Who Is Liable for Insecure Systems?

The notion of liability for insecure computer systems has shifted. Over the past several years, we have seen both the enactment of legislation affecting liability and the appearance of actual liability cases in the courts. For many years, computing professionals have debated the topic of liability for insecure systems. Erin Kenneally provides a summary of this debate’s legal aspects and observes that claims against insecure systems have evolved from a veiled threat into a ripening promise.

Several software flaws can make systems insecure. According to Kenneally, they derive from the current software status quo:

- Vendors are aware of errors and vulnerabilities in software created by their coding.
- Vendors can foresee the potential for mass failures and disruptions due to exploitation of insecure software.
- Users know that misconfigurations and software flaws can allow an intruder to gain unauthorized access to their systems.
- Users know that vendors provide patches, upgrades, and notices to address software vulnerabilities.
- Users have no knowledge of insecure design and no ability to alleviate its effects.
- Onset time for exploiting vulnerabilities starts as soon as the product is installed.

Kenneally concludes that it is reasonable for users to assume that software products will not allow intrusions. She further notes, however, that software products have, to date, had security holes that allowed intrusions costing billions of dollars.

Because of these flaws, an entire industry has developed to try to mitigate security holes after the fact. Software vendors develop and announce patches for security holes. System administrators busy themselves applying security patches. Other vendors sell products such as virus detection tools, and consultants perform penetration testing. Network technicians install firewalls and configure them to block intrusions. Clearinghouses provide a continual stream of information about the latest viruses, worms, and Trojan horses. The issue of liability lurks just beneath the surface of all these activities.
LIABILITY OVERVIEW

Software and data quality share an important relationship. As Joseph Juran observed, “Data are of high quality if they are fit for their intended uses by customers in operations, decision making, and planning.” Recently, we have seen an emphasis on accountability for data and software quality. Specifically examining data, we are told that those who create data models and values must be held accountable for the quality of those models and values. Likewise, those who develop applications must be held accountable for the quality of the data’s presentation.

Sources of corruption

In a related discussion about corrupted data, Robert W. Pautke suggests eliminating the sources of corruption and applying human and financial resources to protecting only the most important data.

Corrupted data is only one piece of the puzzle, however. The suggestion here is that hackers disrupt not only networks, but also applications and data. Indeed, many researchers feel that if we can protect applications, it might not be necessary to defend networks so rigorously. In an editorial commenting on software quality, Karl Reed describes a litany of all too familiar software woes and cites massive “societal costs due to wasted time.”

The notion of accountability for software and data quality naturally leads to the question of manufacturers’ liability for software. Several years ago, a study that used students as attackers to evaluate commercial off-the-shelf (COTS) software products yielded the following results:

- Most attackers performed successful intrusions.
- Several of the intrusions gave the attacker administrator privileges.
- The Internet provides a vast amount of information on how to successfully attack common systems.
- Many attackers broke into the system by using exploit scripts published on the Internet.

Liability concerns related to security fall into several categories. They include concerns about safety-critical systems, Internet-connected consumer appliances, businesses with vulnerable computer networks, distributed denial of service (DDoS) attacks, and vendors who produce commercial software.

Kenneally provides a tool to help determine potential liabilities. System and network administrators can use the liability test worksheet shown in Figure 1 to identify the kind of agents who perpetrate data crimes, the means used to compromise target systems, and the specific nature of these attacks. Figure 2 shows a sample worksheet that links various causes, agents, and data crimes.

Determining liability

Some computing experts have argued that manufacturers and suppliers of computers associated with safety-critical systems should be liable for injury those systems cause, including injury that results indirectly from security holes. This suggests that liability resulting from security problems in Internet-connected appliances could be treated similarly to general-appliance liability. If appliance manufacturers are liable for other types of failure, they or their vendors could also be held liable for failure resulting from security holes.

