State Bar of Michigan

Michigan IT Lawyer

A Publication of the State Bar of Michigan Information Technology Law Section

http://www.michbar.org/computer

Table of Contents

May 2011 ▪ Vol. 28, Issue 3

▪ Bits and Bytes from the Chair................1

▪ Save the Date! 2

▪ A Software Liability Policy for Cybersecurity..3

▪ Using Technology to Leverage your Job Search...17

▪ Publicly Available Websites for IT Lawyers..18

▪ 2011 Edward F. Langs Writing Award ..18

▪ Mission Statement Information Technology Law Section, State Bar of Michigan 18

Michigan IT Lawyer is published every other month. Previously published issues of the *Michigan IT Lawyer*, and its predecessor the *Michigan Computer Lawyer*, are available at http://www.michbar.org/computer/newsletters.cfm. If you have an article you would like considered for publication, send a copy to:

Michael Gallo
2700 Renshaw Drive
Troy, Michigan 48085
e-mail: michael@gallo.us.com

Bits and Bytes from the Chair

By Mark G. Malven, *Dykema Gossett PLLC*

In my last Bits and Bytes article I (incorrectly) observed that Spring had arrived here in Michigan. Consider this my "retraction", and let me repeat my wish that you are enjoying our beautiful Michigan Spring weather.

As your Chair my primary goal is that we as Section leaders provide you with valuable events and resources. To that end, I want to note a recent as well as our upcoming activities.

On Thursday April 21 we had our annual Spring Networking Event at The Post Bar in Novi as a joint event with DetroitNET.org, the IT professionals networking organization. A great time was had by all, seeing old friends and making new acquaintances.

Upcoming activities include:

- IT Law-themed edition of the July Michigan Bar Journal.
- Our Fourth Annual ICLE Information Technology Law Seminar on Wednesday, September 21, 2011.
- Our Annual Meeting will be Wednesday, September 21, 2011 during the lunch session of the IT Law Seminar.

Based on the success of the 2010 seminar, the IT Law seminar will once again be presented as an all-day format, from 9 a.m. to 4 p.m., with a cocktail reception following. The venue is the St. John's Inn in Plymouth with a webcast also available, and the topics will be:

- The Keys to Technology Licensing
- Protecting Trademarks Online
- Understanding IT Security
- Software Copyright for Business Lawyers

Continued on next page

- Representing the Web-based Business
- The Legal Implications of Social Media

Registration will open soon, and we look forward to seeing you there this year.

I hope you find the foregoing valuable and, as always, if you have any ideas, suggestions, comments etc. for how we may be of further service to you as a member of the IT Law Section, please do not hesitate to contact me.

Best regards,

Mark Malven

Save the Date!!!

Please plan on joining us on **Wednesday, September 21, 2011**, at the Inn at St. John's in Plymouth for the fourth annual Information Technology Law Seminar. Based on the success of the 2010 seminar, the conference will once again be presented as an all-day format with a webcast also available. Topics will include:

Information Technology Security Issues for Lawyers,

The Keys to Technology Licensing,

Angry Birds and Killer Apps: Software Copyright for Business Lawyers,

Trademark Concerns on the Web,

Representing the Web-based Business, and

The Legal Implications of Social Media.

The Information Technology Law Section of the State Bar of Michigan is the process of finalizing a great lineup of speakers, is working in cooperation with the Institute for Continuing Legal Education (ICLE) to provide continuing legal education credit, and looks forward to seeing everyone there!

If you know of an organization that may be interested in sponsoring the event, please contact *Charlie Bieneman* at cab@raderfishman.com.

Attend the Information Technology Section's ***Annual Business Meeting***. Mingle with Section peers, learn about opportunities to get more involved with the Section, and participate in the election of Section Council Members. The Information Technology Section's 2011 Business Meeting will be held during the lunch session of the Fourth Annual Information Technology Law Seminar. Be there! Please contact *Mark Malven* at MMalven@dykema.com if you have questions about the event. ■

The *Michigan IT Lawyer* is pleased to present "A Software Liability Policy for Cybersecurity" by Daniel Ray. Mr. Ray is one of three student authors to receive a 2010 Edward F. Langs Writing Award from the State Bar of Michigan's Information Technology Law Section. The statements made and opinions expressed in this essay are strictly those of the author, and not the State Bar of Michigan or the Information Technology Law Section. Comments regarding this article can be forwarded to the *Michigan IT Lawyer*, care of *michael@gallo.us.com*. Enjoy!

A Software Liability Policy for Cybersecurity

By Daniel Ray

Introduction: The Status Quo and the Cybersecurity Problem

It is no secret that virtually all software is shot through with bugs, security holes, and incompatibilities, all of which cause innumerable day-to-day headaches and all-too-numerable losses.[1] Simple mistakes in complex software have famously terminated the Mars Climate Orbiter mission,[2] shut down air traffic control systems,[3] and even cost human lives.[4] Compounding the problem, software developers are loath to disclose these faults,[5] and often devote scant resources to fixing them.[6] Nearly every developer protects itself with an end user license agreement that attempts to waive its liability for the harms their software causes,[7] and nearly every court that has considered the question has found their efforts successful.[8] American consumers[9] and IT professionals alike[10] overwhelmingly distrust the quality of the software they use. In short, the state of software quality is "abysmal."[11]

This abysm has hardly swallowed up the industry, however. It is no secret that the software business exploded in size in the last quarter of the twentieth century, and analysts see more gains in the future.[12] This success seems to be based on giving the customer what he wants. If the number of software defects continues to rise, so does the number of software features.[13] New releases are frequent[14] and prices are low.[15] Examining the software market as a whole, then, one might explain the dearth of well-crafted code as the price consumers are willing to pay for products that offer these other enticing characteristics.[16] While credible arguments describe the software market as one of lock-in,[17] inefficient transfer of costs,[18] or adverse selection,[19] developers have to date successfully countered arguments that their behavior constitutes a market failure and requires public intervention.[20]

Crucially, however, both the arguments in favor of market correction and the unsuccessful policies they supported were founded on what this paper will call the "private paradigm."[21] That is, they address themselves to the relationship between the purchaser of flawed software and its developer,[22] and they prescribe remedies for those harms that reduce the value below what the purchaser paid for it and any consequential damages its defects cause her.[23] But there is a growing awareness of a separate "public paradigm": that of cybersecurity. Over the past decade, the American government[24] and academics[25] have paid increasingly close attention to the risks of attacks carried out using the Internet that affect American interests.[26] The cybersecurity problem comprises three major threats. In the first, state actors or independents can directly attack networks in the United States, in order to commandeer them or knock them out completely.[27] In the second, an actor can target specific systems to compromise or destroy non-Internet infrastructure.[28] Finally, attackers may break into systems to remotely access, exploit, or modify government or private data.[29] In all these scenarios, determining who is responsible for the attack can be extraordinarily difficult or literally impossible,[30] and taking action against even a suspected foreign wrongdoer can have major implications.[31] Strategically speaking, then, defense is preferable to retaliation. By definition, a remote attack against a computer requires a vulnerability, and these vulnerabilities are almost always traceable to software.[32] For this reason, software quality matters for the public paradigm as well as the private one.

The cybersecurity problem presents more reasons to favor public intervention than do traditional, private concerns about flawed software. The first is cybersecurity's inherent externalities. To be sure, software purchasers are the primary victims of hacking attempts targeted at their computers. But once a malicious program has breached one computer, it will have an easier time infiltrating other computers on its network,[33] even if their owners have purchased software without Internet-facing vulnerabilities. Moreover, a compromised computer can be used as to attack other targets without its owner's knowledge. This has led to a breed of attacks that first tries to commandeer thousands or millions of computers indiscriminately, thereby assembling a legion of "zombies." The attacker then deploys the zombie computers against his intended target, all the while ensuring that each zombie computer's performance is not so degraded that an average owner would detect and remove the malware.[34] Finally, a third externality, less remarked upon by the cybersecurity literature, concerns the value of the Internet itself. Because a network's worth comes from the connections between its nodes,[35] an attack that succeeds in knocking a node (or component network of nodes) offline will have harmed not only the owners of those nodes, but also those who remain on the Internet, unable to access the missing nodes.[36] These issues present textbook negative externalities justifying public intervention: while software purchasers will seek software that avoids costly injuries to their own interests, they are (rationally) far less interested in preventing injury to strangers.[37] Under existing laws, victims of these attacks cannot easily remedy the harms of their attacks: the attribution problem[38] and an extraordinarily low rate of prosecution combine to make the Computer Fraud and Abuse Act[39] (CFAA) unavailing,[40] and the existing proposals to hold software developers liable for their bugs will not aid parties not part of the contract between developer and purchaser.[41]

The Internet's status as a network also underlies a second classic market failure: the public goods problem. The externalities referenced above do not stop at threatening other private citizens' computers, or even other businesses' IT systems. Instead, today's physical infrastructure is vulnerable to Internet insecurity. The protection of the nation's power grid,[42] its water and gas pipelines,[43] and other important infrastructure is of vital importance, but it is also subject to free-riding. [44]

This paper suggests a method of responding to these problems and controlling the risks of software vulnerabilities to national security. Part II describes, in outline, a proposal to reduce the number of important vulnerabilities in software by instituting a liability regime for software developers and encouraging a market for insurance against the same. Part III offers two important considerations for evaluating marginal increases in the policy's effectiveness: first, traditional economic costs borne by participants in the software market, and second, inhibitions on innovation that are, in a sense, borne by the Internet as a whole. Part IV examines a number of important aspects of the proposal and suggests how they might be resolved. In particular, it asks what legal test should be applied to determine whether a developer is liable, how the law should treat open-source software, and how damages for liable developers should be calculated. Finally, Part V concludes.

A Model of Software Liability

Which model of liability?

At base, this paper suggests that the federal government mitigate these systemic problems by requiring software developers internalize more of the costs of their security decisions. Imposing liability for software flaws is hardly a new idea, but existing proposals appear to focus exclusively on the private paradigm. [45] Due to its focus on the public paradigm of cybersecurity, this paper proposes allowing a party harmed by an attacker who has exploited a software vulnerability that is relevant to cybersecurity to recover damages from the party responsible for introducing that vulnerability.

