THE TALLINN PAPERS
A NATO CCD COE Publication on Strategic Cyber Security

LIIS VIHUL

THE LIABILITY OF SOFTWARE MANUFACTURERS FOR DEFECTIVE PRODUCTS

Vol. 1, No. 2 2014

CCDCOE
NATO Cooperative Cyber Defence
Centre of Excellence Tallinn, Estonia

Previously in This Series

Vol. 1, No. 1 Kenneth Geers "Pandemonium: Nation States, National Security, and the Internet"

Roles and Responsibilities in Cyberspace

The theme of the 2014 Tallinn Papers is "Roles and Responsibilities in Cyberspace". Strategic developments in cyber security have often been frustrated by role assignment, whether in a domestic or international setting. The difficulty extends well beyond the formal distribution of roles and responsibilities between organisations and agencies. Ascertaining appropriate roles and responsibilities is also a matter of creating an architecture that is responsive to the peculiar challenges of cyberspace and that best effectuates strategies that have been devised to address them.

The 2014 Tallinn Papers address the issue from a variety of perspectives. Some of the articles tackle broad strategic questions like deliberating on the stance NATO should adopt in cyberspace matters, or exploring the role small states can play in this domain. Others touch upon narrower topics, such as the right to privacy in the growingly intrusive national security context and whether software manufacturers should be compelled to bear their burden of cyber security by making them liable for faulty software. The thread running through all the papers, however, is their future-looking approach, one designed to inspire discussion and undergird strategic development.

Submissions

The Tallinn Papers is a peer reviewed publication of the NATO Cooperative Cyber Defence Centre of Excellence. Although submissions are primarily commissioned by-invitation, proposals consistent with the annual theme and dealing with issues of strategic importance will be considered on an exceptional basis. Since the Tallinn Papers are meant for a wide audience, such proposals should assume no prior specialised knowledge on the part of the readership. Authors wishing to submit a proposal may contact the Editor-in-Chief at publications@ccdcoe.org.

The Liability of Software Manufacturers for Defective Products

Liis Vihul[1]

The most effective and cost-efficient route to cyber security – to making our IT infrastructure more resilient to hostile cyber operations – is the development of secure code, for the fewer the vulnerabilities in computer code, the less systems can be manipulated, be it for monetary gain, espionage or system manipulation by States. Yet, construing a legal, economic and policy framework to achieve this objective has eluded the security community for over two decades.

Arguing that design defects in software cannot be entirely avoided, most computer scientists appear opposed to any standardisation as to how perfect lines of code they should be required to write. Moreover, software companies assert that, in light of the unfeasibility of developing absolutely secure code, they should not be required to assume the risks associated with the exploitation of vulnerabilities in their products. The companies fairly note that so long as an incentive to detect and exploit such vulnerabilities exists, successful attacks against their software are inevitable. Accordingly, licencing agreements tend to shield developers from most liability with respect to end-users who purchase their "shrink-wrapped" off-the-shelf products, thereby down-streaming risk.

There are persuasive arguments for the reallocation of risk. The most fervent advocates of software manufacturer liability have been computer security experts, who understand that the fundamental cause of cyber insecurity is the existence of loopholes in hardware and software, the occurrence of which can be reduced with more careful programming.[2] They are joined by end-users who deem it unfair that they should assume the risk of insecure software. After all, others developed it, but the users suffer the consequences of insecure code, which range from inconvenience, such as the recurring need to patch the products, to severe business loss. In light of their limited bargaining power with respect to the reallocation of risk, it is only natural that they would look to the law to mitigate this inequality.

1 Researcher, NATO Cooperative Cyber Defence Centre of Excellence.

2 See, e.g., Bruce Schneier, 'Information Security and Externalities' (*Schneier on Security,* 18 January 2007), available at: https://www.schneier.com/blog/archives/2007/01/information_sec_1.html.

A major obstacle to the reallocation of risk is that civil law seldom imposes liability when its consequences are unforeseeable and potentially unlimited.[3] It is this very unpredictability that underlies the reluctance of legislators, who in principle enjoy the authority to oblige software developers to produce better quality products, to act. The influence of lobbyists promoting the interests of software developers exacerbates the situation. Moreover, the national and foreign security policy goals of States are often pursued through cyber means. Their ability to obtain or alter data, or to manipulate processes, depends on exploitable target computer systems. Somewhat paradoxically, States therefore have an interest in both secure and insecure code.

