MANAGEMENT SCIENCE
Vol. 57, No. 5, May 2011, pp. 934–959
ISSN 0025-1909 | EISSN 1526-5501 | 11 | 5705 | 0934

informs

DOI 10.1287/mnsc.1100.1304

Who Should Be Responsible for Software Security? A Comparative Analysis of Liability Policies in Network Environments

Terrence August
Rady School of Management, University of California, San Diego, La Jolla, California 92093, taugust@ucsd.edu

Tunay I. Tunca
Graduate School of Business, Stanford University, Stanford, California 94305, tunca_tunay@gsb.stanford.edu

In recent years, vendor liability for software security vulnerabilities has been the center of an important debate in the software community and a topic gaining government attention in legislative committees and hearings. The importance of this question surrounding vendor security liability is amplified when one considers the increasing emergence of *zero-day* attacks where hackers take advantage of vulnerabilities before the software vendor has a chance to release protective patches. In this paper, we compare the effectiveness of three software liability policies: vendor liability for damages, vendor liability for patching costs, and government imposed security standards. We find that vendor liability for losses is not effective in improving social welfare in the short run, while liability for patching costs can be effective if either patching costs are large and the likelihood of a zero-day attack is low, or patching costs are small and zero-day likelihood is high. In the long run, when the vendor can invest in reducing the likelihood of security vulnerabilities, loss liability is still ineffective when the zero-day attack probability is high but can increase both vendor investment in security and social welfare when zero-day attack likelihood is sufficiently low. When the zero-day attack probability is high, patch liability is ineffective if user patching costs are large, but partial patch liability can boost vendor investment and improve welfare when patching costs are small. In contrast, in an environment with low zero-day attack probability, full vendor patch liability can be optimal. Finally, comparing the effectiveness of the three liability policies under study, we find that government imposed standards on software security investment can be preferable to both patching and loss liability on the vendor, if zero-day attack likelihood is sufficiently low. However, if zero-day attacks are a common occurrence and patching costs are not too high, partial patch liability is the most effective policy.

Key words: IT policy and management; economics of IS; network economics; enabling technologies; software; liability; zero-day
History: Received November 20, 2009; accepted December 4, 2010, by Sandra Slaughter, information systems. Published online in *Articles in Advance* March 25, 2011.

1. Introduction

As both public and private entities increasingly rely on the Internet, communications systems, and electronically transmitted data to perform essential business functions, the role and importance of regulation in maintaining security is becoming more apparent. The U.S. government has identified protecting the nation's 17 critical infrastructure and key resources as one of four primary goals of its strategy for homeland security and has particularly identified the importance of its cyber infrastructure, which plays a fundamental crosscutting role, to national security and the stability of the economy (U.S. Department of Homeland Security 2007). However, securing the cyber infrastructure presents difficult challenges because it relies on efforts from both the software industry and a large number of dispersed users (National Infrastructure Advisory Council 2003). These challenges and the nature of the threat have been well recognized by authorities. In 2008, in a speech at the Chamber of Commerce, then Homeland Security Secretary Michael Chertoff pointed out the U.S. government's stance as, "While the vast majority of the nation's cyber infrastructure is in private hands, the reality is that its benefits are so widely distributed across the public domain, and so integrated and interdependent in the various different sectors of our economy, that we face clear national security risks and consequences with respect to its protection. No single person or entity controls the Internet or IT infrastructure. There is no centralized node, or database, or entry point. No single person, or company, or government can fully protect it. On the other hand, the failure in even one company, or one link of the chain,

can have a cascading effect of everybody else. That is why protecting our IT systems and networks has to be a partnership in which all of us have to bear our share of responsibility" (Chertoff 2008).

Spurring much of the newfound reflection on how to coordinate securing our cyber infrastructure was the economic impact and extensive penetration of several recent large-scale attacks on networked computer systems running vulnerable software. Widespread attacks like Code Red, SQL Slammer, Sasser, and Conficker cost businesses on the order of billions of dollars, infected hundreds of thousands of computers across the globe, and affected networked systems in a vast range of industries including nuclear power plants, airlines, hospitals, and banks (Moore et al. 2002; Poulsen 2003; Keizer 2004, 2009; Markoff 2009). Worse yet, security vulnerabilities and attacks on those vulnerabilities continue to be on the rise, and, because of the increasing interdependence of networked systems, many believe that the true potential of these attacks remains to be realized (Ghosh 2009).

There has been some initial regulatory steps toward mitigation, especially on arguably the most critical element of the cyber infrastructure, the Internet. The Internet is a system of interconnected computer networks spanning the world and offering a wide variety of capabilities relating to information exchange. It is an important public resource whose interconnectedness is integral to its benefits, but the fact that its underlying networks and computer systems also exhibit *security interdependence* poses great risks. Internet security is clearly in the public's best interest, but, thus far, private companies and end users did not necessarily have adequate economic incentives to incur costly investments to create more secure software products and properly maintained computing systems. In the face of increasing security attacks and the large associated economic losses, stricter regulatory approaches to securing the Internet may be needed (Krebs 2003, Schneider 2005, Lewis 2009). However, to strategically design policies aimed at improving network security, one must develop a better understanding of how decisions of individual actors influence aggregate measures of security and social welfare, as well as how to appropriately influence their decisions toward serving public interest.