Dozens of questions spring immediately from such a broadly worded policy, and this paper responds to several of them below. But one question preempts all others: under what legal theory should this liability be imposed? By the beginning of the twentieth century, claims against manufacturers for flaws in their goods have sounded in negligence,[46] and in its second half plaintiffs have also succeeded on the theory of product liability.[47] Between the two, products liability is typically the preferable theory from the plaintiff's perspective, because unlike negligence it does not demand a showing of unreasonableness in the defective product's design or manufacture.[48] Like so many concepts in the common law, however, neither negligence nor products liability comfortably accommodates computers and the Internet, and the particular challenges of public paradigm cybersecurity only amplify these difficulties. To name only a few doctrinal barriers to recovery,[49] both theories traditionally reject claims for purely economic losses, as many claims are likely to be.[50] Negligence doctrine forbids recovering for losses caused by another's crime,[51] while a plaintiff pleading products liability must first convince a court that the defendant's software is actually a "product."[52] These hurdles are in addition to the private[53] and statutory[54] laws that have also helped software developers to date avoid even one adverse judgment arising out of an attack.[55] Contemplating all of the above, this paper embarks from the premise that a new law is in order.

The Force Multiplier: Liability Insurance

Once developers are made to internalize some of the costs of their software's vulnerabilities, they will seek insurance. Liability insurance spreads the costs of an adverse judgment among many other developers, which in turn should help to prevent any one firm from collapsing under the weight of an adverse judgment. More importantly, insurance will also help to secure software and decrease the cybersecurity risk borne by the public. This is, of course, because insurers charge lower premiums to less risky policyholders.[56] Over time, new standards coalesce[57] or existing ones get new teeth.[58] While this effect is seen in most or all industries,[59] liability insurance's cost-spreading function is especially useful in the software context. If a software developer is held liable for their vulnerabilities but cannot purchase insurance, a single judgment could push it into bankruptcy. This would hardly enhance cybersecurity: while it would likely halt future sales of the vulnerable product, it would all but ensure that existing copies go unpatched. At the same time, the security flaw that led to the plaintiff's suit has by definition been published, and data on the exact risk profile it has caused may very well be part of the public record. For this reason, insurance is not merely a useful, but in fact a necessary, part of this proposal.

Fortunately, a measure of cybersecurity insurance already exists.[60] Responding in part to HIPAA and the Gramm-Leach-Bliley Act[61] and in part to the increasing importance of networked data to business, the first such policies were written in the around the time of September 11.[62] Unfortunately, the rate of coverage is still low,[63] as firms deem security spending a costly indulgence in bad economic times.[64] This is a reasonable decision; firms understand that their liabilities under today's law are a function of their business (location of servers, number of customer records stored, etc.). Placing liability on culpable software *developers* rather than on software *users* will help to ensure that insurance decisions also improve security for the externalities discussed above. The goal, then, must be to encourage firms to offer new liability insurance, modeled after their nascent "cyberinsurance" policies. The law should introduce as few new variables as possible into the risk equation.[65] This means a well-specified liability regime that gives considerable weight to measures that avoid uncertainty, even at the cost of some close tailoring of remedy recovered to harm suffered.[66]

Evaluating the Policy: Economic and Generativity Costs

There are other important characteristics by which one can evaluate a software liability policy. One, of course, is the economic costs it imposes on society. It goes without saying that a producer of goods, newly saddled with liability it has previously avoided, will seek to raise its prices. Indeed, many critics of earlier software liability proposals have based their arguments on this point.[67] It is important to remember, however, that raising prices is precisely what liability law is meant to do: by driving unsecure producers out of the market, the range of software available for consumers becomes safer.[68] Because this policy is motivated at base by national security, it assumes that some cost is worth bearing. Still, part of what has made the Internet such a dynamic part of society is its low costs to entry.[69]

Another social cost goes less remarked upon, though it is just as implicated by a law that would change the software industry. Professor Jonathan Zittrain has applied the term "generativity" to the principle of user-initiated innovation and disruption, and he considers generativity to be the foundation of the Internet as it exists today.[70] A generative platform — like a personal computer, but unlike an iPod — is not designed with a single purpose in mind. Instead, it allows users or third parties to access its components and simply runs the code it is given.[71] Closed applications and tethered appliances are as popular as ever among today's consumers.[72] Part of these products' attraction is their user experience: a gatekeeper can reduce the prevalence of buggy or vulnerable software for a device,[73] and to the end consumer, this security means that the product "just works."[74] Unfortunately, as Professor Zittrain and others argue, generative endpoints are vital to the Internet's continued vitality.[75] Returning to the issue at hand, then, a proposal for software liability yields generativity costs to the extent it encourages software developers to evade security threats by moving from generative to closed products.

This paper operates from the uncontroversial assumption that the Internet's status quo is worth protecting. Security that would destroy the object it protects is no security at all. Just as in the wider and ongoing debate surrounding the United States' response to terrorism,[76] designers of an effective cybersecurity policy must bear in mind the values they wish to protect. And just as in the wider context,[77] cybersecurity policymakers must weigh its benefits against its costs.[78] To be sure, the effects are not unidirectional: a safer Internet ecosystem will reduce the drive to lock down platforms,[79] and more secure operating systems will reduce the need for costs for individual protective measures.[80] But if ignored, either price or generativity could undermine the effectiveness of the policy this paper proposes, and they will therefore be the principal criteria by which this paper evaluates the solutions it suggests for the problems discussed below.

Policy Questions

What is the proper standard for secure software?

The discussion of existing law above concluded that neither negligence nor product liability is a perfect fit for enhancing cybersecurity. Yet even the federal statute proposed here must decide between the models: should taking due care in developing a piece of software save its developer from liability for any vulnerabilities it nevertheless contains? For a number of reasons, this paper concludes that it should. The question is at base one of relative values: how much weight does society place on cybersecurity risks versus the costs identified above? Strict liability would enhance security to a greater degree than negligence, and under this piece's consistent assumption this will mean fewer successful attacks. Thus, if (at the extreme) society demanded total protection against cyber risks, strict liability would be the correct choice, since negligence would leave the cost of avoiding some software vulnerabilities out of the software's price, resulting in excessive risk-taking.[81] Of course, even if it is possible to build perfectly secure software, doing so would not mean perfect cybersecurity.[82] Fortunately, society's preferences are certain to be more moderate — thus, they will require a tradeoff. While a formal model would be useful, there is reason to think that in practice, even a society whose values tilt significantly in favor of cybersecurity should prefer negligence. This is due to the way that the new law would take effect. Assuming (as is sensible) that liability would only attach prospectively, software developers under a negligence system would work quickly to certify their projects as compliant with the security standards that would naturally represent due care. So long as the standards were made clear before the law took effect, this would be an expensive but a delimited task. Under a strict liability system, on the other hand, developers may well choose to withhold their upcoming products, confident that their competitors would be sued out of existence the moment they shipped their own. If this situation comes to pass, the software market becomes a game of attrition, with socially useful software essentially held hostage by the firms with the greatest development budgets. Even if developers do not react to the new law in this way, the flood of "zero day" litigation that will greet a strict liability regime would significantly impede the law's objectives. The damages awarded to plaintiffs with valid claims would immediately overwhelm the young liability insurance market, while barraters' overlapping suits will clog dockets and produce a tangle of conflicting precedent. A system of negligence, on the other hand, would produce far fewer initial suits, as developers will hasten to advertise their compliance with published standards. For these reasons, this paper concludes that the law should use a negligence standard, and rely on other, more yielding variables to tune its tradeoff between costs and effectiveness.

Of course, settling on negligence for flaws only invites another question: for which flaws does the standard apply? It is tempting to provide a cause of action for private paradigm harms as well as public ones. Certainly, all software has bugs, and providing a legal remedy for them has been the object of reformers for more than thirty years.[83] As discussed above, however, there are colorable arguments that, as the software market has matured, it has settled on an efficient level of quality relative to price, features, and the like.[84] Even if this supposition is incorrect (and there are certainly reasons to dispute it[85]), new liability for harms that affect only the purchaser or user of software would distract developers from improving the *security* of their products (thereby preventing the risks to third parties that are the public paradigm's gravamen) if efforts to fix the two types of bugs are distinguishable. At the lowest level, they certainly are: software testers uncover problems in software, and programmers fix them. A given bug report might complain, for instance, that the software's user interface behaves unexpectedly,[86] or that certain external input will cause a crash with unsecured memory.[87] Clearly, the first report implicates the private paradigm; the second, the public. Even at a higher conceptual level, however, security vulnerabilities will almost always be recognizable as such. Unlike flaws in a program's functionality, whose character and severity depend on individual applications' design and specification, security flaws fall into a small number of discrete categories wholly independent of application or developer.[88] All of this means that a system of liability ought to be able to target security vulnerabilities without including less dangerous bugs as well.[89] Because targeting these private paradigm harms would impose economic and generative costs on society without directly contributing to cybersecurity, this proposal should not address them.[90]

Adding another element to the statutory cause of action will also help to prevent inefficient litigation: that the plaintiff have suffered an actual attack and incurred actual losses. This need not necessarily be a part of a risk-based liability regime — plaintiffs have been recovering since the 1980s for exposure to toxic substances without alleging actual loss,[91] for instance. In the software context, however, proving that an actual attack has occurred will be especially useful. This acts as a prioritization mechanism, to ensure that the firms held liable are those whose vulnerabilities have become moved from theoretical to real-world threats.[92] The requirement will also control the volume of litigation, preventing a race to the courthouse at the moment the law takes effect and maintaining today's systems for responsible vulnerability disclosure.[93] While it is unrealistic to imagine that the courts could act to neutralize the risk of individual vulnerabilities after they are announced but before they are exploited,[94] focusing on the most dangerous vulnerabilities should be enough to encourage all developers to keep these bugs out of their software.

How does open-source software fit in?

This paper has previously argued that the motivations behind and debates over software liability for the private paradigm differ fundamentally from a liability regime that is meant to enhance cybersecurity. However, many of the discussions surrounding the earlier proposals are instructive. Perhaps the greatest bone of contention among reformers was how the new policy would treat open source software. Definitions, philosophies, and even competing names for the open source phenomenon abound.[95] For present purposes, however, a technical description will suffice: open source software is distinct from proprietary, or "closed source" software, in that its source code is distributed freely and licensed such that anyone can improve it.[96] The volume of software that shares these twin characteristics is growing at an exponential rate,[97] and the total volume of code made available under open source licenses has doubled every thirteen months.[98] With software like Apache (representing 53.84% of web servers online[99]), MySQL (used to manage over eleven million databases[100]), and PHP (by one measure the most popular web scripting language[101]) so widely deployed, open source software is critical in assessing cybersecurity vulnerabilities.

