatures. Unfortunately, electronic signatures are susceptible to virtually all of the vulnerabilities of security components,[115] making the validity of most e-signatures technically questionable. The legal validity of such signatures is a different issue. Some states have enacted specific electronic acts that define and govern digital signatures. Other states have adopted uniform acts, such as the Uniform Electronic Transactions Act (U.E.T.A.)[116] which broadly validates the use of electronic signatures and records and imposes minimum requirements on the form and technology used.[117] The Electronic Signatures in Global and National Commerce Act (E.S.I.G.N.)[118] ensures that such instruments are accepted nationally.

These design flaws should not obfuscate the fact that the system implementation may be flawed. Some of these inherent design flaws expose the user to corruption, which in turn allows an attacker to

113. "The result is products that aim to meet a market. Customers want more features rather than strong security, so vendors introduce complexity, which leads to greater likelihood of failure." Alexandra Wever Morales, *Intrusion Detection*, Software Development Online, Apr. 2000, (quoting Prof. Eugene Spafford, director of Purdue University's Center for Research in Information Assurance and Security), http://sdmagazine.com/documents/s=745/sdm0004d/0004d.htm (last visited Apr. 1, 2003).

114. U.C.C. § 1-201 (39) (2000).

115. John R. Michener & Steven D. Mohan, *Clothing the E-Emperor*, IEEE COMPUTER, Sept. 2001, at 116.

116. California enacted U.E.T.A. effective January 1, 2000. CAL. CIV. CODE § 1633.1 et al. (2003).

117. Drafted and presented to the States in 1999 by the National Conference of Commissioners on Uniform State Laws, U.E.T.A. had been adopted by 28 states and was pending in 17 more as of April 2001.

118. Electronic Signatures in Global and National Commerce Act, Pub. L. No. 106-229, § 101, 114 Stat. 464, 464.

compromise the financial credentials (typically credit card numbers) of third parties.[119] Such compromises can impose substantial costs to both financial institutions and end users.

The market for software and system products with integrated security functionality as well as stand-alone security products exhibits unusual and reprehensible characteristics because virtually all of the players (the vendors, the analysts, and the customers) are unwilling to address the underlying lack of system integrity. Indeed, the systems managers' and executives' ignorance and their "sticking their collective heads in the sand" inaction concerning security issues must be viewed as willful. Because vendors are confident in their ability to avoid liability for shoddy work, they can produce products faster and cheaper than a competitor who is serious about ethically and functionally addressing these issues.

D. *Some Security Issues of e-Commerce Servers*

In a commercial setting, vendors desire to limit their liability; buyers, however, want a representation that what they buy will work for the intended purpose and, if it does not, they will be made whole for subsequent damages. This open-ended liability for consequential damages is unattractive to a vendor, who will often attempt to limit the scope of a warranty. Even where scope is limited, the vendor will also attempt to limit its liability for damages for a breach of warranty.

An e-commerce server or application is a complex and security-critical product.[120] End users, from individuals to large organizations cannot be expected to a) understand the myriad of vulnerabilities[121] that these systems may possess, or b) the variety of attacks that can be mounted upon them. Consequentially, it should be the vendor's responsibility to provide a product that is merchantable for the purpose for which it is sold.

As an example of a security related product, consider the case of an e-commerce server. To simplify, assume that an e-commerce consumer

119. Flaws in e-commerce servers have allowed wholesale capture of credit card numbers stored in the systems. Even if the e-commerce software does not have easily exploitable holes, if the underlying operating system is vulnerable, an attacker may exploit these vulnerabilities to plant Trojans or modify existing modules ("corrupting" them) to allow the systematic exploitation of information stored by the e-commerce software. The attackers can then use these credit card numbers for fraudulent transactions.

120. Testing a received product cannot assure the security properties of a product. The vendor must still be held liable for consequential damages for concealing improper features in a product, such as a hidden back door that compromises the security of the product.