Many analysts recommend that the burden of protection against attack should lie with companies and businesses that have the resources to provide such protection, not with the general public. They discuss liability for monetary loss as a result of hacking, as well as reducing liability by the appropriate use of contracts and insurance. Another report suggests that firms most capable of protecting against DDoS attacks are those most...
likely to be liable if damage occurs. The National Academy of Sciences recommends that software companies be held liable in cases in which buggy or insecure software causes security gaps.

Another view suggests that security is both an economic and technical issue. Finding all the security holes in a particular piece of software is expensive, but this must be done to ensure security. Yet it costs a hacker little to find a single security hole to exploit. So the numbers tend to work against the security effort. An emphasis on making software easy to use provides another disincentive that tends to mitigate against increased security.

Some companies demand liability clauses in contracts with vendors to hold them responsible for any security breach connected to their software. Officials at these companies expect such language to become more prevalent in contracts. They see the trend as a constantly reinforced motivator urging programmers to focus on security. They also recommend that companies protect against liability suits by adopting policies that adhere to information security standards and best practices.

**Diverse viewpoints**

A recent series of articles by two software liability experts shows just how divergent individual viewpoints can be. On the one hand, some experts argue that expecting consumers to create their own security software is impractical. They assume, rather, that it is only reasonable for software manufacturers to ensure their products’ reliability. The notion of fitness for use suggests that such software provides a basis for strict liability, a concept that has been applied in personal injury cases, but not so much in property damage cases and even less in economic damage cases. In these latter cases, the courts often accept contractual disclaimers of liability, such as those found on shrink-wrapped software.

Network security and the associated liability issues have also received attention. A negligence argument can be made in support of liability for insecure networks. This argument would need to show that the insecure party “had a duty to use reasonable care in securing its computer systems, breached that duty by failing to employ adequate security, and was a reasonably recognized cause of actual damages.”

Since strict liability is not applicable in all cases, the courts must build on other concepts, such as “warranty of fitness, misrepresentation, abnormal danger, negligence, fraud, lack of clarity, and unconscionability to find liability for all security product failures.” The doctrine of unconscionability has been applied to contracts and, when safety is at issue, disclaimers have been invalidated. This argument suggests that litigants can successfully make liability arguments in the US legal system for at least one class of software products.

The opposing view suggests that liability is not the appropriate tool for reducing the number and severity of software security holes. This argument suggests that software does not have the characteristics of other physical products that would support legal liability. For one thing, a liability case could take years to move through the courts, in which case the software—having a relatively short
Users demand flexibility, but this makes a given software product’s uses less predictable.

life cycle—could become obsolete before the case came to trial. Thus, even if software liability is confirmed, the outcome could be irrelevant.

Another argument against software liability holds that manufacturers cannot predict how software will be used or where and how it will be installed. This lack of predictability makes it impossible for manufacturers to warrant software for fitness of use. For example, users can tailor many commercial software products through preferences. Users demand this flexibility, yet it makes a given software product’s uses less predictable, thus making it secure is more difficult.

In the past, the courts have distinguished between manufacturing defects and design defects, with the notion of software liability typically resting on design defects. A software manufacturing defect could occur, for example, if an improperly produced software CD made the product either impossible to install or otherwise invalid. Liability actions would seldom take place in such a case, because the standard disclaimers usually provide for only replacing the product or refunding the purchase price.

The final argument that liability is inappropriate for software suggests that many software manufacturers would leave the business if faced with liability suits, leaving few remaining manufacturers to service a large and burgeoning marketplace.

Given the diversity of views regarding how to assign liability when a system is compromised, it’s not easy to identify a uniform approach to solving the problem.

**UCITA**

The National Conference of Commissioners on Uniform State Law developed the Uniform Computer Information Transactions Act from a series of more than 20 three-day meetings by the Drafting Committee. As many as 120 interested observers attended each session. Now in its 112th year, the NCCUSL comprises more than 300 lawyers, judges, and law professors appointed by the US’s 50 states, the District of Columbia, Puerto Rico, and the US Virgin Islands. Its members draft proposals “for uniform and model laws on subjects where uniformity is desirable and practicable, and work toward their enactment in legislatures.”