Of course, it is not for nothing that open source software receives special attention when discussing a system of liabilities, for private harms and public alike. The problem in bringing suit against an open source project is immediately evident: who is legally responsible for its development, and can that entity be deterred by standard tort law? The largest open source projects do indeed have formal, legal structures (and, presumably, the legal teams this level of organization suggests).[102] A chart plotting the entire population of open source projects against the number of their contributors (or their users) would have a very long tail, however,[103] and the vast majority of these smaller projects have no distinct legal status at all. Attempts at targeting individual contributors of flawed code, too, would be futile with respect to these projects. The majority of open source contributors do so for free and lack anything like assets sufficient to cover the damage their code could cause;[104] despite some projects' stated policies,[105] some contributors submit their code under a pseudonym precisely to avoid personal liability.[106] There is a second side to the practical problem, as well. Even if a project or the contributor of its faulty code is "deterrable" (that is, formally amenable to suit and possessed of assets sufficient to pay a judgment), it may still be immensely difficult to prove that the project has done something worth deterring in the first place. Although sophisticated versioning and bug tracking systems are increasingly available and are widely used,[107] few open source projects can match the documented, institutional controls that exist in even smaller commercial software development houses.[108] While such policies are not strictly necessary to the production of secure code (they are certainly not sufficient), they would be much easier to prove in a negligence proceeding.

This practical difficulty in part informs the call, heard loudest from those in the open source community themselves, that open source software should remain immune from liability.[109] Fundamentally, they argue, open source software is more secure than closed source software for the same reason that research documented in *Nature* is more trustworthy than research documented on a blog: it benefits from peer review.[110] A maxim attributed to Linus Torvalds, the architect of Linux, holds that "given enough eyeballs, all bugs are shallow."[111] Of course, this does not mean that open source software is impeccable. As an initial matter, there are some kinds of flaws that even the world's finest programmer cannot discern by scrutinizing source code. In a famous address, computer science pioneer Ken Thompson described one such bug, which allowed him to bypass the user authentication program on any Unix system.[112] Moreover, in practice even "visible" bugs can be all but undetectable, especially if this is part of their author's goal.[113] Nor does every project have the number of eyeballs its lead developers (or Mr. Torvalds) think it deserves. If there is an "efficient market" for contributions to open source projects, more important projects will have more contributors. The factors contributors consider when choosing a project (by available accounts) do not perfectly align with those that determine a project's relevance to the cybersecurity problem.[114] It is also not true in every case that the collaborative development model avoids the market pressures that encourage proprietary software developers to ship buggy products. While many open source projects are content to track down flaws and release new versions at a deliberate pace, others' market share demands maintaining a release schedule in parallel with a more popular proprietary application.[115] Likely more common still are those open source projects that, even without strong pressures from competing, proprietary products, bind their hands by announcing target release dates.[116]

Despite all of these objections, the open source partisans have a point. At worst,[117] open source software as a category is not observably less secure than closed source software.[118] While the debate rages between the "more secure" and "less secure" camps, too many other factors affect risk[119] for the availability of a program's source code to significantly affect the final outcome. For the reasons discussed, however, there is ample reason to think that it will be more difficult than proprietary software to cover effectively with the kind of liability system described herein. Therefore, exempting open source software from liability would mean selecting, from a pool of equally risky items, a set of items whose risk (on average) will be more expensive than average to mitigate. As with the other aspects of the policy that pit effectiveness against economic and generativity costs, the social efficiency of this trade depends on the weight given to cybersecurity — that is, what level of risk society is willing to tolerate. If society's tolerance is nil, then every flawed piece of software must be caught and corrected, no matter what the cost. If, on the other hand, society is willing to strike a balance between cybersecurity, software prices, and the generative Internet, immunizing open source software appears to be a very sensible tradeoff. Categorical immunity will let open source continue to compete with proprietary software, thereby limiting the latter's ability to raise its prices and preserving a buffer to offset the losses to generativity from those proprietary developers who pursue security by locking down their software against legitimate third-party uses. At the same time, while immunity will certainly change the calculus for firms choosing how they will develop their products,[120] any competitive costs proprietary developers sustain versus their open source competitors will be at least partially offset by their amenability to suit for losses.[121]

How should damages be measured?

The damages available under the liability system proposed here are its most important component. If, as recommended above, the statutory cause of action requires an actual attack, it is only just that the successful plaintiff receive compensatory damages. Because this proposal's purpose is deterrence, however, the damages it prescribes should be scaled toward that goal. Similarly, because insurance is so important to the scheme, and because insurance abhors unpredictable risks, a firm's liability exposure should be determinate even before it is sued. These two requirements inform the recommendation that the statute provide a schedule of damages[122] that varies in proportion to the risk the vulnerability imposes on society.[123]

How should the schedule correlate recovery with risk? The degree of threat a given software vulnerability poses is a function of many variables.[124] Attempting to factor all of these into the schedule would quickly produce an unwieldy, unpredictable mechanism, and therefore undercut its very purpose. For this reason, this paper proposes using the number of copies of the software in current use on computer systems in the United States as a rough but easily determined index. A program's installation base tracks its total security risk closely because the marginal cost of expanding an attack is minimal. Once a vulnerability is discovered and exploited, the damage it can cause depends in large part on the number of machines the exploit can target.[125] By scaling the amount a negligent developer can expect to pay in damages to the market share it seeks, the proposal avoids a point raised by the "monoculture" argument's detractors,[126] that mandating an assortment of competing applications will place a high burden on software users.

To be sure, awards based on the installation base of the culpable software could be quite large. To take perhaps the most extreme example from the range of consumer-level software, Windows XP was in use on more than four hundred million computers late in its lifecycle.[127] This enormous multiplier does not necessarily mean that the awards will fail judicial review, however.[128] Instead, just as with the other policy aspects discussed herein, the tremendous potential damages awards is important insofar as it threatens to deter even welfare-enhancing software. Law and economists' suggestions about "optimal deterrence" (whereby the expected harm caused by a defendant's act is divided by the probability of detection and prosecution)[129] are inapposite here, as it is in general the *most* detectable bugs that pose the greatest risk of exploitation.[130] It is far more parsimonious to confront the problem directly. In the current proposal, over-deterring recoveries could come from outlying plaintiffs. For instance, while financial institutions are important parts of the information infrastructure, their extraordinary exposure to losses from cybersecurity risks gives them far greater than average incentives to sue under this cause of action than other businesses or individuals. Therefore, it is sensible to cap the portion of compensatory damages that serve as the multiplicand for the installation base multiplier. If the optimal cap is low enough, the scaled damages' multiplicand may be better cast simply as a fixed fine.[131]

When considering how to scale damages based on installation base, it is important to address the issue of unlicensed software. A software industry trade group estimates that twenty percent of PC software is pirated.[132] While the BSA's numbers are widely and justly disputed,[133] the share of unlicensed software is almost certainly enough to make a significant difference to firms attempting to calculate their liability exposure. Unfortunately, while an unlicensed copy of an application poses precisely as much cybersecurity risk as their legitimately purchased counterparts, extending the calculation of damages to cover them could incur special costs. The effect of including unlicensed software would be to impose on developers significantly higher costs per pirated copy — not merely a lost sale, but a liability many times in excess of a license's selling price. This enhancement could push rational developers to impose severe and limiting copy protection schemes on their products. These policies would have a deleterious effect on the Internet's generativity.[134] This is another area, then, that merits further research: how much of the average vulnerable program's installation base would be ignored by a rule that does not account for unlicensed software?

Conclusion

Like the software it has criticized, the system that this paper has proposed is anything but elegant. Like the software it has criticized, it is certain to include a number of flaws that will compromise its effectiveness and therefore need to be resolved in future iterations. The liability system itself is probably not even ready for pre-release, since it has left fundamental questions of design unanswered pending more research.[135] But, like the software it has criticized, this paper's proposal has a use despite its flaws. It recognizes that today's Internet may well exist in a fortunate but fragile bubble. If the volume of scholars who predict an increasing public risk from foreign attacks, professional cybercriminals, and others are correct, the status quo is not sustainable. On the other hand, it recognizes that heavy-handed policies could be as dangerous as none at all. If nothing else, the proposal should show that there is enough flexibility in the idea of holding software developers liable for the risks they impose to strike a balance between these competing interests. A liability system cannot solve the problem of vulnerable software overnight, but the sooner it is designed and implemented, the sooner it will begin to help. ■

Endnotes

1 *See* Nat'l Inst. of Standards & Tech., The Economic Impacts of Inadequate Infrastructure for Software Testing 5-15, 8-1 (2002), *available at* http://www.nist.gov/director/planning/upload/report02-3.pdf (calculating that the ineffective tools used in software testing cost commercial users $38.3 billion annually, in the form of labor, failure, lack of expected performance, redundancy, delayed profits, and forfeited sales); *see also id.* at 8-7 (noting that the estimated cost to users does not include costs to "residential households" and that the estimate likely undercounts the costs to the public sector).

2 Robin Lloyd, *Metric Mishap Caused Loss of NASA Orbiter*, CNN.com, Sept. 30, 1999, http://www.cnn.com/TECH/space/9909/30/mars.metric.02/ (faulting a miscommunication between two engineering teams, one using English units of measurement and the other using metric units).

3 Linda Geppert, *Lost Radio Contact Leaves Pilots On Their Own*, IEEE Spectrum, Nov. 2004 (faulting a countdown timer that triggered a buffer overflow and shut down upon reaching zero).

4 *See, e.g.*, Nancy Leveson, SafeWare app. A (1995) (describing the Therac-25 medical linear accelerator, which killed two patients with massive overdoses radiation because the software controlling it did not check whether safety mechanisms were in place before activating its high-power X-ray).

5 *See* Rahul Telang & Sunil Wattal, Impact of Software Vulnerability Announcements on the Market Value of Software Vendors — an Empirical Investigation (unpublished manuscript, Feb. 2005), *available at* http://www.heinz.cmu.edu/research/280full.pdf (measuring the negative effect of vulnerability announcements on software firms' stock prices).

6 *See* Ashish Arora, Jonathan P. Caulkins, & Rahul Telang, Sell First, Fix Later: Impact of Patching on Software Quality (unpublished manuscript, Oct. 2004), available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=670285.

7 *See* Florence Marotta-Wurgler, Competition and the Quality of Standard Form Contracts: An Empirical Analysis of Software License Agreements 10, 11, tbl. 4 (unpublished manuscript, Aug. 22, 2005), *available at* http://ssrn.com/abstract=799274 (concluding, from a sample of 647 license agreements from "almost all well-known software publishers as well as . . . hundreds of smaller companies," that 89% disclaimed consequential, special, incidental, and special damages, that 72% disclaimed foreseeable losses, and that 34% waived damages under all legal theories).