121. Vulnerabilities can arise from the sheer number of layers, modules and packets, as well as the combinatorial explosion of interaction among layers, modules and packets. For further discussion regarding combinatorial explosions, see *supra* note 89.

will use a credit card to buy from the server and there will be either a physical delivery of goods or a controlled download of one or more files. At the minimum, merchantability requires the following properties (*NOT* a complete list):

- Generation of complete and detailed purchase record/ invoice/audit trail;
- Ability to conduct a "secured" credit card transaction using one or more of HTTPS/SSL/TLS/IPSec protocols; and
- Ability of the system to either deliver the product or to instruct another system to ship (or enable the download) of a specified product to the appropriate consumer.

Now the vendor must make some assumptions concerning the operational environment. Do they trust the application and/or interface host to the public network to survive an attack; and, if not, is it necessary to take any action to minimize the damage in event of a successful attack? If their assumptions are inadequate, the product's merchantability is called into question since it is unlikely that the security properties of the product will be adequate. This is to be distinguished from currently adequate assumptions that become inadequate as technology naturally evolves; thus, a currently merchantable product may become unmerchantable through no fault of the vendor.

Suggested Security Measures

The experience over the past several years of repeated break-ins[122] clearly illustrates the vulnerability of standard systems.[123] An e-commerce system should be able to a) deliver its functionality in the face of sophisticated attack, and b) minimize the damage done by

122. On February 18, 2003, the New York Times reported the compromise of many millions of credit card numbers. *Entry Made into Credit Card Accounts*, N.Y. TIMES, Feb. 18, 2003, at A19. At the University of Texas at Austin, a student was charged with stealing 55,000 student records. CNET News.com, *Student Accused in Texas Data Heist* (Mar. 17, 2003), *at* http://zdnet.com.com/2100-1105-992732.html.

123. We can view "standard systems" as being those whose security is primarily dependent upon "discretionary access controls" (DAC) and which generally operate with some form of "all powerful" administrator to manage the system. DAC are controls that the user can set, such as if they want files to be readable by the public. Similarly, the system administrator chooses what rights are associated with files and processes. Conversely, more secure systems typically implement "Mandatory" or "Non-Discretionary" access controls (MAC) in addition to the user-manageable DAC rights. MAC controls override DAC. Hardened systems generally do not have "all-powerful" administrator roles. Rights are typically assigned to specific roles, and even trusted system programs operate with no higher privileges than absolutely necessary. This is called the principle of least privilege.

successful attack.[124] The systems/subsystems that directly interface to the user/public network are the most vulnerable to attack and must be not be unduly trusted with valuable information. The systems/ subsystems that perform back-end functionality must have carefully designed interactions with the front-end system to minimize the consequences of breaches of the front-end system. Reasonable starting points for system constraints are:

- If the system stores customer financial credentials, the system must be designed in such a way that customer financial credentials can be used internally, but not accessed by the external user interface. The customer financial credential subsystem must be a trusted subsystem that will not reveal its contents, *even if the host system is compromised*.
- Similarly, the subsystem handling the transaction database, accounting records, and the audit information must be implemented as a trusted subsystem that will protect this information, *even if the host is compromised*.
- All transactions and operations require strong binding of the user/customer/process identity[125] with the operation performed and all operations involving customer credential and financial transactions are to be securely logged.

Unless the product is based upon a trustworthy platform, a reasonable security engineer would expect the front-end server that is interfacing to the users to be compromised,[126] perhaps frequently. An e-commerce application that simply runs on a standard commercial platform cannot be expected to survive such a compromise: such capabilities are usually not designed in. The situation is not technically hopeless. It is possible, with explicit effort, to construct complexes (with system functionality partitioned between multiple systems with carefully defined and restricted interfaces between the systems) to significantly hinder attackers and minimize the consequences of

124. John Michener, *System Insecurity in the Internet Age*, IEEE SOFTWARE, July/Aug. 1999, at 62.

125. If the transactions are high value, it is important to require the establishment of trusted paths for user authentication and the binding of user activities and processes with transactions performed. These processes also must be securely audited.