Should reallocation occur as a matter of law, it is likely to do so slowly. Any opening of the door to manufacturer liability will occur first in situations in which the number of potential plaintiffs is limited, the scope of the possible liability clear and predictable and litigation practicable. It will also necessitate some degree of acceptance of the norms by the manufacturers themselves, which would seem to fly in the face of software development and sales realities. Particular care will have to be taken to ensure responsive legislation does not generate undesirable consequences beyond its intended scope. The fact that civil liability tends to be rather abstract, applying uniformly to all civil affairs, will prove especially problematic in this regard. Clearly, the task of developing an effective legal framework that renders software developers liable for the damages caused by the exploitation of their products is a complex task.

Civil Liability Generally

There are two categories of civil liability: contractual liability arising out of a contractual relationship between parties, and tort (or delictual) liability that is based on wrongful harm that has occurred in a non-contractual relationship. In the current context, the primary legal relationship between a software company and an end-user is contractual in nature, as a licencing agreement is a form of contract. If the end-user suffers harm as a result of a malicious hacker exploiting a vulnerability in the software, the contractual relationship governs the recovery of damages from the manufacturer. Any claim for damages from the hacker would be governed by the law of tort.

Contractual relationships are based on the presumption of freedom of contract

3 The American Law Institute, 'Restatement of The Law Third, Torts: Liability for Economic Harm. Tentative Draft No. 1' (4 April 2012), § 1 comment c.

such that the parties to a contract are generally free to decide upon the terms and conditions thereof in order to maximise mutual benefit. This presumption, however, only applies so long as the parties are in an equal negotiating position. With respect to mainstream software products, users, due to a lack of viable alternatives, commonly find themselves in a situation in which the only available option is to purchase the software under the vendor's standard terms. This being the case, contract law provides no realistic options to redress the recovery of consequential damages caused by flawed software because courts are unlikely to invalidate a one-sided liability-limiting agreement.[4] As a result, mainstream software end-users wishing to utilise certain software applications are typically caught on the horns of a dilemma: they either have to accept the conditions set forth in licence agreements drafted by the manufacturers or refrain from entering into the agreement. By contrast, when large-scale IT procurement contracts are being negotiated, the extent to which purchasers can insist on specified conditions, including shared liability for software vulnerabilities, dramatically increases.[5]

The law can, of course, substantively limit the freedom of contract vis-à-vis liability. For example, because a contract for the development of an IT environment enabling the concealment of the financing of terrorism would likely violate domestic law prohibitions on such financing, the contract would be void and therefore unenforceable. However, legal constraints on the right to freely conclude contracts for the sale of software to end-users are rare; the majority of them stem from consumer protection legislation.

Consumer protection legislation only extends to end-users who acquire and use products in their private capacity, outside their business or professional activities. The most common consumer protection norm is that which disallows the vendor to impose a standard contractual term restricting or excluding its liability for bodily injury. Should they exist, such provisions are void. While domestic consumer protection legislation may extend the prohibition to provisions designed to exclude the manufacturer's liability for non-bodily injury caused intentionally or negligently, it is uncommon.

Contract terms can be ambiguous. In this regard, the American Law Institute has observed that "[a] contract can allocate a risk without mentioning it explicitly; silence may itself serve as an allocation if the risk falls within the scope of

4 Michael L. Rustad, Thomas H. Koenig, 'The Tort of Negligent Enablement of Cybercrime' 20 *Berkeley Technology Law Journal* 1558-9 (2005).

5 Jeremy Newton, 'System Supply Contracts' in Chris Reed (ed.), *Computer Law* (7th ed, 2011) 4.

activity the contract governs."[6] Importantly, should harm occur outside a licence agreement's coverage, plaintiffs can seek recovery for that harm through tort law. For example, if a licencing agreement does not exclude the developer's liability for a particular harm or any harm altogether, the user is entitled to recourse through tort law. Despite the existence of a contractual relationship in this case, recovery through tort law remains available because the court would protect the initial bargain the parties had reached, and not because it would penalise the defendant for failure to reach a broader one.[7]

The licence agreements of major software companies, however, are explicit as to their conditions of liability. For example, the licence terms for Microsoft Office 2010 desktop application software only allow recovery for direct damages up to the amount paid for the software.[8] Nevertheless, not all licence agreements are necessarily as broad regarding the terms of liability. Adobe accepts liability for gross negligence or intentional misconduct of the company or its employees in its terms for licensure.[9] Despite the fact that Adobe's general terms of use accept liability in these particular circumstances, they are still as precise and clear as Microsoft's, thereby reducing the potential for uncertainty or dissension regarding the scope of liability.