8 *See, e.g.*, i.Lan Sys., Inc. v. Netscout Serv. Level Corp., 183 F. Supp. 2d 328, 334–39 (D. Mass 2002) (upholding a clause in an end user license agreement purporting to limit liability and disclaim all implied warranties); Mortenson Co. v. Timberline Software Corp., 998 P.2d 305 (Wash. 2000) (en banc). (upholding a term disclaiming liability for consequential damages). In general, courts will uphold end user license agreements "unless their terms are objectionable on grounds applicable to contracts in general (for example, if they violate a rule of positive law, or if they are unconscionable)." ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1449 (7th Cir. 1996); *see also* Nathan J. Davis, Note, *Presumed Assent: The Judicial Acceptance of Clickwrap*, 22 Berkeley Tech. L.J. 577 (2007).

9 *See* Edsger W. Dijkstra, *The End of Computing Science?*, 44 Comms ACM 92 (2001) ("The average customer of the computing industry has been served so poorly that he expects his system to crash all the time, and we witness a massive worldwide distribution of bug-ridden software for which we should be deeply ashamed.").

10 *See, e.g.*, Paul Thurrott, *Report: Most Users Do Not Trust Microsoft*, Windows IT Pro, Apr. 1, 2003, *available at* http://windowsitpro.com/article/articleid/38543/report-most-users-do-not-trust-microsoft.html (citing a study showing that seventy-seven percent of surveyed professionals distrust the quality of Microsoft's products).

11 Shari Lawrence Pfleeger, *Solid Software: Is It Rocket Science?*, *in* Software Quality — ECSQ 2002 at 7, 7 (Jyrki Kontio & Reidar Conradi eds., 2002).

12 *See Forrester sees global tech spending rebound in 2010*, Reuters.com, Jan. 12, 2010, http://www.reuters.com/article/idUSTRE60B2ER20100112 (predicting that global software sales will grow by 9.7% in 2010).

13 The market pressure to add features, and the phenomenon of "software bloat" it can yield, is memorably captured by Jamie Zawinski's Law of Software Envelopment: "Every program attempts to expand until it can read mail. Those programs which cannot so expand are replaced by ones which can." Jwzhacks, http://www.jwz.org/hacks/ (Last visited Jan. 17, 2009).

14 The software industry exerts considerable pressure to release new products and revisions early and often. *See* Steven R. Rakitin, *Balancing Time to Market and Quality*, Software Quality Professional, June 1999, at 54, *available at* http://www.swqual.com/images/Balancing.pdf.

15 Hedonic pricing models show that real (that is, quality-adjusted) software prices consistently fall over time. *See* Neil Gandal, *Hedonic Price Indexes for Spreadsheets and an Empirical Test for Network Externalities,* 25 RAND J. Econ. 160 (1994) (finding an average 15% decline in real prices of spreadsheet software every year from 1986 to 1991); Sendil K. Ethiraj et al., Determinants of Price in Custom Software: A Hedonic Analysis of Offshore Development Projects (unpublished manuscript, June 2004), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=569875 (average 14% decline per year in real prices of custom software).

16 The fact that price and features trade against quality has not gone unnoticed in American culture. Consider the popular joke that posits a conversation between the heads of Microsoft and General Motors. Bill Gates says that if GM could only build cars like Microsoft built software, they'd cost $25 and get a thousand miles to the gallon. His opposite number responds drily, "Sure, but who would want a car that crashes three times a day?"

17 *See* Ross Anderson, Security Engineering 221–23, 860 (2d ed. 2008); *see also* Jonathan L. Zittrain, The Future of the Internet and How to Stop It 176–77 (2008).

18 *See* Nat'l Inst. of Standards & Tech., *supra* note 1, at 1-12 to 1-13 & tbl. 1-5 (showing that the marginal cost of fixing a flaw increases enormously if done when the software is on the market rather than when it is still in development).

19 *See, e.g.* David Rice, Geekonomics: The Real Cost of Insecure Software 23–24 (2007).

20 *See, e.g.*, Letter from the Linux Foundation and Microsoft to Prof. Robert A. Hillman, Dean Maureen A. O'Rourke, & Prof. Lance Liebman (May 19, 2009) (available at http://pdfserver.amlaw.com/ca/soft0721.pdf) [hereinafter Linux-Microsoft Letter] (disputing the necessity of a non-disclaimable warranty of no known, hidden, material defects in the ALI's draft Principles of the Law of Software Contracts); *cf.* Business Software Alliance, President Obama Opposes Mandating Security Standards (May 2009), http://global.bsa.org/cybersecuritydashboard/news/dictate1_may09.html ("[W]e support the President's stance against mandates of country-specific, government-created security standards, which would be sharply at odds with the global scale of the Internet and of the technology ecosystem").

21 This paragraph refers especially to three successive attempts at regulating software quality. In the early 1990s, the American Law Institute moved to amend the Uniform Commercial Code with what would become Article 2B. *See* David A. Rice, *Digital Information as Property and Product: U.C.C. Article 2B*, 22 U. Dayton L. Rev. 621, 627–28 (1997). When the ALI withdrew Article 2B from consideration in 1999, following widespread criticism from independent developers and consumer groups, the National Conference of Commissioners on Uniform State Laws renamed the proposal the Uniform Computer Information Transactions Act and sought to have it enacted at the level of the fifty states. James S. Heller, *UCITA: Still Crazy After All These Years, and Still Not Ready For Prime Time*, 8 Rich. J.L. & Tech. 5, 33 (2001). Only two states adopted the UCITA (four actively sought to prevent its application to their citizens), and in 2004 the ALI began work on a new project, the Principles of the Law of Software Contracts. *See* Maureen A. O'Rourke, *An Essay on the Challenges of Drafting a Uniform Law of Software Contracting*, 10 Lewis & Clark L. Rev. 925, 926 (2006). Unlike its conceptual predecessors, the Principles do not seek to bind courts directly, and they admit that they do not reflect settled law. *Id.* In light of the criticism the Principles have received already, *see, e.g.,* Linux-Microsoft Letter, *supra* note 20, courts may be wary of relying on them.

More recently, the European Commission has pressed for software liability. *See* To the Espiner, *EC Wants Software Makers Held Liable For Code*, ZDNet UK, May 8, 2009, at http://news.zdnet.co.uk/software/0,1000000121,39649689,00.htm. The European proposal, too, focuses chiefly on the private paradigm of software quality. *See* Press Release, European Commission, Consumer Rights: Commission Wants Consumers to Surf the Web Without Borders, *available at* http://europa.eu/rapid/pressReleasesAction.do?reference=IP/09/702 ("Licensing should guarantee *consumers* the same basic rights as when they purchase a good: the right to get a product that works with fair commercial conditions."(emphasis added)).

22 All three of the proposals discussed above were drafted in response to, and in the terms of, contract law. *See, e.g.*, Press Release, National Conference of Commissioners on Uniform State Laws, UCITA Facts (Mar. 14, 2003) [hereinafter "UCITA Press Release"], *available at* http://www.nccusl.org/nccusl/ucita/UCITA_Facts.pdf ("UCITA rejects radical concepts. It incorporates traditional principles of freedom of contract."). For this reason, all assume privity between harmer and harmed. *See, e.g.*, U.C.I.T.A. § 113(a) (The effect of any provision of this [Act], including an allocation of risk or imposition of a burden, may be varied by agreement of the parties." (alteration in original)).

23 U.C.I.T.A. § 809(a) (2000).

24 *See* U.S. Dep't of Homeland Sec., The National Strategy to Secure Cyberspace vii (2003) [hereinafter National Strategy] ("Of primary concern is the threat of organized cyber attacks

capable of causing debilitating disruption to our Nation's critical infrastructures, economy, or national security.").

25 *See, e.g.*, Susan W. Brenner & Marc D. Goodman, *In Defense of Cyberterrorism: An Argument for Anticipating Cyber-Attacks*, 2002 J.L. Tech. & Pol'y 1 (2002).

26 The first decade of the twenty-first century also saw the first large-scale Internet attacks. In 2007, Estonia saw widespread attacks on its government's Internet connections and its private financial systems. *See* Edward Skoudis, *Information Security Issues in Cyberspace*, *in* Cyberpower and National Security 171, 177–78 (Franklin D. Kramer, Stuart H. Starr, & Larry K. Wentz eds., 2009). In 2003, an entity launched a number of attacks meant to expropriate sensitive information from several critical military agencies. The Department of Defense believes that the attacks originated in China. *See* Clay Wilson, *Cyber Crime*, *in* Cyberpower and National Security 415, 424, *supra*.

27 *See* Franklin D. Kramer, *Cyberpower and National Security: Policy Recommendations for a Strategic Framework*, *in* Cyberpower and National Security 3, 16, *supra* note 26. These attacks include those that target infrastructure critical to the Internet itself.

28 *See* John A. McCarthy et al., *Cyberpower and Critical Infrastructure Protection: A Critical Assessment of Federal Efforts*, *in* Cyberpower and National Security, *supra* note 26, at 543.

29 *See* Wilson, *supra* note 26, at 423–24 (discussing espionage against government secrets); Ellen Messmer, *Cyber Espionage: A Growing Threat to Business*, PC World, Jan. 21, 2008, *available at* http://www.pcworld.com/businesscenter/article/141474/cyber_espionage_a_growing_threat_to_business.html (against private secrets); Linda Tucci, Electronic medical records at *risk of being hacked, report warns*, SearchCIO.com, Sept. 19, 2007, *available at* http://searchcio.techtarget.com/news/article/0,289142,sid182_gci1273006,00.html (discussing modification and corruption of medical records).

30 Indeed, the "attribution problem" is a fundamental obstacle in cybersecurity. More than covering his tracks, a skilled attacker can frame unwitting intermediaries. *See* Jeffrey Hunker, Robert Hutchinson, & Jonathan Margulies, *Attribution of Cyber Attacks on Process Control Systems*, *in* Critical Infrastructure Protection II 87, 88–93 (Mauricio Papa & Sujeet Shenoi eds., 2008). *But see* Richard L. Kugler, *Deterrence of Cyber Attacks*, *in* Cyberpower and National Security, *supra* note 26, at 309, 309–310 (arguing that adversaries behind the most significant attacks will generally be willing to attack openly).