126. The almost universal assumption that servers exposed to the public will survive dedicated attack over long periods of time is unwise. If vulnerable systems are exposed to untrusted users, system functionality must be partitioned to detect and minimize the impact of such compromises. Given the wide variety of vulnerabilities and our history in defending systems, the average engineer's expectation that we can keep servers, based upon commercial grade operating systems, running standard applications defies all reasonable judgment. *See* MICHENER, *supra* note 124.

successful compromise of the public interface systems.[127] Such systems, while having superior security properties, will always be more expensive and harder to maintain than a single conventional system running the same functionality as different processes on the same system.[128]

E-commerce installations that do not exhibit reasonable damage confinement characteristics are not, in the authors' opinion, merchantable[129] for general use. The compromise of substantial numbers of user accounts/credit/debit card numbers is strong evidence that the systems in question were either inadequately engineered, improperly configured, or both. The implementation of an e-commerce installation must limit a successful attacker's compromise to no more than the new credentials (credit card numbers) entered into the system from the time of compromise to the discovery of the compromise.[130] The system(s) must have watchdog processes that allow the timely discovery of attacks and compromises. It must have the appropriate tools to repair the damage, determine the compromised users, and invoke the appropriate business processes to deal with the issues.

Modern computing systems are highly complex. While such complexity is acceptable for environments with large and highly trained professional staff to manage them, it is not an acceptable characteristic for systems that are to be operated by small organizations with less skilled or trained staff. A product that is too complex to configure, in a secure manner, for use by managers in a typical operating environment is neither secure nor merchantable.[131] The fact

127. Such a complex of suspicious systems (i.e. systems that do not simply trust that the provided data are correct and, therefore, have their own data verification capabilities) would effectively behave as a special purpose partitioned appliance, with damage confinement explicitly engineered into it. It would accomplish, with multiple systems and controlled interfaces, what software cannot do on an untrustworthy platform.

128. Doing so entails running more computers and requires constraining the communications between the computers so that compromised front-end computers are unable to compromise the back-end systems. This could possibly involve the introduction of dedicated "guard" systems between the front-end and back-end systems. This is inherently more expensive that simply running each operation as a separate process on a single computer.

129. It might be possible to use such an application within a corporate intranet, where the server is inaccessible from the Internet. The documentation and material for such a limited application must clearly note that its security limitations make it unsuitable for use on the Internet, that it does not provide damage limitation in the event of penetration, and that it is NOT to be used as an Internet-accessible e-commerce platform.

130. Indeed, it would be possible to automatically refer new enrollees to a dedicated system for just this function. The user would return to the front-end server once the enrollment was completed. This would block the attacker's ability to acquire such credentials.

131. As an analogy, a number of valuable industrial products are so dangerous to handle that suppliers will only distribute them to customers who can provide strong assurance of their skills to safely handle the substances and prevent improper exposures either among the appliers

that an expert could configure it does not make the product secure because the average manager is not an expert. The complexity of a product may also adversely affect security. For instance, it is easier to secure a relatively simple piece of software than a large and complex software piece. When systems are designed for security, security-critical components are abstracted out and layered beneath other functionality modules so that the minimized security component can be hardened and inspected for problems (something that cannot be done well when the security components are scattered throughout a much larger system).

Homogeneous environments composed of similar machines, be they Microsoft Windows 2000, Sun's Solaris, or LINUX, are subject to catastrophic failure when (not if) an attacker finds and exploits a system loophole. The best analogy here of the possible level of catastrophic destruction is the damage that would occur if hoof-and-mouth disease got loose in a commercial pig farm. Good design involves overlapping systems to provide an additional measure of security, such that if one piece of one system's security is compromised, the comparable piece of another system will secure the breach. The discovery of a loophole in Microsoft XP early in its deployment illustrates the potential extent of such a failure.[132] A manufacturer who assures consumers that his system should be the only system present should be held liable for the full costs of such catastrophic failure, when they do occur. Similarly, consulting organizations that recommend homogeneous complexes of systems[133] should bear full liability for the costs of such catastrophic failures.

Although, e-commerce servers are common, high-value targets, they are by no means the only ones. The user's clients (PC's/workstations/ PDA's) are typically much more vulnerable. They are not as well protected or managed, and they can potentially be used as an avenue into more protected environments. Transactions

or to third parties. The customers may be required to carry substantial liability insurance as well. For instance, in the late 1970's, vendors of large-scale sprayed-foam urethane insulation required the applier to use an air-supplied respirator and to ensure appropriate public protection. Much like the insulation analogy, the complexity of a security product may require a higher customer skill level since the consequences of a security mistake are so high.