In common law legal systems,[10] the basis for non-contractual liability is the tort of negligence. To establish liability, the defendant has to breach a duty of care, as a result of which the plaintiff suffers harm. Additionally, the risk of harm has to have been reasonably foreseeable to the defendant. In civil law legal systems,[11] by contrast, no individual negligence tort exists; instead, negligence is a form of fault or culpability. Civil law torts accordingly generally consist of three elements similar to those of a criminal offence: the objective event, which encompasses the action of the tortfeasor, the harm, and the causal link; the unlawfulness of

6 *Supra* note 3, § 3 comment c.

7 *Ibid.*

8 Microsoft Corporation, 'Microsoft Software License Terms. Microsoft Office 2010 Desktop Application Software' pt. 26, available at: http://www.microsoft.com/en-us/download/details.aspx?id=13653.

9 Adobe General Terms of Use (16 October 2012), pt. 14.1, available at: http://www.adobe.com/misc/terms.html.

10 Common law is "[t]he body of law based on the English legal system, as distinct from a civil-law system." It is derived from judicial decisions, rather than from statutes or constitutions. – *Black's Law Dictionary*, "common law" (9th ed. 2009).

11 Civil law or Roman law is "[o]ne of the two prominent legal systems in the Western world, originally administered in the Roman Empire and still influential in continental Europe, Latin America, Scotland, and Louisiana, among other parts of the world." *Black's Law Dictionary*, "civil law" (9th ed. 2009).

the objective event; and fault. Not every objective event that results in harm is unlawful and even if harm results, liability will not attach if the actor was not at fault, i.e. acting intentionally or negligently.

Duty of Care in Software Manufacturing

In the common law context, "[a] person acts negligently if the person does not exercise reasonable care under all the circumstances."[12] In other words, negligence is assessed against a reasonableness standard; failure to take reasonable care in a particular circumstance qualifies as negligence. Factors to be taken into account in ascertaining whether negligence has occurred are the foreseeable likelihood that the person's conduct will result in harm, the foreseeable severity of any harm that may ensue, and the burden of precautions to eliminate or reduce the risk of harm.[13]

Under English common law, the traditional formulation of a duty of care dates back to a landmark 1932 case in which Lord Atkin seminally stated that "The liability for negligence […] is no doubt based upon a general public sentiment of moral wrongdoing for which the offender must pay. […] The rule that you are to love your neighbour becomes in law, you must not injure your neighbour […]. You must take reasonable care to avoid acts or omissions which you can reasonably foresee would be likely to injure your neighbour."[14] Today, reasonable care is defined as conduct that shows "ordinary care", avoids creating an "unreasonable risk of harm" and demonstrates "reasonable prudence".[15]

Of course, the level of care that is considered reasonable depends on the context in which it operates, and may vary from fairly loose requirements to very strict ones. Examples of the latter include activities that encompass a high level of risk of injury by nature, such as those involved in the medical profession or the infant food industry. Accordingly, "[i]n software law, as in any other area of tort law, the greater the risk, the greater the duty of care. […] A software vendor that markets tailored software to a hospital or to a financial institution, for example, would have a higher duty of care to produce secure software than a vendor

12 Restatement (Third) of Torts: Physical and Emotional Harm (2010), § 3 (United States).

13 *Ibid.*

14 *Donoghue v Stevenson*, [1932] All ER Rep 1; [1932] AC 562 (HL) 580. Lord Atkin went on to state "Who, then, in law is my neighbour? The answer seems to be - persons who are so closely and directly affected by my act that I ought reasonably to have them in contemplation as being so affected when I am directing my mind to the acts or omissions which are called in question."