31 "In the event of a massive cyberattack against the country that was perceived as originating from a foreign source, the United States would consider launching a counterattack or bombing the source of the cyberattack*.*" Ellen Messmer, *U.S. Cyber Counterattack: Bomb 'Em One Way or the Other*, Network World, Feb. 8, 2007, *available at* http://www.networkworld.com/news/2007/020807-rsa-cyber-attacks.html?page=1 (paraphrasing Mark Hall, co-chair of the Department of Defense's National Cyber Response Coordination Group, in an address given in the wake of a significant distributed denial of service attack against the DNS system).

32 *See generally* Ross Anderson, *supra* note 17, at 117–26, 635–52; Ivan Krsul, Software Vulnerability Analysis (unpublished Ph.D. dissertation, May 1998), *available at* http://ftp.cerias.purdue.edu/pub/papers/ivan-krsul/krsul-phd-thesis.pdf.

33 *See* Geoffrey Heal & Howard Kunreuther, You Only Die Once: Managing Discrete Interdependent Risks 2 (unpublished manuscript, June 2003), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=419240.

34 *See* Evan Cooke, Farnam Jahanian, & Danny McPherson, The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets (unpublished paper presented at 2005 Steps to Reducing Unwanted Traffic on the Internet Workshop, July 7, 2005), *available at* http://www.usenix.org/events/sruti05/tech/full_papers/cooke/cooke_html/; Rick Wash & Jeffrey K. MacKie-Mason, Security When People Matter: Structuring Incentives For User Behavior 2 (unpublished paper presented at 2007 International Conference on Electronic Commerce, August 19–22, 2007), *available at* http://hdl.handle.net/2027.42/60131.

35 This principle was most famously stated in Metcalfe's Law, which states that the value of a network increases in proportion to the square of the number of its users. *See* Bob Metcalfe, *Metcalfe's Law: A Network Becomes More Valuable As It Reaches More Users*, InfoWorld, Oct. 2, 1995. This "law" has been criticized as optimistic, since most users will not try to connect to the missing nodes and therefore not miss out on an expected benefit, at least directly. Beckstrom's Law is meant to account for this fact by modeling actual transactions on the network. *See* Rod Beckstrom, A New Model for Network Valuation (unpublished manuscript, Mar. 13, 2009), *available at* http://valuenetworks.com/docs/Beckstroms_%20Law.pdf.

36 Even under Beckstrom's more parsimonious valuation, theses harms need not be insignificant. Even the temporary loss of a major email or VOIP provider, for instance, could have serious economic consequences. The rise of "cloud computing," which transfers software and data traditionally stored locally on user's own hardware to the Internet, only raises the stakes. *See* Jonathan Zittrain, Op-Ed., *Lost in the Cloud*, N.Y. Times, July 19, 2009, at A19.

37 *See* Steven Shavell, *Strict Liability Versus Negligence*, 9 J. Legal Studies 1, 12–17, 20–22 (1980).

38 *See supra*, TAN 30.

39 18 U.S.C. § 1030 (2008).

40 Rice, *supra* note 19, at 73 ("Approximately 5 percent of cyber criminals are caught."); *see also id.* at 329 n.1 (discussing other estimates).

41 *See* notes 21 & 22, *supra*. Even if attribution was not such an obstacle, the privity requirement in the proposals outlined above render them neatly applicable to crashes, corruption, and data loss (all of which harm either the software licensee herself, or one with whom she has contracted), but much less so to the intentional attacks (which are carried out by one third party and can harm several others) with which this paper is concerned.

42 *See* Peter Behr, *As Smart Grid Expands, So Does Vulnerability to Cyber Attack*, N.Y. Times, Nov. 19, 2009, at http://www.nytimes.com/cwire/2009/11/19/19climatewire-as-smart-grid-expands-so-does-vulnerability-25941.html.

43 *See, e.g.*, Edward Skoudis, *Evolutionary Trends in Cyberspace*, in Cyberpower and National Security 161–162, *supra* note 26.

44 *See* William D. O'Neil, *Cyberspace and Infrastructure*, in Cyberpower and National Security 113, 139, *supra* note 26.

45 *See* Rice, *supra* note 19, at 288–291, Michael D. Scott, *Tort Liability for Vendors of Insecure Software: Has the Time Finally Come?* 67 Md. L. Rev. 425 (2008); Vincent R. Johnson, *Cybersecurity, Identity Theft, and the Limits of Tort Liability*, 57 S.C. L. Rev. 255 (2005); Michael L. Rustad & Thomas H. Koenig, *The Tort of Negligent Enablement of Cyber Crime*, 20 Berkeley Tech. L.J. 1553 (2005); Bruce Schneier, *Liability and Security*, Crypto-Gram Newsl. (Counterpane Internet Security, Santa Clara, Cal.), Apr. 15, 2002, http://www.schneier.com/crypto-gram-0204.html#6. None of these works directly confront the market failures outlined above, however. For instance, Professor Johnson's account of cybersecurity risks focuses on database breach and identity theft. Although significant, these threats avoid the problem of the unknown victim and (as Professor Johnson notes) may be covered by tort law's doctrine of the voluntary assumption of duty. *See* Johnson, *supra*, at 278–80.

46 *See MacPherson v. Buick Motor Co.*, 217 N.Y. 382 (1916) (Cardozo, J.) (affirming a judgment for a plaintiff injured by a faulty wheel).

47 *See Greenman v. Yuba Power Products, Inc.*, 59 Cal. 2d 57 (1963) (Traynor, J.) (imposing strict liability on the manufacturer of a faulty home woodworking tool).

48 *See* Restatement (Third) of Torts: Products Liability § 1 cmt. a (1998). Of course, with fewer barriers to recovery comes more risk of over-deterrence. *See infra*, TAN 67–81. *See generally* A. Michael Polinsky & Steven Shavell, *The Uneasy Case for Product Liability*, 123 Harv. L. Rev. (forthcoming Apr. 2010).

49 Earlier work has considered in some detail how software developers might be made liable for their software's flaws under current law. *See* Scott, *supra* note 34; Johnson, *supra* note 34.

50 *See, e.g.*, Scott, *supra* note 34, at 453–56, 470–71; Johnson, *supra* note 34, at 296–303.

51 *See, e.g.*, Scott, *supra* note 34, at 450–53.

52 *See, e.g.*, *id.* at 461–67.

53 *See* sources cited above, notes 7–8.

54 With the passage of the 2001 USA PATRIOT Act, Congress expressly forbade "action[s] . . . brought under [the CFAA's civil provision] for the negligent design or manufacture of computer hardware, computer software, or firmware." Pub. L. 107-56, § 814(e) (2001) (amending 18 U.S.C. § 1030(g)). This amendment does not, however, appear to forbid or preempt non-statutory actions.

55 *See* Rice, *supra* note 19, at 184.

56 *See* Schneier, *supra* note 34 ("A company doesn't buy security for its warehouse— strong locks, window bars, or an alarm system — because it makes it feel safe. It buys that security because its insurance rates go down. The same thing will hold true for computer security.").

57 *See* Jay P. Kesan, Ruperto P. Majuca, & William J. Yurcik, Cyberinsurance as a Market-Based Solution to the Problem of Cybersecurity — A Case Study (unpublished manuscript, 2005), *available at* http://infosecon.net/workshop/pdf/42.pdf.

58 While it is beyond the scope of this proposal to consider all of the extant standards on network and information security that might be brought to bear on the cybersecurity problem, suffice it to say that the International Standards Organization, the National Institute of Standards and Technology, and consortia representing several sectors have published standards. Today, while specific recommendations for software development are few and incentives for compliance and certification are low, considerable groundwork has been laid for definite and effective standards that could support legal liability. *See* Info. Security Forum, The Standard of Good Practice for Information Security (2007), Int'l Standards Org., 27002:2005: Code of Practice For Information Security Management (2005). *See generally* Idaho Nat'l Lab., A Comparison Of Cross-Sector Cyber Security Standards (2005), *available at* http://www.inl.gov/scada/publications/d/a_comparison_of_cross-sector_cyber_security_standards.pdf.

59 Liability insurance's ability to improve market standards was perhaps best illustrated in the creation of the National Fire Protection Association by a group of property insurers. The group was instrumental in devising early building and electrical codes, and remains an active standard-setting body today. *See* Casey Cavanaugh Grant, *The Birth of NFPA*, http://www.nfpa.org/itemDetail.asp?categoryID=500&itemID=18020&URL=About%20Us/Overview/History (1996). *See also* Fleming James, Jr., *Accident Liability Reconsidered: The Impact of Liability Insurance*, 57 Yale L. J. 549, 559–63 (1948).

60 *See* Brian Krebs, *White House Pushing Cybersecurity Insurance*, Wash. Post, June 27, 2002 ("Only a handful of insurers currently offer cybersecurity policies. Coverage areas now include theft of data, denial-of-service and virus attacks, Web site defacement and subsequent outages, credit card fraud and cyber-extortion.").

61 Kesan, Majuca, & Yurcik, *supra* note 46, at 6–7, 9.

62 *See* Krebs, *supra* note 48.

63 *See* Robert Richardson, 2008 CSI Computer Crime & Security Survey 12 (2008) (indicating that thirty-four percent of polled firms carry "cyber insurance"). Coverage has gained only slightly since 2004 (the first year data is available), when twenty-eight percent of respondents had taken out policies. Lawrence A. Gordon et al., 2004 CSI/FBI Computer Crime and Security Survey, 7 fig. 10 (2004).

64 *See* Deborah Gage, *Security Worries Ratcheting Up; Spending Down*, Info. Wk., July 28, 2009, *at* http://www.informationweek.com/news/security/management/showArticle.jhtml?articleID=218700125 (reporting on a survey of CIOs and security managers that shows twenty percent of respondents expecting to cut investments in the next year and fifty-seven percent expecting their budget to be their greatest challenge over the same period, even as most reported an uptick in attacks over the last year).

65 One problem insurers have confronted is the difficulty in predicting the risks from vulnerabilities. *See* Anya Martin, *Cyberinsurance Offers Affordable Security*, Atlanta Bus. J., Mar. 22, 2002, *available at* http://www.bizjournals.com/atlanta/stories/2002/03/25/focus10.html. A policy that holds developers liable for their vulnerable software can best avoid complicating this issue further if it lies transparently on the underlying question of vulnerabilities.