The NCCUSL approved UCITA, a uniform statute intended to codify current law and practice in contracts for computer information. UCITA does not purport to answer every question, but it does provide a framework within which courts can analyze questions. The rationale for UCITA’s development stems from the combination of growth in the information economy and the corresponding lack of a uniform framework for licensing. UCITA sets up a series of default rules that apply in the absence of a specific agreement by the interested parties.

Virginia adopted UCITA first, after a one-year study period by a special legislative committee. Although Virginia’s legislature considered many amendments, it ultimately adopted UCITA in 2000 with no significant change. Maryland also adopted the act in 2000.

UCITA’s scope is limited to transactions in computer information. Such a transaction constitutes “an agreement or the performance of it to create, modify, transfer, or license computer information or informational rights in computer information.” UCITA defines computer information as information in electronic form, obtained from or through the use of a computer, or which is in a form capable of being processed by a computer. Under UCITA, contracts can be made computer to computer or human to computer. Further, the act codifies existing case law for shrink-wrap and click-wrap contracts.

Section 105 states that federal law preempts UCITA, as do state consumer protection laws. Contracts under the act may not contain terms that violate fundamental public policy. For shrink-wrap contracts, a licensee may not manifest assent to the terms of a license until he or she has had an opportunity to review them. If the license is presented after payment, the license must provide the licensee with a cost-free right of return.

For click-wrap contracts, similar rules prevail. The licensee must have the opportunity to review the terms prior to manifesting assent. Further, UCITA encourages pretransaction disclosure of terms in Internet transactions:

You ‘manifest assent’ if, after having an opportunity to review a record or term, you authenticate the record or term, or intentionally engage in conduct or make statements with reason to know that the other party or its electronic agent may infer from the conduct or statement that you assent to the record or term. … A person has an opportunity to review a record or term only if it is made available in a manner that ought to call it to the attention of a reasonable person and permit review.

UCITA creates, for the first time, statutory implied warranties in information transactions. The warranties include noninterference and noninfringe-
ment, merchantability of the computer program, informational content, fitness for the licensee’s purpose, and system integration.

Many computing professionals perceived that the UCITA framework provided liability protection to vendors. Those involved in developing UCITA, on the other hand, argue that all UCITA does is transfer existing legal results from one medium to another. Many responsible professional software organizations and engineers feel that UCITA goes too far in protecting vendors from liability. These issues continue to generate considerable discussion.

The UCITA authors contend that consumer advocates sought broad consumer protections within UCITA, rather than leaving it to individual states to develop them—a precedent that arises from traditional consumer law. The original UCITA positions on reverse engineering, public comment, and electronic self-help were opposed by computing professionals, as were the original positions on default rules for number of users and duration of license. Some computing professionals oppose shrink-wrap contracts on principle. The NCCUSL, never responding to this last point.

In 2001, the American Bar Association appointed a special working group to evaluate UCITA. After a three-day meeting held to discuss these concerns, the ABA issued a report suggesting 11 specific and substantive changes, including the following:\(^{11}\)

- **Electronic self-help banned.** Vendors of digital information, including software, may not disable the use of that information by electronic means if an information contract is breached. Vendors have an expedited remedy for a material breach of contract in a court of law.

- **A state’s consumer protection law trumps UCITA.** An information contract is expressly subject to and cannot waive any consumer protection provided in state or federal law. Included are laws providing for conspicuous disclosure, unfair or deceptive trade practice laws, and laws relating to electronic signatures and records.

- **Right to criticize protected.** Information contract terms that prohibit criticism of an information product are unenforceable. Parties can contract in a manner consistent with other law, such as the law of trade secrets.

- **Remedies for known material defect preserved.** Such remedies are expressly made available for information products as fully as for defective goods or services.

- **Reverse engineering for interoperability expressly authorized.** An information contract cannot prohibit reverse engineering done for the purpose of making an information product work together with other information products.