132. "This is pretty much the worst default vulnerability in any Windows operating system," a security expert said. Jay Lyman, *Microsoft Stung By Severe Windows XP Security Flaw*, NEWSFACTOR NETWORK, Dec. 20, 2001, *at* http://www.newsfactor.com/perl/printer/15458.

133. In systems that are otherwise homogeneous, internal firebreaks should be created to provide damage isolation and confinement. Guards, which may be viewed as variants of application proxy firewalls, can provide such isolation properties. *See* John R. Michener & Steven D. Mohan, *How to Shrink Holes in Corporate Data Dikes*, IEEE IT PROFESSIONAL, Jan./Feb. 2002, at 41.

conducted from such untrusted clients of the users do not have significant integrity. The security features advertised to protect user's critical information can quite easily be circumvented. User authentication and transaction authorization for application programs such as banking, stock trading, and the like are inherently vulnerable to capture and misuse by attackers with modest skills. Security products aimed at the client platform frequently make inappropriate assumptions about the client platform security properties. The result is that the advanced technologies (such as fingerprint authentication) the security product vendor are supporting may have worse security properties than the old technology (passwords) that they are replacing.[134] U.C.I.T.A. clarifies the rights of persons whose transactions (e.g. the enforceability of releases) fall within its scope.[135] Even if a jurisdiction in which the parties reside has not enacted U.C.I.T.A., the parties can generally provide that their contract will be governed by U.C.I.T.A.[136]

IV. POTENTIAL SOLUTIONS

Given the importance of security to e-commerce and the potential social costs of security failures, it would be logical for the legal system to encourage strong protections. Given that e-commerce issues are often—perhaps always—of national or international scope, it would also be logical for the legal framework controlling contracts for security products to be consistent nationwide. As outlined above, *supra* Section II, the law does not provide adequate protections, nor is it consistent. These legal inconsistencies argue in favor of a unified approach to the resolution of warranty issues, either through the adoption of U.C.I.T.A. or a U.C.I.T.A.-like uniform state code, or through a federal statute.[137] In either case, however, an approach must be taken that rationally reflects the economic realities of modern software transactions. Currently, all

134. This problem is far more severe that is generally recognized. The level of vendor misrepresentation concerning product security properties can be best described as "caveat emptor". The issues were presented at the 1998 CardTech/SecurTech conference. *See* John R. Michener and Daniel G. Fritch, *Integrity of Advanced Authentication Technologies: Biometrics, Smartcards, and Client Protocols*, 1 CARDTECH/SECURTECH CONFERENCE PROCEEDINGS 191–210 (1998).

135. U.C.I.T.A. § 208.

136. U.C.I.T.A. § 103(e).

137. The U.C.C. and U.C.I.T.A. carve out exceptions to this general assumption for "consumer" transactions between an individual and a merchant. This carve-out acknowledges the unequal balance of bargaining power and knowledge between the individual consumer and the merchant. While the U.C.C. and U.C.I.T.A. address some issues of imbalance between consumer and merchant, the heightened protections given to consumers do not address the problems encountered by commercial entities in security transactions.

structures in place for determining the risk of failure fail to account for the modern realities of information technology transactions.

An underlying assumption of commercial contract interpretation is that each party has an equal opportunity to evaluate the risks that the contract allocates. The business-to-business model worked when each party actually did have access to the same resources (e.g. attorneys who explained contract risks). In contrast, in an information technology transaction, a business cannot solely rely upon legal counsel because the risk does not arise from the terms of the contract. Instead, the risk arises from the infrastructure of the product itself. For the businesses to have an equal bargaining position they each must also be able to evaluate the product's infrastructure. The U.C.C. and U.C.I.T.A. carve our exceptions to this general assumption for "consumer" transactions between an individual and a merchant on the basis that the assumption of equal bargaining power and knowledge is accurate when analyzed in the context of the individual consumer (who may enter into this particular type of transaction only once) and the merchant (who, hopefully, is entering into thousands of similar transactions).