15 *Supra* note 12, § 3 comment a.

marketing to the home entertainment market."[16]

The duty of reasonable care that applies to a particular activity may be set forth in the law itself. In such cases, ascertaining a breach of the duty simply requires that the conduct be assessed against the threshold set forth in the legislation. Should there be no clearly articulated standard binding upon the actor, conduct has to be evaluated at the abstract level, that is the expected conduct of an average reasonable person in the same or similar circumstances. This standard applies to both natural and legal persons. Non-binding instruments such as guidelines, recommendations, instructions, codes of best practice, and the like are helpful in this regard.[17]

The software industry lacks binding formulations of the due diligence that manufacturers are obliged to exercise. Therefore, to determine the precautionary and due diligence measures which all companies are reasonably expected to follow to ensure the security of their products would necessitate an examination of industry practices. Departure from industry custom by a manufacturer could indicate negligence.

It is clear that secure software that is less susceptible to malicious exploitation also best serves the interests of manufacturers, who therefore invest in the security of their products on their own initiative. Brands or products that are perceived to be more secure than those of their competitors, as was the case with Apple operating systems relative to Windows, naturally enjoy a market advantage. Products that cause user inconvenience or present security risks are liable to poor publicity, which in turn reflects in poor sales. However, software vendors cease to voluntarily invest in security once doing so no longer makes sense from a business perspective; Schneider observed that "[i]f we expect software vendors to reduce features, lengthen development cycles and invest in secure software development processes, it needs to be in their financial best interests to do so."[18] Unsurprisingly, the economic incentives that motivate manufacturers diminish once the cost of their efforts to ensure security exceeds the risk undiscovered vulnerabilities may present.

Considering the lack of a legislative reasonable care standard, courts will have

16 Michael L. Rustad, Thomas H. Koenig, *supra* note 4, p. 1571.

17 Liis Vihul et. al., 'Legal Implications of Countering Botnets – Joint report from the NATO Cooperative Cyber Defence Centre of Excellence and the European Network and Information Security Agency (ENISA)' (2012), p. 62.

18 Bruce Schneier, Information Security: How Liable Should Vendors Be? (*Schneier on Security,* 28 October 2004), available at: https://www.schneier.com/essay-073.html.

to establish the due diligence standard in software development when presented with a negligence claim against a manufacturer. Rather than adopting a standard at the level of industry practice, the courts may assess reasonable care at a higher level and a manufacturer may be found negligent even if able to demonstrate that the efforts it takes to minimise design flaws are comparable to those of other developers.

The assertion that industry custom is representative of reasonable care was famously renounced in the 1932 US case The T.J. Hooper wherein the court did not excuse the defendant company for not having radio receiving sets on-board its tugboats. Had it had such equipment, the defendant would have been in a position to hear the forecast predicting a worsening of the weather, and thereby prevent barges carrying cargo from sinking. Even though equipping carriers with radio receivers did not form general industry custom at that time, Judge Learned Hand held that "in most cases reasonable prudence is in fact common prudence; but strictly it is never its measure; a whole calling may have unduly lagged in the adoption of new and available devices. It never may set its own tests, however persuasive be its usages. Courts must in the end say what is required; there are precautions so imperative that even their universal disregard will not excuse their omission."[19] Applying the same principle, software developers should not assume that compliance with prevalent industry practices intended to assure the security of software products will suffice to satisfy the reasonable care standard.

It can be anticipated that in determining the appropriate and reasonable precautions that are required of software developers, courts will take note of documents such as the 2011 CWE/SANS Top 25 Most Dangerous Software Errors, a catalogue developed jointly by software security experts and corporations detailing twenty-five common yet highly dangerous vulnerabilities in software.[20] "[These errors] are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all. The Top 25 list is a tool for education and awareness to help programmers to prevent the kinds of vulnerabilities that plague the software industry, by identifying and avoiding all-too-common mistakes that occur before software is even shipped."[21] It would seem fair to presume that a court would conclude that each software manufacturer should reasonably be expected to take measures to avoid

19 Re Eastern Transportation Co. (The T.J. Hooper), (1932) 60 F.2d 737, 740 (2d Cir.).

20 'CWE/SANS Top 25 Most Dangerous Software Errors' (2011), available at: http://cwe.mitre.org/top25/#Guidance.

21 *Ibid.*

the most common and known programming defects in the products it releases.