- **Special open source software provisions.** The act expressly excludes coverage of open source software if only copyright permission is given and is not part of a contract. If there is a contract, there are no implied warranties if the transaction generates no commercial gain.

NCCUSL also adopted 38 amendments to improve clarity, with no substantive effect. The ABA House of Delegates was to consider UCITA approval in February 2003. The ABA, however, preferred not to take a position on UCITA when it became clear that a consensus was unlikely to emerge.

UCITA was under consideration in additional states. The American Electronics Association, a high-tech trade association with more than 3,000 members, endorsed UCITA, but software professional organizations and individual professionals continued to express concern.

The NCCUSL has dropped active lobbying for adoption of UCITA.\(^{15}\) To date, only Virginia and Maryland have actually adopted UCITA. Although another five states were expected to pass the proposal last year after amendments were made to address consumer concerns, this did not occur. The amendments did not satisfy the American Library Association, the American Bar Association, or the Americans for Fair Electronic Commerce Transactions. The NCCUSL is not expected to revisit UCITA for another four to five years. At present, it seems likely that if litigation occurred, a vendor would try to benefit from having the case heard in Virginia or Maryland.

Lacking a uniform code that specifically describes the level of system security computer users can reasonably expect, the average user must look elsewhere, for example, to software vendors for improved security.

**COTS DEVELOPMENT PROCESSES**

Most COTS vendors focus on providing features. They typically eliminate the most serious problems to be first to market, then fix other problems after release. This focus on features tends to result in buggy, insecure software because, until recently, security has not been a vendor priority. Major vendors—including IBM, Microsoft, and Sun—have
Major vendors have produced tools to aid in the development and debug process.

Computer

produced tools to aid in the development and debug process and to eliminate bugs that could lead to security holes.

**Microsoft security push**

Microsoft has made a major commitment to security improvements in its software, spurred on by a now-famous memo from Bill Gates (www.computerbytesman.com/security/billsmemo.htm). This memo, drafted in January 2002, resulted in significant changes within Microsoft. For one thing, during February and March 2002, all Windows feature development stopped while the Microsoft team analyzed design, code, test plans, and documentation. For a market-driven company, this is an unusual step, and it shows just how seriously Microsoft took this security initiative.

Training courses were developed and delivered to support the Windows Security Push. As part of this effort, the team observed that security is not just a layer added after the fact, but a consideration that pervades all of development.

A key element of the design process was the construction of threat models. In this regard, there is some parallelism between the Microsoft effort and the guidance of the Common Criteria. Microsoft uses the following steps when constructing threat models:

1. Decompose the application to determine the system’s boundaries or scope.
2. Determine threat targets and categories using components from the decomposition process and determine the threat categories for each target.
3. Identify attack mechanisms using a threat or attack tree approach.
4. Respond to the threats with mitigation techniques appropriate to each threat.

Obviously, this was and continues to be a massive effort for Microsoft, which stated that the need for more secure platforms to support future solutions, a prior successful effort to push .NET, and the ability to respond to a new generation of Internet threats prompted this push. Further, the company expects that customers will perceive improved security in the products.

I wonder, however, whether a concern about possible future liability is also part of Microsoft’s agenda. Regardless of the motives, this effort seems to have great potential benefits for consumers of commercial software products. Yet a recently published report \(^7\) takes an opposing view and suggests that Microsoft has engaged in monopolistic practices that have in part contributed to excess complexity and security flaws. The report further observes that to improve security we must first address the dominance of Microsoft’s products.

**Liability litigation against Microsoft**

In a recent California development, a consumer claiming to be a victim of identity theft via computer hacking held Microsoft responsible for failure to secure its software against viruses, worms, and other cyberattacks, and initiated a lawsuit. Whether this will evolve into a class action suit remains to be seen, but the outcome will be extremely interesting from a liability viewpoint, especially in light of the Security Breach Information Act enacted in California effective July 2003. This act requires that people and companies doing business in the state notify customers of suspected security breaches and provide for civil damages, including financial awards.