Products designed to provide information security compound this difficulty. With security products, the seller cannot reveal to the buyer all the information necessary to evaluate the risks without reducing the value of that product. Safety measures must necessarily be confidential to help maintain security.[138]

The seller should not be forced to solely bear the risk of loss associated with a security product's failure. Nonetheless, unlike the current socially inefficient system, the seller should not be able to absolve itself of all risk either.

It is socially inefficient to allow a seller to absolve of any responsibility for harm resulting from a product's failure. This would shift responsibility away from the party most able to prevent it and would shift the cost of production from the seller to society. As the seller never accounts for the cost of a security breach, the product will not reflect the transaction's true cost. It may be that the total true cost of the transaction is prohibitive.

A. *Possible United States Governmental Role*

A major portion of the software and Internet community views the U.S. Government with considerable suspicion, perhaps because of governmental interest in monitoring communications. The controversy about

138. The most secure systems are secure even when their mechanisms are known; nonetheless, even with such robust systems, the lack of knowledge of the mechanisms provides yet one more hurdle for a would-be hacker.

key-escrow in cryptography[139] (for example, the Clipper chip in the early 1990's) did not ease this suspicion. Despite the common perception of the government as "Big Brother," the Government's interest in securing the national infrastructure (including local and international telecommunication networks, the domestic Internet, and electronic financial and medical information systems) is largely the same as the interest of other users of this infrastructure. Fortunately, the Government has more talent and knowledge than other users; therefore, it is more capable of devising and employing standards of quality that products must meet if they are to be sold to the Government or its contractors.

The U.S. Government has become increasingly concerned about the vulnerabilities that have been and are being created in the U.S. economy. These concerns are due to the computer systems and software industry's negligent approach. The Government itself has become vulnerable due to its use of commercial off-the-shelf products. In 1997, the National Security Agency (NSA) and the National Institute of Standards and Technology (NIST) established the National Infrastructure Assurance Program (NIAP)[140] to educate the industry and promote good practices. The NSA also established the Information Assurance Technical Forum (IATF) and an associated web site to discuss security and assurance problems and approaches for their management and solution.[141] While the IATF is focused on providing information assurance guidance to the U.S. Department of Defense, its analysis and advice is applicable to the commercial environment. The Government is starting to get commercial attention for safety and security by requiring that products sold to the government meet its standards.[142] Also, evaluation organizations are offering certification. Certification involves examining design documentation, source code, specify proper usage of the product, and specify the product's function. The evaluation process can be demanding,

139. Key escrow is a system where an escrow agent keeps a copy of a user's master key. This key can be obtained from the escrow agent if the user's master key is lost or if an authorized agent submits a request for the key accompanied by the appropriate documentation, as appropriate for the local legal environment. In corporate environments, key escrow is called key recovery and is an absolute business requirement for keys used for data secrecy. Where key escrow applies to individuals, it is a matter of national legal requirements. There are no international standards here.

140. *See* http://niap.nist.gov, *supra* note 3.

141. *See* http://www.iatf.net, *supra* note 3.

142. Starting July 1, 2002, NSTISSP # 11 mandates federal government use of evaluated products if handling, processing, storing, transmitting, etc. classified data. The NIAP website contains a link for frequently asked questions regarding NSTISSP # 11. *See* National Information Assurance Partnership, NTISSP#11 Frequently Asked Questions, at www.niap.nist.gov/cc-scheme/nstissp-faqs.html (last visited Mar. 20, 2003) (on file with the Michigan Telecommunications and Technology Law Review).

expensive, and time consuming where security was not appropriately considered in the initial design, and where good development practices were not followed. The evaluation level that a product receives is a function of its measured properties and of the quality and thoroughness of its design and implementation documentation.[143] Product evaluations are done under the "Common Criteria"[144] and the results are associated with an "assurance level."[145]

Commercial levels of evaluation are typically in the EAL2 through EAL 4 range.[146] The assurance ranges associated with the old Trusted