At whatever level courts eventually recognise the duty of care, if at all, the challenge which plaintiffs can expect to face is that often it will be very difficult to prove causation, i.e. that their injury was suffered as a result of the manufacturer's breach of the requisite duty of care. To meet this challenge, a plaintiff will first need to be sufficiently technologically savvy to be confident that his or her own actions did not cause or contribute to the harm that ensued. Vulnerabilities in software in and of themselves usually do not generate harm and go unnoticed unless they are so significant that they render the application dysfunctional and unfit for purpose. In the latter case, the law of warranty should adequately handle the issue. By contrast, vulnerabilities in the code become problematic once a malicious third party exploits them. Such cases will demand significant effort on the end-user's part to determine the root cause of the problem. To overcome this evidentiary obstacle, advocacy for strict liability tends to emerge. Under the strict liability doctrine, which is a non-fault based liability regime, liability is imposed on the tortfeasor irrespective of whether the plaintiff has acted negligently.

Strict Liability

The domain of strict liability is generally reserved for inherently abnormally dangerous activities or ultrahazardous activities. These include the production or handling of explosives and radioactive materials, blasting and keeping wild or dangerous animals. If a person suffers injury as a result of these activities, liability attaches to the defendant regardless of whether reasonable care was exercised or even exceeded, involving all feasible precautions. Considering the limited set of circumstances that the strict liability regime governs its extension to software manufacturers is unlikely even if certain public policy considerations support it. In fact, legislators have refused to extend strict liability to donor blood, vaccine, asbestos and fast food providers, despite the potentially dramatic consequences associated with their handling or use. This is because the socially desirable benefits these products provide appear to outweigh their potential negative effects.[22] Moreover, the extra-legal arguments for imposing a negligence-based liability on software manufacturers, such as the potentially industry-chilling effect of indeterminate financial liability, apply in the case of strict liability. Indeed, the fact that the onerous strict liability does not excuse the defendant

22 Seldon J. Childers, Don't Stop the Music: No Strict Products Liability for Embedded Software, 19 *University of Florida Journal of Law and Public Policy* 166 (2008).

even if the latter had taken all necessary precautions exacerbates those effects.

Whether software is regarded as a product or as a service has critical legal consequences. It has long been a contentious issue, challenging legal scholars and practitioners alike, and disparate approaches have been taken in different jurisdictions. Nevertheless, some cases seem clear. For instance, a program sold on a physical medium and equally available to all interested buyers is generally characterised as a product. Cloud applications, by contrast, are usually considered to be a service, as often denoted by the term "software as a service". Similarly, software that is designed on specification for a particular customer to meet unique requirements tends towards characterisation as a service. When software is qualified as a service the possibility of liability for professional malpractice surfaces. This doctrine is again centred on the exercise of due care; liability arises if the programmer has acted negligently, as assessed against reasonable conduct by similarly situated professionals.

When software is considered to be a product, product liability law can attach. Product liability forms a part of tort law, but the specifics and preconditions of its application vary by jurisdiction. Complicating matters is the fact that product liability does not clearly fall under either the traditional negligence-based tort regime or that of strict liability. In the United States, the strict liability approach to products, which began in the1940s, started to be opposed by manufacturers and insurers who faced a "growing tide of products liability litigation."[23] Today, US product liability practice is in flux. It now reflects elements of negligence-based liability and, in general, "seems to be narrowing rather than extending the liability of manufacturers."[24] This approach accords with European Union legislation, under which product liability is a hybrid of negligence-based and strict liability.[25] By this doctrine, the burden of proof is reversed, and it is for the manufacturer, i.e. the defendant, to demonstrate that the defects in product, in this case the software bugs, were unavoidable, even if due care had been exercised. According to Art. 7(e) of the EU product liability directive, an argument in support of the latter claim could be that the state of scientific and

23 *Ibid*., p. 132.

24 *Ibid.*

25 Council Directive 85/374/EEC of 25 July 1985 on the approximation of the laws, regulations and administrative provisions of the Member States concerning liability for defective products. Official Journal L 210, 7 August 1985 pp. 29-33. Amended by Directive 1999/34/EC of the European Parliament and of the Council of 10 May 1999 amending Council Directive 85/374/EEC on the approximation of the laws, regulations and administrative provisions of the Member States concerning liability for defective products. Official Journal L 141, 4 June 1999 pp. 20-1.

technical knowledge at the time when the product was put into circulation did not allow for discovery of the defect.[26] In litigation, this could be a weighty assertion for manufacturers in that specific techniques that exploit a particular vulnerability may be developed only after the software program has been introduced in the market.