66 *Cf.* Kathleen Sullivan, *The Supreme Court, 1991 Term—Foreword: The Justices of Rules and Standards*, 106 Harv. L. Rev. 22, 62 (1992) (arguing that rules promote greater certainty of outcome than do standards, even as they appear arbitrary at their margins).

67 *See, e.g.*, Seldon J. Childers, Note, *Don't Stop the Music: No Strict Products Liability For Embedded Software*, 19 U. Fla. J.L. & Pub. Pol'y 125, 159 (2008) (predicting that software developers will be unable to obtain insurance, and thus that costs of adverse judgments will be passed to consumers directly); Luther Martin, *Software liability is a bad idea*, Superconductor, Oct. 9, 2008, http://superconductor.voltage.com/2008/10/software-liabil.html (invoking the Coase Theorem); Howard A. Schmidt, *Give Developers Secure-Coding Ammo*, Nov. 3, 2005, http://news.cnet.com/Give-developers-secure-coding-ammo/2010-1002_3-5929364.html; *cf., e.g.*, Sandra A. Slaughter, Donald E. Harter, & Mayuram S. Krishnan, *Evaluating the Cost of Software Quality*, 41 Comms. ACM 67 (1998), *available at* http://www.cse.buffalo.edu/~hungngo/SCE-Papers/p67-slaughter.pdf (pointing out that repairing some software flaws is socially inefficient no matter what the motivation).

68 *Cf.* Polinsky & Shavell, *supra* note 37, at III. *Cf. also id.* ("[Most critics of product liability] stress that product liability raises product prices and causes firms to withdraw products, though they do not recognize that these consequences are socially undesirable only if the litigation cost-related component of the price increase is sufficiently high.").

69 *See, e.g.*, Clay Shirky, *Weblogs and the Mass Amateurization of Publishing*, Networks, Economics, and Culture mailing list, *available at* http://www.shirky.com/writings/weblogs_publishing.html.

70 *See* Zittrain, *supra* note 17, at 7–9.

71 *See id.*, at 17–18. Networks too can be described in these terms, with the open, provider-independent expanses of the Internet contrasted against the "walled garden" services provided by CompuServe in years past or by Facebook's centrally managed application platform today. *See id.*, at 23–30.

72 To focus on the two examples cited above, Apple's iPod and Facebook each dominate their respective market. *See Notes of Interest From Apple's Q1 2010 Conference Call*, Apple Insider, Jan. 25, 2010, http://www.appleinsider.com/articles/10/01/25/notes_of_interest_from_apples_q1_2010_conference_call.html (reporting that the iPod's share of the MP3 player market is almost seventy percent); *Led by Facebook, Twitter, Global Time Spent on Social Media Sites up 82% Year over Year*, NielsenWire, Jan. 22, 2010, http://blog.nielsen.com/nielsenwire/global/led-by-facebook-twitter-global-time-spent-on-social-media-sites-up-82-year-over-year/ ("With 206.9 million unique visitors, Facebook was the No. 1 global social networking destination in December 2009 . . . .").

73 *See* Zittrain, *supra* note 17, at 101–102.

74 *See, e.g.*, Scott Forstall, Senior Vice President of iPhone Software, Apple, Inc., Apple Special Event, Keynote Presentation at the Yerba Buena Center for the Arts (Jan. 27, 2010), *available at* http://events.apple.com.edgesuite.net/1001q3f8hhr/event/index.html (averring that Facebook's mobile application "just works" on Apple's new iPad).

75 *See id.*, at 8–9.

76 *Compare* David Cole & James X. Dempsey, Terrorism and the Constitution: Sacrificing Civil Liberties in the Name of National Security (2002) (arguing that trading security for constitutional rights itself violates the spirit of the Constitution) *with* Richard A. Posner, Not a Suicide Pact: The Constitution in a Time of National Emergency (2006) *and* Bruce Ackerman, *The Emergency Constitution*, 113 Yale L.J. 1029 (2004) (arguing that security and individual rights can be reconciled).

77 *See* Ross Gittins, *The Terrifying Cost of Feeling Safer*, Sydney Morning Herald, Aug. 26, 2008, *available at* http://www.smh.com.au/business/the-terrifying-cost-of-feeling-safer-20080826-435l.html; John Mueller, The Quixotic Quest For Invulnerability: Assessing the Costs, Benefits, and Probabilities of Protecting the Homeland (unpublished manuscript presented at the National Convention of the International Studies Association, Mar. 26–29, 2008), *available at* http://psweb.sbs.ohio-state.edu/faculty/jmueller/ISA2008.pdf.

78 *See, e.g.*, Kesan, Majuca, & Yurcik, *supra* note 46, at 23–25.

79 *See* Zittrain, *supra* note 17, at 40–52.

80 For instance, as security holes in operating system vulnerabilities are sealed, the need for aftermarket defenses on the individual user's level may actually decrease. This could spell the end for the economically inefficient (and technically ineffective) consumer antivirus industry. *See* Juhani Eronen et al., *Vulnerability vs. Critical Infrastructure – A Case Study of Antivirus Software*, 2 Int'l J. Advances Security 72 (2009); Carlos del Cacho, *The Economics of Software Products: An Example of Market Failure* 9–12 (Munich Personal RePEc Archive, Paper no. 17877, 2009), *available at* http://mpra.ub.uni-muenchen.de/17877/.

81 *See* Shavell, *supra* note 37, at 3.

82 For instance, no degree of network security can prevent infiltration by a bribed or disgruntled employee. *Cf.* Wilson, *supra* note 26, at 425.

83 *See, e.g.*, Michael C. Gemignani, *Product Liability and Software*, 8 Rutgers Computer & Tech. L.J. 173 (1980); Susan Nycum, *Liability for Malfunction of a Computer Program*, 7 Rutgers J. of Computers, Tech., & L. 1 (1979); *see also supra*, note 21 (describing the sequence of private paradigm software liability proposals).

84 *See supra*, TAN 12–20.

85 *See* sources cited above, notes 17–19.

86 *See, e.g.*, Ria Klaassen, "After Deleting a Bookmark, Bookmarks-Menu Closes," Mar. 3, 2006, https://bugzilla.mozilla.org/show_bug.cgi?id=329369.

87 *See, e.g.*, Juan Becerra, "Crash While Loading Video Clip [@ Memcpy — Oggplay_Data_Handle_Theora_Frame]," June 24, 2009, https://bugzilla.mozilla.org/show_bug.cgi?id=504843.

88 *See* Giri Vijayaraghavan & Cem Kaner, Bug Taxonomies: Use Them to Generate Better Tests (unpublished manuscript presented at Software Testing Analysis and Review East, 2003), *available at* http://www.testingeducation.org/a/bugtax.pdf ("Among all sub-specializations within computer science, computer security and vulnerability analysts have probably employed taxonomies in the largest way to classify security holes, vulnerabilities and other related security breaches.").

89 This distinction is not perfect; some vulnerabilities will appear as performance flaws to innocent users. For instance,

an unchecked input that permits a buffer overflow might be used to inject code into memory by an attacker, while an unsuspecting user might experience the same problem as a software crash and a cryptic error message about a segmentation fault. However, if the hypothetical program's developer were held liable for security vulnerabilities but not (assuming she uses a standard EULA) for mere crashes and data losses, she will classify the buffer overflow as a security vulnerability and prioritize it accordingly.

90 If the assumption on which this argument relies, that efforts to improve security are generally separable from more general efforts at improving user-facing functionality, proves false for some developers or development models, affected firms should be able to capitalize on their investment by charging a higher price for their software than their competitors. In some cases, the market may even support these firms' offering liability for private paradigm flaws as a "feature."

91 *See* David Rosenberg, *Individual Justice and Collectivizing Risk-Based Claims in Mass-Exposure Cases*, 71 N.Y. U. L. Rev. 210, 216–24 (1996).

92 Of course, requiring an actual attack implies requiring that that attack have produced actual losses for the plaintiff. Similarly, background legal principles are assumed: the attack must come from a hostile outsider, the plaintiff itself or a security researcher seeking a proof of concept. If, on the other hand, the little-understood market for the disclosure of security flaws comes to threaten cybersecurity rather than help it, these requirements could be lifted, in order to give security professionals a competing avenue to profit from their work.

93 *See generally* Stephen A. Shepherd, "Vulnerability Disclosure: How Do We Define Responsible Disclosure?" (unpublished manuscript, 2003), *available at* http://www.sans.org/reading_room/whitepapers/threats/how_do_we_define_responsible_disclosure_932?show=932.php&cat=threats. At present, the responsible disclosure norm is widely observed in the security community, but many security researchers are frustrated by developers that fail to apply a similar standard of responsibility in correcting disclosed flaws. Brian Martin, "Responsible Disclosure — Old Debate, Fresh Aspects?!", on OSVDB Blog, Nov. 15, 2009, at http://blog.osvdb.org/2009/11/15/responsible-disclosure-old-debate-fresh-aspects.

94 *See* Stefan Frei et al., "Modelling the Security Ecosystem — The Dynamics of (In)Security" 14 (unpublished manuscript, presented at the Workshop on the Economics of Intformation Security, June 2009), *available at* http://www.springerlink.com/content/n85p24761500u21j/ ("Exploit availability 30 days after disclosure continuously exceeds 90% since 2004."). Moreover, the idea of a patching period assumes "white hat" hackers and responsible disclosure. In many cases, a security flaw is only discovered publicly once it is exploited by a hostile attacker. *See id.* at 8.

95 *See, e.g.* Yochai Benkler, *Coase's Penguin, or, Linux and* The Nature of the Firm, 112 Yale L.J. 369, 371 n.2 (distinguishing the use of the terms "free software" and "open source software").

96 *See, e.g.*, K.S. Sampath Kumar, Understanding Free Open Source Software 2 (version 3.0i, 2009). Readily available source code means that open source software may be compiled by anyone, and thus compiled versions (that is, those that end users can run without further processing) are almost always distributed without charge as well.

97 Amit Deshpande & Dirk Riehle, *The Total Growth of Open Source*, *in* Proceedings of the Fourth Conf. on Open Source Systems 197, 197 (2008).

98 "Open Source Projects Double Every 14 Months!," *on* Analytics, http://www.amit-deshpande.com/ (Apr. 14, 2008, 11:59 AM), http://www.amit-deshpande.com/2008/04/open-source-projects-double-every-14.html.

99 *See* "January 2010 Web Server Survey," *on* Netcraft, http://news.netcraft.com/ (Jan. 7, 2010), http://news.netcraft.com/archives/2010/01/07/january_2010_web_server_survey.html.