143. While this may seem strange, what is being established is the level of "assurance" or trust that the reviewing organization has in the implementation. Lacking "formal" mathematical tools to prove the correctness of sophisticated implementations, higher levels of "assurance" are established by design and careful review. Reviewers will look at architectural specifications, high-level design specifications, test specifications, code review reports, and the code itself. Reviewers look for consistency, correct implementation, and clarity. Higher levels of assurance require progressively more demanding processes and reviews and yield higher levels of trust that there are no unexpected behaviors in the product. These issues are discussed in exhaustive detail in the "rainbow" books, of which the "orange" book is a well-known member. The rainbow books may be found at http://www.radium.ncsc.mil/tpep/library/rainbow.

144. *See* SCHNEIER, *supra* note 54, at 131–33; ANDERSON, *supra* note 88, at 526. Anderson also points out some of the weaknesses of Common Criteria evaluations.

The Common Criteria and the older ITSEC criteria evaluate products on two independent axes: the first is the functionality, which is specified by a "protection profile", and the second by an assurance level, which is chosen from among one of a set of assurance levels. The protection profile specifies what the product does and does not do. The assurance level is a rough level of trust that the evaluating agency can establish concerning the correctness of a product, as well as its lack of improper or unexpected characteristics. As the assurance level climbs to higher numbers, the level of proof and the associated effort involved to establish correctness climbs rapidly.

145. The security evaluation of systems is a complex issue and a number of different criteria have been developed over time. We use the Common Criteria because it is a shared evaluation standard that is intended for use by industry and lower security governmental applications.

146. The Common Criteria has seven assurance levels (EAL1 to EAL7). The Common Criteria's assurance levels are roughly equivalent to the IT Security Evaluation Criteria (ITSEC) at one lower criteria level. For instance, the Common Criteria's EAL1 is roughly equivalent to ITSEC's E0. EAL2 is similar to E1, etc.

EAL1 represents inadequate assurance. At the EAL 2 level, there shall be a security target and an informal description of the architectural design of the evaluated Target Of Evaluation (TOE). Functionality testing shall indicate that the TOE satisfies its security target. EAL3 adds that there shall be an informal description of the detailed design. Evidence of functional testing shall be evaluated. There shall be a configuration control system and an approved distribution procedure. EAL4 builds on EAL3 by additionally requiring the source code and/or hardware drawings corresponding to the security mechanisms to be evaluated. Evidence of testing of those mechanisms shall be evaluated. EAL5 additionally requires an underlying formal model of security policy supporting the security target. The security enforcing functions, the architectural design and the detailed design shall be specified in a semiformal style. In EAL6, there is also a close correspondence between the detailed design and the source code and/or hardware drawings. Finally, EAL7 adds that the security enforcing functions and the architectural design shall be specified in a formal style, consistent with the specified underlying formal model of security policy. For further explanation, see Defence Signal's Directorate, *Evaluated Products List*, *at* http://www.dsd.gov.au/infosec/aisep/EPL/section8.html (last modified Mar. 25, 2002).

Computer security evaluation would map the B1 class to an EAL4 level of assurance, with C2 mapping to an EAL3 level of assurance which may be viewed as the level of trust that may be placed in of the evaluation. Assurance levels range from EAL2 through EAL4 for general commercial systems and up to EAL7 for specialized Governmental systems.[147] Higher assurance levels may be necessary for critical functionality.[148]

The NSA, through NIAP and the IATF, is trying to provide a framework for the assessment of standard critical infrastructure components (firewalls, biometrics, etc.) and is planning to provide guidelines on using these components.[149] While the NIAP/IATF has little impact at this time in the commercial industry, its evaluations and guidelines are likely to provide a reasonable basis for the expectations of users and the efforts of integrators in the future. Unless and until a large and sophisticated neutral organization[150] is formed to provide guidelines upon the specification, evaluation, and usage of system components to provide secure operations, it seems likely that the NIAP/IATF standards and guidelines will eventually provide the minimum operational and specification standards for standard components. We should expect that the recommended evaluation levels of such components would gradually increase as the sophistication

147. *Id.* at §§ 23.3.1 and 23.2.1.