Economic Damage as a Basis for Liability

The most problematic aspect of trying to solve the cyber security challenge by subjecting software manufacturers to delictual liability is that neither common nor civil law tends to regard economic damage alone as a sufficient basis for liability. In other words, actors ordinarily bear a duty of care only if their conduct risks physical harm. However, economic damage is the primary type of harm faulty software causes. Examples include the cost of contracting IT professionals to restore the functionality and integrity of a computer or computer network which has been infected due to vulnerabilities in a software application, the financial harm caused by the unauthorised disclosure of sensitive data and the cost of replacing the faulty software programs and licencing alternative applications that are perceived to be more secure.

The law of tort, on the other hand, has traditionally been concerned with actions that result the physical impairment of the human body ("bodily harm") or of property ("property damage"). Bodily harm includes physical injury, illness, disease, impairment of bodily function and death.[27] Bodily harm and property damage as a consequence of cyber operations generally belong to the domain of national security where we can expect actors to possess enhanced cyber capabilities to produce such effects. In this context, however, actions will likely not be followed up by civil lawsuits.

There are two primary reasons for excluding the recovery of purely economic loss from tort law. First, economic loss usually occurs in a non-self-limiting fashion, whereas physical injury is usually restricted in terms of time and scope. For example, while a careless driver threatens harm only to those nearby,[28] defective software can cause pecuniary harm to all its users, of whom there may be millions. Due to the fact that economic losses tend to proliferate in this way, defendants would be subjected to indeterminate liability out of proportion

26 EU Member States could choose whether to opt in to this clause when transposing the directive into national legislation, and most did so.

27 *Supra* note 12, § 4.

28 *Supra* note 3, § 1 comment c.

to their culpability. This prospect would, in turn, generate an exaggerated pressure to limit or avoid the production and release of new software programs altogether, thereby stifling innovation.[29]

Second, risk of economic loss is in principle well suited to allocation by contract. Such risk is therefore generally allocated by contract; recovery occurs through the application of contract law.[30] Parties to a contract enter into it voluntarily, and are therefore in a position to weigh the risks that are involved, and consequently manage them appropriately. The most common means of managing risk is through the purchase of insurance. Indeed, since a contractual relationship between the manufacturer and each licensee already exists, it would be a suitable tool for fostering the regulation of software manufacturer liability, should policy determine that software companies excessively and unfairly limit their liability towards end-users. If so, the issue could be addressed by limiting the manufacturers' right to freedom of contract and requiring them to assume greater responsibility for their software.

Amending legislation to encompass solely economic losses in order to hold software manufacturers liable lacks perspective, as it would require a fundamental reform in tort law, the effects of which would reach far beyond the software industry. However, even if such an approach were pursued, it would not provide a strategic solution to the cyber security conundrum, for every end-user would have to demonstrate economic loss. If the economic damage consists of the procurement of professional IT services to reinstall applications, having the computer's operating system reinstalled or purchasing an alternative licence for a different program, it is likely to be too small, and the associated costs of legal proceedings too high, to encourage the victims to undertake recourse to court against the manufacturer.

Insurance is another problematic aspect which bears on the potential liability of software manufacturers. To be a viable solution, especially in a strict liability legal regime, the risks for which commercial enterprises are held responsible must be insurable.[31] Commentators have opined that the prospect of insurance for software developers is unlikely in light of the extent of the potential liability. Furthermore, the unpredictability of their possible liability makes it impossible to actuarially calculate its cost, which could result in inappropriately expensive

29 *Ibid.*

30 *Ibid.*

31 James A. Henderson, Jr., Why Negligence Dominates Tort, 50 *UCLA Law Review* 390 (2002).

liability insurance.[32] The approach would also negatively affect access to a variety of applications at a fairly low price that users have hitherto enjoyed, since the cost of insurance would reflect in the product price. Users may be unprepared to assume such cost in exchange for a degree of extra security, especially since the potential risks they are facing from insecure software are predominantly monetary and fairly limited in scope.

Inducing Software Developers to Write More Secure Code

Governments are understandably concerned with the highly problematic and worrisome consequences of cyber crime and the threat cyber attacks pose to national security, the root causes of which are to a great extent exploitable software applications. Yet, hoping to mitigate these risks through the imposition of software manufacturer liability enforced by end-users would demonstrate a lack of strategic foresight.