100 *See* Charles Babcock, *Sun Locks Up MySQL, Looks to Future Web Development*, InformationWeek, Feb. 26, 2008, *available at http://www.informationweek.com/news/software/open_source/showArticle.jhtml?articleID=206900327*.

101 *See* Tiobe Software, TIOBE Programming Community Index for January 2010 (2010), *available at http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html*.

102 For instance, the Firefox browser is backed by the Mozilla Foundation, a 501(c)(3) non-profit corporation, *see* Mozilla.org Announces Launch of the Mozilla Foundation to Lead Open-Source Browser Efforts, (July 15, 2003), *available at http://www-archive.mozilla.org/press/mozilla-foundation.html*, and Fedora Linux is a product of Red Hat, Inc., a for-profit corporation, *see* Red Hat Inc., Annual Report (Form 10-K), at 1, 3 (Apr. 29, 2009), *available at http://investors.redhat.com/secfiling.cfm?filingID=1193125-09-91983*. However some of these major projects are organized outside of the United States. The Ubuntu Linux distribution, for instance, is published by Canonical Group Ltd., a United Kingdom company. *See* Canonical, http://www.canonical.com/aboutus (last visited Jan. 10, 2010).

103 The three largest free open source project hosting websites alone can claim more than 420,000 projects between them. *See* Jonathan Rosenberg, "The Meaning of Open," *on* The Official Google Blog (Dec. 21, 2009), *http://googleblog.blogspot.com/2009/12/meaning-of-open.html* (reporting "over 250,000 projects" on Google Code); SourceForge, *http://sourceforge.net/softwaremap/trove_list.php?stquery=&sort=group_ranking&form_cat=18* (last visited Jan. 10, 2010) (160,245 projects on SourceForge); Launchpad, *https://launchpad.net/projects/* (last visited Jan. 10, 2010) (15,907 projects on Launchpad).

104 *See* Paul H. Arne, Open Source Software Licenses: Perspectives of the End User and the Software Developer 19 (2004), *available at http://www.zdnetasia.com/whitepaper/open-source-software-licenses-perspectives-of-the-end-user-and-the-software-developer_wp-1862513.htm* ("Open source software is frequently developed by multiple persons who work for multiple employers, who provide code on a more or less anonymous basis (at least to the [e]nd [u]ser or [d]eveloper), who are relatively judgment proof, [and] who provide the code for free . . . ."). Moreover, like the projects to which they contribute, many open source developers reside outside the jurisdiction of American courts.

105 *See, e.g.*, Jonathan Corbet, How to Participate in the Linux Community 4 (2008), *available at http://ldn.linuxfoundation.org/documentation/how-participate-linux-community* ("It is imperative that all code contributed to the kernel be

legitimately free software. For that reason, code [submitted to the Linux kernel] from anonymous (or pseudonymous) contributors will not be accepted.”).

106 *See* Slashdot: Real Name for Open Source Development?, *http://yro.slashdot.org/article.pl?sid=08/11/17/1746239* (last visited Jan. 11, 2010). Of course, the same reasons that these contributors to legally unorganized projects would be difficult to sue individually make them difficult to insure individually.

107 A versioning (or version control) system creates a permanent record of each change each user has committed to the project’s files. The number of freely available, hosted versioning systems has greatly increased over the last decade years, with services like GitHub, Launchpad, and Google Code joining stalwarts like SourceForge. Each of these services includes a built-in bug tracker, which is meant to allow users to register observed flaws in the software and coordinate contributors’ efforts in correcting them. Some projects employ freestanding bug trackers; perhaps the most commonly deployed is the Mozilla Project’s Bugzilla.

108 *See* Heather J. Meeker, *Legal Issues: The Contribution Conundrum*, Enterprise Open Source J., July/Aug. 2006 (“The top open source projects are now run more diligently and professionally than many private start-up software companies.”).

109 *See, e.g.*, Dana Blankenhorn, *Open Source Shrugs at EU Liability Plans*, ZDNet, May 13, 2009, at *http://blogs.zdnet.com/open-source/?p=4208*. The ALI’s Principles of the Law of Software Contracts enshrine something like immunity from liability for defects for open source software, but it demarcates its safe harbor based on the software’s price, not the way in which it was developed. Principles of the Law of Software Contracts, § 3.05(b) (Proposed Final Draft 2009) (“A transferor that receives money . . . in exchange for the software warrants . . . that the software contains no material hidden defects of which the transferor was aware at the time of the transfer.”) This orientation has drawn criticism. *See* Linux-Microsoft Letter, *supra* note 20.

110 *See, e.g.*, Jaap-Henk Hoepman & Bart Jacobs, *Increased Security Through Open Source*, 50 Comms. ACM 79 (2007) (“Our main argument is that opening the source allows independent assessment of the exposure of a system and the risk associated with using the system, makes patching bugs easier and more likely, and forces software developers to spend more effort on the quality of their code.”).

111 Eric S. Raymond, The Cathedral And the Bazaar 30 (2001). “Linus’s law” is stated more concretely: “Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.” *Id.*

112 Ken Thompson, Fellow, AT&T Bell Labs., Reflections on Trusting Trust, Turing Award Lecture, 1983, *in* 27 Comms. ACM 761 (1984). Thompson begins by noting that, had he simply added code for a backdoor in `login`’s own source code, it would have been quickly spotted and removed. Instead, he modified `cc` (Unix’s C compiler) to add the backdoor code to login when `login` was compiled. He then added code to `cc` that would add his first modification whenever it detected that it was compiling a new version of itself. This created the possibility for a massive security vulnerability that would be literally invisible to even the finest programmer perusing both `login` and `cc`’s source. Though Thompson’s own hack was only a proof of concept, at least one recent virus has used the method he described. *See* Richard Cohen, “Compile-a-virus - W32/Induc-A,” *on* SophosLabs Blog (Aug. 18, 2009), *http://nakedsecurity.sophos.com/2009/08/18/compileavirus/*.

113 Programmers are reminded of this fact annually by the well-known Underhanded C Contest. Entrants in the competition compete by “writ[ing] innocent-looking C code implementing malicious behavior . . . . The main goal . . . is to write source code that easily passes visual inspection by other programmers.” About the Underhanded C Competition, *http://underhanded.xcott.com/?page_id=2* (last visited Jan. 13, 2010).

114 In this context, open source software would most deserve immunity if the riskiest projects had the most volunteers (that is, the most “eyeballs”). Risk is a function of many variables, but the installed base is of particular importance. *See infra*, TAN 122–130. Certainly, the installation base is relevant to a contributor’s decision of which projects to join (not least because no one contributes to a project he has never heard of). At the same time, many other factors influence that decision, and many of these do not relate directly to the project’s risk profile. *See, e.g.*, Andrew Schofield & Grahame S. Cooper, *Participation in Free and Open Source Communities: An Empirical Study of Community Member’s Perceptions*, *in* 203 IFIP Int’l Fed’n for Info. Processing 221–31 (Ernesto Damiani et al. eds, 2006); Robert M. Sauer, *Why Develop Open Source Software? The Role of Non-Pecuniary Benefits, Monetary Rewards and Open Source Licence Type*, 25 Ox. Rev. of Econ. Pol’y 605–19 (2007). Therefore, it is impossible to say a priori that all of the open source projects most important to the cybersecurity problem will have the most volunteers contributing to them.

On a similar note, even if the allocation of programmers to projects is risk-efficient in the long run, the rate of sign-ups may not keep up with sudden changes in risk profile. *Cf., e.g.*, Zittrain, *supra* note 17, at 154 (“[T]oday’s obscure but useful back-water application can find itself wildly popular and relied upon overnight.”).

115 The best example of this situation is undoubtedly that of Mozilla’s Firefox browser. As publishers of the number one and two browsers, respectively, Microsoft and Mozilla’s release schedules frequently coincide. *See, e.g.*, Rhys Blakely, *Firefox 2.0 to Reignite Browswer Battle*, The Times, Oct. 23, 2006 (noting that Firefox 2.0’s release came “just days” after Internet Explorer 7’s). More recently, Firefox 3.5 shipped within three months of closed-source competitors Internet Explorer 8, Apple’s Safari 4, Google’s Chrome 2, and Opera Software’s Opera 10.

116 The developers of the WordPress blogging platform, for instance, announce planned releases months in advance, although its competitors do not appear to. *See* “Roadmap,” http://wordpress.org/about/roadmap/ (last visited Jan. 29, 2010).

117 While comparing two programs’ security presents many challenges, by at least one measure open source software appears to beat their proprietary competitors. *See* Ashish Arora et al., An Empirical Analysis of Software Vendors’

Patching Behavior: Impact of Vulnerability Disclosure, at 26 (unpublished manuscript, Jan. 2006), *available at* http://www.heinz.cmu.edu/%7Ertelang/disclosure_jan_06.pdf finding that open source developers are much more likely to patch discovered vulnerabilities than are proprietary developers).

118 The Defense Department has reached the same conclusion. *See* Frequently Asked Questions regarding Open Source Software (OSS) and the Department of Defense (DoD), http://cio-nii.defense.gov/sites/oss/Open_Source_Software_%28OSS%29_FAQ.htm (last visited Jan. 12, 2010) ("[T]he DoD will not reject consideration of a [commercial, off-the-shelf] product merely because it is [open source software]. Some [open source software] is very secure, while others are not; some proprietary software is very secure, while others are not. Each product must be examined on its own merits.").

119 *See, e.g.*, Bruce Schneier, *Full Disclosure and the Window of Exposure*, Crypto-Gram Newsl. (Counterpane Internet Security, Santa Clara, Cal.), Sept. 15, 2000, http://www.schneier.com/crypto-gram-0009.html#1 (charting the importance of who discovers a vulnerability, how the vulnerability is announced, how soon an automated exploit is released, how quickly the vendor releases a patch, and how quickly users install it).

120 *See* Matt Asay, *EC Takes Three Steps Back on Software Liability*, CNET News, May 12, 2009, at http://news.cnet.com/8301-13505_3-10238475-16.html?tag=col1;post-4208.

121 *See* William M. Landes & Richard A. Posner, *A Positive Economic Analysis of Product Liability*, 14 J. Legal Stud. 545–46 (1985) (noting the converse: that where the law allows it, some consumers will trade a waiver of liability for a lower price).

122 *Cf.* James F. Blumstein, Randall R. Bovbjerg, & Frank A. Sloan, *Beyond Tort Reform: Developing Better Tools For Assessing Damages For Personal Injury*, Yale J. on Reg. 171, 173 (1991) ("[Scheduled damages] would promote consistency and predictability of overall valuations by narrowing the very broad and standardless discretion currently accorded to juries.").