148. The German digital signature law places stringent technical requirements for the validity of electronic signatures. It requires that the signing private key be protected by a hardened smart card, which does the signing, that the smart card reader be evaluated to E2 (approximately equivalent to EAL3), and it requires that the smart card's handling of the user's private key and the commercial Public Key Infrastructure Certificate Authorities be evaluated to E4 (approximately the same as EAL5). *See* Verordnung zur digitalen signatur, SigV, § 17, ¶ 1, *available at* http://www.iid.de/rahmen/sigv.html. The E4 requirement for the PKI CA is far more demanding than the E2 for the reader. The E4 requirement for the smart card key handling is not exceptionally difficult to meet for such a special purpose operation. While generally regarded in the industry as overly demanding, the technical authors view requirements for such levels of assurance as reasonable, given the importance that such technologies may have in the future.

149. *See* http://www.iatf.net/protection_profiles.

150. It appears rather unlikely that many users or organizations want the U.S. Department of Defense establishing commercial practices and standards. The DOD is a sophisticated customer that it trying to protect its own systems and infrastructure as well as supporting U.S. civilian infrastructure. In the process, they are giving the sophisticated computer user community a free ride. In time, it may be necessary for some organization to rise up and provide similar standards and best practices for the commercial community, including suppliers of critical infrastructure. It is necessary that any replacement organization take on the protection of the users, and not the convenience and protection of the manufacturers, as its mission. The IETF (the organization establishing standards for the Internet) cannot fulfill this role. Its focus is upon establishing protocols, and it suffers from the classic committee syndrome of throwing too many features (after all, most of the staff are supported by various commercial organizations and are expected to get "their" features included) into various standards, creating serious problems of integration, testing, and assurance. They are not concerned with problems of vendor implementation assurance or quality, which are critical issues for any security implementation.

of attacks grows. Similarly, we should expect the introduction of new standard components to support additional services and to provide more defenses against new or shifting attacks.

Aspiring to "Best Practices"

While there is no reason to require that manufacturers submit their products for evaluation under NIAP/IATF guidelines, these guidelines and evaluation criteria do provide useful reference standards. The standards are established by a large, knowledgeable, security-sensitive organization that is trying to establish solid, fact-based requirements and procedures that provide effective security to the deploying organization. A manufacturer of a product providing security functionality should be prepared to show that they have used development and oversight processes at least as thorough and demanding as those of equivalent NIAP rated systems (if applicable) in order to establish due diligence in the development of their product. There should also be clear and unambiguous directions for the proper usage of their product, including known dependencies and known and reasonably expected product characteristics.[151] Clearly, organizations pioneering new components and developing new technologies will not have the guidance that NIAP provide.[152] As long as they can show that they used prudence, due care, and diligence in developing their product and its associated marketing and usage materials, manufacturers of such leading edge products should be regarded as pioneering best practices.

When manufacturers have their systems and their appropriate usage evaluated by competent organizations, such as required by the U.S. Government's NIAP program, they should be viewed as following best practices.[153] Organizations that meet or exceed these same standards and

151. For example, if the product is a secure document delivery product using the user's Internet browser as a client (the user's access to the product), the documentation should clearly state that the product can be compromised by an attack upon the browser and that maintaining the user's expectations of security requires the use of certain browser settings (settings to be provided by the vendor). It should also state any other conditions that are required for proper operation and what the product does and does NOT do. For example, in addition to the classic browser attacks of applets and ActiveX, such a system is vulnerable to spyware capture of access passwords, and illicit copying of protected documents, perhaps by spyware, to non-controlled destinations.

152. NIAP/IATF are concerned with the properties and proper usage of existing products. Innovators who develop new technologies will be operating outside of the NIAP/IATF purview but, as these new technologies are understood and their characteristics documented, they will be integrated into the NIAP/IATF environment. *See also* discussion on NIAP/IATF in *supra* note 3.

153. Unfortunately, best practices may still be rather inadequate. Microsoft has recently completed an EAL4 evaluation of Windows 2000 for the governmental marketplace. The protection profile against which Windows 2000 was evaluated can be viewed as stating that such

requirements are also following best practices, although they must expect that they may be required to prove their capabilities, assurance, processes, and the comprehensiveness and appropriateness of their documentation, in event of mishap.