First, delictual liability normally does not support the recovery of pure economic damage, the type that end-users are most likely to suffer. This is the most significant formal roadblock shielding manufacturers from liability. Second, it would be highly complex for either legislatures or courts to articulate the duty of care that applies to software manufacturers because different applications create different expectations of duty of care. For example, the reasonable care that should be exercised when programming a word processing tool is logically less than that for a SCADA system operating critical infrastructure or software that is employed in medical devices. However, if the assumption is that all coding errors should be regarded as equally dangerous because for attackers any vulnerability will suffice to, for example, acquire access into a sensitive system, the price of a less critical program like a word processing tool would rocket due to the extensive precautions that would have to be taken to assure security. End-users would resultantly shoulder a disproportionate share of cyber security transactional costs. Third, even if the material law in theory successfully enabled software manufacturers to be held liable, end-users might not be motivated to exploit this possibility because potential recovery would not outweigh the financial, emotional and temporal burden of judicial proceedings.

With regard to the latter observation, the consequences of an articulated duty of care could prove contradictory, depending on how probable software manufacturers would consider that successful lawsuits were brought against

32 See, e.g., Childers, *supra* note 22, pp. 137, 139, 173.

them. If they viewed this likelihood to be high, the possibility of litigation would create a deterrent effect towards the careless flaws in software code. Yet, the recognition's effects would not be limited to those desirable. It can be anticipated that the prospect of liability would decelerate innovation and in the broader scale reduce employment opportunities and tax revenue, as well as increase the price of software applications. Today, inexpensive software has qualitatively improved societal well-being – health, social engagement, employment, increase in safety, decrease in work load, etc.,[33] and exposure to liability could negatively impact all of these aspects of daily life. By contrast, if manufacturers regarded successful proceedings to be unrealistic, it would not motivate them to invest more in the security of their products.

Assuming for the sake of argument that software companies are not putting a reasonable emphasis on securing their products, the focus should shift from insisting on the civil liability of software manufacturers. A more promising approach is to create incentives to write better code by requiring certain activities and procedures during software development. In fact, various efforts to structure and formalise software development in order to enhance security are already underway, a prime example being the Secure Software Development Life Cycle.[34] Additionally, software products could be required to undergo obligatory penetration testing by a certified penetration testing company prior to their release to the market.

It is incontrovertible that most companies test their products on their own initiative, since it is in their business interest that the programs are fully functional and secure. However, such testing probably only goes so far – once the costs exceed the financial risks, it no longer makes sense to pursue additional precautionary measures. Indeed, today it makes more financial sense for a manufacturer to shorten the production cycles and release new innovative products quickly, and distribute patches once vulnerabilities are discovered. This propensity to rely on patching rather than investing more in the pre-release phase so that obviously insecure products never reach the market is a root cause of cyber insecurity. There is no guarantee that end-users will diligently install each patch as soon as it is provided by the manufacturer, resulting in vulnerable applications and devices constantly being connected to the internet, and thereby susceptible to exploitation.

33 *Ibid*., p. 165.

34 Introduction to Secure Software Development Life Cycle, available at: http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle.

By contrast, if penetration testing was a legal requirement, and an end-user suffered harm as a result of the manufacturer's failure to comply with the norm, the manufacturer would be liable for the resulting loss. Of course, any such approach must be informed by technological and economic reality, and therefore requires careful consideration prior to adoption. But setting specific obligations out in legislation, as opposed to basing them only on industry initiative and thus leaving compliance subject only to industry scrutiny, would also open the door to administrative oversight by the State, and allow for appropriate sanctions in the case of noncompliance. The State, as opposed to individual end-users, is a much more powerful and effective guarantor of national cyber security than end-users, who are primarily concerned with the functionality of software programs in support of their everyday activities.

Cyber security can only be achieved through a multi-stakeholder approach, in which each participant in today's interconnected social construct bears a degree of responsibility. No single player in this system can be expected to carry the full burden of cyber security, or the full blame for cyber insecurity. Responsibility for enhancing the security of software must be shared between manufacturers, who release it into the market; legislators, who are in a position to impose security requirements on the manufacturers; end-users, who must keep the software applications they use up-to-date and patched; and finally also the education system, which has to support awareness among end-users.