123 *See* Richard Craswell, *Deterrence and Damages: The Multiplier Principle and Its Alternatives*, 97 Mich. L. Rev. 2185, 2189–2192 (1999).

124 *See, e.g.*, Schneier, *supra* note 120.

125 *See* Daniel Geer et al., CyberInsecurity: The Cost of Monopoly (unpublished manuscript, Sep. 27, 2003), *available at* http://cryptome.info/0001/cyberinsecurity.htm (using the example of Microsoft Windows to discuss the risk of "cascade failures" that can compromise entire software "monocultures").

126 *See* Roger Grimes, *Don't Fall for the Monoculture Myth*, InfoWorld, Apr. 24, 2009.

127 Jeremy Kirk, *Tardy XP Service Pack: No Cause for Concern?*, PC World, Jan. 19, 2006. It need not be said that the Windows family of operating systems are also responsible for considerable cybersecurity risks. *See, e.g.*, John Markoff & John Schwartz, *Expert Says Windows XP Aids Vandals*, N.Y. Times, June 4, 2001, at C7.

128 Large-scale damages are subject to two main lines of attack. The more certain of these is the Supreme Court's recent decisions concerning punitive damages. In *BMW of North America v. Gore*, 517 U.S. 559 (1996), the Court held an award to have violated the Due Process Clause as "grossly excessive" relative to the harm for which the plaintiff deserved compensation. *Id.* at 562 (internal quotation marks omitted). In 2003, the Court elaborated that in most cases, punitive damages should not reach ten times the value of compensatory damages. State Farm Mutual Automobile Insurance Co. v. Campbell, 538 U.S. 408, 425 (2003). These cases are distinguishable as applying to jury awards rather than to legislation, however, and in fact both note that analogous statutory damages may set the standard for what is reasonable. *State Farm*, 538 U.S. at 429; *BMW*, 517 U.S. at 583–84. Furthermore, the doctrine suggests the possibility that awards that are as large as required to serve the public's deterrent goals will not violate due process. *See BMW*, 517 U.S. at 584.

Separately, many argue that the statutory damages provided in the Copyright Act, 17 U.S.C. § 504(c) (2008), can be unconstitutional. *See, e.g.*, Pamela Sameulson, *Unconstitutionally Excessive Statutory Damage Awards in Copyright Cases*, 158 U. Penn. L. Rev. Pennumbra 54 (2009). These arguments have never won the day in court, however, and they do not translate well to the network security context. Rather than approximating actual harm and willfulness, as under the Copyright Act, the damages provided for herein are meant expressly to deter.

129 *See* Louis Kaplow, *Optimal Deterrence, Uninformed Individuals, and Acquiring Information about Whether Acts Are Subject to Sanctions*, 6 J.L. Econ. & Org. 93 (1990).

130 It is by now axiomatic that obscurity is no basis for a secure system. *See, e.g.*, Rebecca T. Mercuri & Peter G. Neumann, *Security by Obscurity*, 46 Comms. ACM 160 (2003), *available at* http://203.130.231.110/pub/books/Communication-ACM/November-2003/p160-mercuri.pdf; *cf.* Anderson, *supra* note 17, at 517, 716–18 (referring to obscurity's failure in smartcard systems and copyright marking). Certainly, developers should not be encouraged to hide their flaws rather than fix them. Nevertheless, this does not mean that they should announce them before they are corrected.

131 As a point of reference on what number that limit or fixed fine ought to be, the CFAA authorizes fines of up to $10,000 for organizations that access computers without access, 18 U.S.C. § 1030, 3571(c)(7) (2008) (but expressly *not* for software vendors, *id.* § 1030(g)). Since attackers are inarguably more culpable than those who negligently enable their attacks, a base amount of less than $10,000 would seem to place this liability proposal in line with existing law.

132 Bus. Software Alliance, Sixth Annual BSA-IDC Global Software Piracy Study 6 tbl.1 (2009) , *available at* http://global.bsa.org/globalpiracy2008/index.html.

133 *See, e.g.*, Ivan P.L. Ping, On the Reliability of Software Piracy Statistics (unpublished manuscript, Sept. 1, 2008), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1099325; *BSA's Sweden Piracy Stats Entirely Estimated*, Posting to MichaelGeist.ca, May 20, 2009, at http://www.michaelgeist.ca/content/view/3987/196/;

HOME BACK NEXT

Mike Masnick, *BSA's Canadian Piracy Numbers Based On Hunches, Not Actual Surveys*, on TechDirt, May 27, 2009, at http://www.techdirt.com/articles/20090527/1125035034.shtml.

134 *See* John Walker, *The Digital Imprimatur: How Big Brother and Big Media Can Put the Internet Genie Back in the Bottle*, Knowledge Tech. & Privacy, Fall 2003, at 24–77, *available at* http://www.fourmilab.ch/documents/digital-imprimatur/.

135 It has not, for instance, broached the important question of how responsibility for attacks is to be apportioned among the developers of all the software on a compromised system that may have contributed to the system's vulnerability. Nor has it addressed whether and how the federal government might catalyze compliance with the new law by funding software developers themselves or standards-setting organizations. On the latter topic, consider the provisions of the Communications Assistance for Law Enforcement Act that authorized the Attorney General to fund compliance with the law's wiretapping requirements, 47 U.S.C. § 1008 (2008), and the aborted POSSE initiative, which saw the Defense Department fund security improvements in major open source software projects, *see* Robert Lemos, *Defense Agency Pulls OpenBSD Funding*, CNET News, Apr. 17, 2003, at http://news.cnet.com/2100-1016-997393.html.

Using Technology to Leverage your Job Search

By Susanna Brennan

There are so many different websites and tools today for conducting a job search. It can be overwhelming, especially if you're working full time and job searching at the same time. Here are a few tips for making the task more efficient:

1. Your online profile:
    - Set up a profile in LinkedIn and keep it up to date. Connect with friends, family members, colleagues, etc. to grow your network. LinkedIn is a great research tool for job searchers and is increasingly being used by companies to recruit.
    - If you have a resume posted on sites like Monster or Career Builder, keep it up to date. Companies access these sites daily too, and you don't want your credentials to be passed over because of out of date information. If you're concerned about confidentiality, mask identifying information.
    - If you have a Facebook page, consider deactivating it or minimizing the information on it during a job search.

2. Finding jobs:

- Set up a search on aggregator sites such as indeed.com or simplyhired.com to create customized job alerts, which are sent to your e-mail every day, week, etc. These sites pull jobs from all different websites. There are a number of free apps for tablet computers that perform the same function. Most of these sites or applications allow you to upload your resume or create a career profile, so you can easily apply for jobs.
- Join groups on LinkedIn – nearly every group has a "Jobs" or "Career Discussions" tab – and again, you can receive these updates every day, week, etc. Participate in discussions with people who you'd like to work with or may be able to make a connection for you.
- Avoid sending your resume en masse and setting yourself up for disappointment when you don't get a response. Focus your efforts on the jobs you want and that you know or reasonably believe that you are qualified for. Use the rest of your time using technology to do research and strengthen your connections. ■

Susanna Brennan, an IT Law Section Council Member and Recruitment Director for Kelly Law Registry, is available to assist Section Members with resume feedback, interviewing tips, and job search assistance. Susanna is an experienced recruiter and career consultant who places attorneys and legal professionals in contract and permanent positions with law firms, corporate legal departments, and other organizations in the Midwest. For more information, please contact Susanna at brennsc@kellylawregistry.com or (248) 952-0539.

HOME BACK NEXT

Publicly Available Websites for IT Lawyers

Following are some publicly available websites relating to varying aspects of information technology law practice. Some of these websites may require payment for certain services. Neither the State Bar of Michigan nor the IT Law Section endorses these websites, the providers of the website, or the goods or services offered in connection therewith. Rather these websites are provided for information purposes only and as possible useful tools for your law practice.

Please provide any feedback or recommendations for additional websites to *michael@gallo.us.com*.

Security and Privacy

- *http://www.ncjrs.gov/pdffiles1/nij/219941.pdf* - National Institute of Justice Special Report, Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition (2008).
- *http://www.privacyrights.org/fs/fs18-cyb.htm#Interactive_Use* – Privacy Rights Clearinghouse provides a fact sheet regarding 'Online Privacy: Using the Internet Safely'
- *http://www.privacyrights.org/fs/fs7-work.htm* – Privacy Rights Clearinghouse provides a fact sheet regarding 'Workplace Privacy and Employee Monitoring'
- *http://www.ipc.on.ca/images/Resources/privacyintheclouds.pdf* - Privacy in the Clouds – a white paper on privacy and digital identity. ■

2011 Edward F. Langs Writing Award

Essay Competition Rules

1. Awards will be given to up to three student essays, which in the opinion of the judges make the most significant contribution to the knowledge and understanding of information technology law. Factors to be taken into consideration include: originality; timeliness of the subject; depth of research; accuracy; readability; and the potential for impact on the law.
2. Essay must be original, deemed to be of publishing quality, and must not have been submitted to any other contest within the previous 12 months.
3. Essay must be typed, double spaced, at least ten pages in length, must contain proper citations listed as either endnotes or footnotes, and must have left, right, top, and bottom margins of one inch.
4. Essay must include the submitter's name, email address, mailing address, telephone number, and school attended.
5. A total of $1,500 in US dollars shall be divided between the award winning essays, and all rights to award winning essays shall become the property of the State Bar of Michigan.
6. The Information Technology Section of the State Bar of Michigan reserves the right to make editorial changes, and to publish award winning essays in the Section's newsletter, the *Michigan IT Lawyer.*
7. Essay must be submitted as a Microsoft Word document, postmarked by June 30, 2011, and emailed to ***dsyrowik@brookskushman.com***. ■

Mission Statement Information Technology Law Section, State Bar of Michigan

The purposes of the Section are to review, comment upon, and appraise members of the State Bar of Michigan and others of developments in the law relating to information technology, including:

(a) the protection of intellectual and other proprietary rights;

(b) sale, leasing, distribution, provision, and use of, hardware, software, services, and technology, including computer and data processing equipment, computer software and services, games and gaming, information processing, programming, and computer networks;

(c) electronic commerce

(d) electronic implementation of governmental and other non-commercial functions;

(e) the Internet and other networks; and

(f) associated contract and tort liabilities, and related civil and criminal legal consequences.