Organizations not willing to pursue and live up to such standards should not be involved with security or security-critical products. Without the processes and controls needed for independent evaluation, the manufacturer, let alone the customer, has no understanding of the product's security properties. Security products not subject to the Government's program requirements should be subject to product liability protections when sold to ignorant consumers.

B. *Potential Legal Solutions*

In the security related product market, imbalances between seller and buyer produce socially inefficient results. What then are potential solutions? Several possibilities suggest themselves, some more radical than others.

The first is simply for courts to recognize that attempts to shift the risk of loss to a customer in the security context is unfair because of the inherent nature of the transaction, which, in most cases, rises to the level of unconscionability. This solution, however, is questionable because the court's are reluctant to find freely negotiated contracts unconscionable, especially in a business context.

Another solution is to shift the burden to the producer via statute, either as a stand-alone statute or folded into a comprehensive computer information statute such as U.C.I.T.A. This statutory solution could take two forms: either the statute sets forth a simple default rule, or the statute permanently shifts the burden to one side or the other, without allowing variation. Each has its disadvantages.

The weaker first construct suffers from the obvious deficiency that the rule could still be varied and such variations would inevitably occur through form contracts,[154] and, perhaps, even negotiated contracts where one party (generally assumed to be the seller) has access to greater knowledge of the law and the risks.

Nonetheless, even simple default provisions would help, at least if it places the burden on the party likely to have the greater power to reduce the risk *and* require that any modification of the default terms be explicit and conspicuous. The modifying clause at least brings the other party's

systems should not be generally connected to the Internet. *See* Jonathan S. Shapiro, *Understanding the Windows EAL4 Evaluation*, IEEE COMPUTER, Feb. 2003, at 103.

154. Alex Y. Seita, *Uncertainty and Contract Law*, 46 U.PITT.L.REV. 75, 127.

attention to the issues. Nonetheless, while this assists in solving the issue of the customer's ignorance of the law, it does not address the more difficult issue that the buyer is still incapable of accurately assessing the actual magnitude of the risk involved and should not be left without recourse as a result.

The other alternative is to require the default risk allocation to be mandatory. Of course, this need not place the risk with the same party in all circumstances. For example, the statute could allocate risk to the seller in all transactions and where the seller has made particular representations about the product (even if it later attempts to disclaim the representations), but then relieve the seller of liability and risk where the buyer was modified the seller's code in an unauthorized manner. The advantage of this approach is that it prevents the seller from imposing upon the buyer terms that are economically inefficient to society for its sole advantage.[155] The disadvantage is that the law is substituting the judgment of the legislature as to the fairest and most economically efficient allocation of risk for the parties' judgment. The court believes the parties should be in a better position to evaluate the particulars of the transaction.

CONCLUSION

The commercial software and systems industry now provide critical infrastructure for our society. This industry developed in an environment where it could pass the risks of its errors and oversights to its customers who had no way of knowing what their risk exposure was, is, or will be. Willful and systematic security oversights in the industry created widespread vulnerabilities that create the potential for exceedingly large third party losses. The imbalance in knowledge between the supplier and purchaser is so extreme that the legal assumption of comparable sophistication should be abandoned and the presumption should instead be that the purchaser is an ignorant consumer with associated product protections.[156] Executives and management in the software and systems industry are usually willfully ignorant of and unresponsive to critical security issues and their implications for product design and development. The software and systems industry, and the associated consulting

155. *Id.* at 130.

156. Clearly very large and sophisticated organizations can be held to higher standards. Manufacturers who have their systems and their appropriate usage evaluated by appropriately competent organizations, such as those the U.S. Government's IATF program requires, are following best current practices. Their customers, who will be provided the evaluation report and appropriate usage, should not be treated as ignorant.

organizations, must be held to reasonable standards of implicit and/or explicit product merchantability with full consequential liability. This allows a customer to know what a product will and will not do, and how to reasonably maintain it so that it will perform its function without creating or allowing unacceptable losses for the customer or third parties.