**OTHER REMEDIES**

Although applying the notion of due diligence to software security could help reduce the possibility of companies being found liable for security holes, developers and users could also try other approaches.

**Reduce vulnerable features**

Only a few sophisticated users are competent to exploit many vulnerable features in COTS software. Vendors could thus provide a basic software set that does not contain advanced features, such as macros, for use by the average end user. Developers who need more sophisticated features could purchase an alternative version. These users would probably have the know-how to patch their own systems.

**More centralized services**

If consumers used centralized services such as WebTV instead of personal computers, the number of targets for attack would decrease. Many users do not need the myriad features that PCs offer. They only use e-mail, browsers, and a few standard office applications.

I don’t know how likely it is that users will opt for Web-based services, but securing a smaller number of professional systems seems less daunting than securing every single personal computer in use.

**Licensing and certification**

Licensing and certification of software engineers could be viewed as a form of due diligence on the part of companies developing software, especially
COTS products. However, the number of licensed software engineers in the US is small, probably in the double digits. Further, as of October 2003, the IEEE Computer Society has certified only 150 software engineers worldwide as software development professionals. Given this small number, we can’t expect licensing or certification processes to produce enough software engineers who can develop secure software.

Certification processes do exist for security professionals, but they generally apply to systems administrators responsible for securing systems after the fact, rather than software engineers engaged in initial development. Nevertheless, for companies using commercial software, employing certified system administrators could potentially help reduce security holes.

**Software degree programs**

Universities offer many degree programs in software engineering, both graduate and undergraduate, along with several degree offerings in information security. The content of such programs could contribute to ensuring that the programs’ graduates will develop and maintain software that has fewer bugs and better security measures.

Unfortunately, if they cover information security at all, software engineering and computer science degree programs often treat it as an elective topic or lump it with other quality attributes or ethics. The tragedy here is that the vast majority of security holes result from only a few coding errors, such as buffer overflow. Thus, in the long term software development requires attention to security throughout its life cycle, not just after a system is operational.

**COTS risk assessment**

Assessment of the security risks associated with COTS software could be considered a form of due diligence on the part of organizations acquiring COTS software. Typically, red teams and intrusion detection tools help shore up the use of COTS, not risk assessment at the time of acquisition. For example, the @stake Hoover Project examined 45 e-business applications and developed application security profiles for each of them. This study focused on applications for two reasons:

- application-level attacks can easily traverse most firewalls and
- that is where the money is.

The most secure applications in the study contained about one-fourth of the defects found in the least secure applications. This study could provide insights for other application software users and developers. Much more could be done to assess and mitigate security risks associated with application software in general, and COTS software specifically.

Software is like the automobile. Years ago, no one considered suing an automobile manufacturer over injuries resulting from an accident. There was no such thing as a recall. Then Ralph Nader so effectively raised awareness of automotive problems that liability for automobile manufacturers resulted.

A similar epiphany will take place in software. Rather than viewing computers and their associated software as a giant black box, companies and consumers will decide that software should, reasonably, be fit for use at some level. They will also think it reasonable for manufacturers to have shown due diligence.

Therefore, companies and software vendors will try to protect themselves, although what constitutes a reasonable level of due diligence has yet to be determined. Although the courts may not be the best venue for resolving this, liability cases will take place and an evolution of best practices will occur as well. It’s unlikely that 100 percent security will ever be achieved, but significant improvement relative to the status quo likely will occur, with liability cases being just one catalyst.

**References**


Nancy R. Mead leads the CERT Survivable Systems Engineering team at the Software Engineering Institute, Pittsburgh. She is also a senior member of the SEI’s technical staff in the Networked Systems Survivability Program and a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. Mead received a PhD in mathematics from the Polytechnic Institute of New York. She is a senior member of the IEEE and a member of the ACM. Contact her at nrm@sei.cmu.edu.