One corrective means to address the underlying incentive problems that has received intense debate in the security community is the ownership of liability for network security losses. Many liability advocates believe that software vendors are selling products with excessive security vulnerabilities and need to be held accountable in order to control high social costs that are caused by these vulnerabilities (see, e.g., Collins 2007, Schneier 2008). On the other hand, there is also support against vendor liability because some experts feel that such impositions would unduly punish vendors when hackers are the true culprits (see, e.g., Haase 2004, Asay 2009). Presently, many security attacks exploit known vulnerabilities for which patches are already available. For instance, patches were available for the vulnerabilities exploited by almost all high-profile worms such as the ones mentioned above up to six months in advance of each attack. In virtually all of these cases, large losses could have been mostly avoided by proper patch maintenance by users (Schweitzer 2003).

Notably, the window of opportunity for patching has been decreasing steadily, and *zero-day* attacks have started to emerge on the Internet. A zero-day attack is defined as "an exploit, worm or a virus capable of crippling global web infrastructure either prior to, or within hours of, a public announcement of a computer system vulnerability" (McBride 2005). In September 2006, a critical zero-day vulnerability in Microsoft's implementation of Vector Markup Language (VML) was exploited by cyber criminals. Exploitation kits were being sold in Eastern Europe, which led to thousands of sites being infected and preying on any user who viewed infected images (Singel 2007). A recent attack on another zero-day vulnerability led to roughly 100,000 sites being compromised. Furthermore, the attack wiped out backend data from these sites using SQL injection (Goodin 2009). Several resources are available that provide information regarding zero-day vulnerabilities and security threat levels, including TippingPoint's Zero Day Initiative, the National Vulnerability Database, the SANS Institute, and IBM's X-Force threat reports. For example, zero-day client-side software exploits have grown from being under 20% in 2004 to over 80% in 2008, with a striking 94% of all Web browser-related exploits having public exploits with 24 hours of disclosure (IBM 2008). The SANS Institute (2009) finds that file format vulnerabilities have recently been favored for zero-day attacks. Security researchers have also demonstrated how easy these vulnerabilities can be discovered; they claimed to have found over 20 zero-day vulnerabilities in Mac OS, Microsoft Office, and Adobe Reader using simple techniques (Keizer 2010). These examples demonstrate that different classes of software face different zero-day risks, and, furthermore, system administrators are often aware of which classes tend to exhibit greater zero-day risk, as is currently the case for some of the software products above. In an environment with an increased possibility of zero-day attacks, vendors must take a more progressive role to reduce the damage caused by these exploits. Such a dramatic change to the security landscape will make and is making the liability issues even more highly disputed.

In summary, in the current network environment, there are serious incentive problems among various actors whose decisions impact the overall security of the cyber infrastructure; the risks associated with attacks on this infrastructure are growing in number and potential impact, and the importance of the role of regulation is increasingly understood and debated. However, answering *how* regulation can actuate a shift toward preferable outcomes, such as an increasingly secure cyber infrastructure and higher social surplus associated with these public resources, is not well understood and requires formal analysis. In this paper, our purpose is to begin exploring this important question by analyzing a model that captures both security interdependence and the primary underlying incentives of actors.

We investigate how liability policies can be used to increase Internet security considering the effects of interconnectivity and the resulting interdependence of users' security actions on one another. There are two economic factors, the roles of which we will be focusing on when we explore the effectiveness of the mechanisms. First, users who do not patch impose negative network externalities on other users—the larger the population of users who do not patch the software, the larger the network risk associated with vulnerabilities. Second, patching is a costly undertaking. If a vendor makes a patch available when there is a vulnerability and a user decides to patch her system, testing, and installing that patch brings a cost burden on the user. Thus, many users do not patch their systems even if a patch is available (see, e.g., August and Tunca 2006 and the references therein). Combining this behavior with the network effects discussed above, we see that the costliness of patching and negative security externalities increase the security risk faced by all users; hence, both the value of the software product for users and, consequently, social welfare are reduced.

In light of these factors, we begin by studying the economics of the network environment in a short-run setting, where the security level of the software is taken as given. In the long run, the role of liability on increasing Internet security also impacts the incentives of software providers to adjust the security quality of their products at the design and development stage. Therefore, we subsequently study a long-run setting where vendors can invest in security quality in response to policy. We analyze and contrast three classes of security liability policies, namely, (i) *loss liability policies*, where the software vendor is liable to partially or fully compensate users' losses incurred in case of a zero-day attack; (ii) *patch liability policies*, where the software vendor is held liable to compensate patching costs incurred by users if a vulnerability is discovered before it is exploited; and (iii) *security standards policies*, where regulation enforces a certain standard of security to be achieved by the vendor during software development to mitigate security vulnerabilities. The former two policies can be applicable in both short-run and long-run settings, whereas the last one is only applicable in the long run. We explore the role of zero-day attacks by studying and comparing these policies in two security environments, with low and high zero-day attack likelihood, respectively. For each class of policies and each zero-day environment, we characterize the optimal policy and study its effectiveness. We then compare and contrast their effectiveness and generate policy recommendations based on our analysis. Our goal is to answer the question whether vendor security liability can be an effective method to improve security and the net social value generated by software, and, if so, how.

The rest of this paper is organized as follows. In §2, we provide a review of relevant literature and position our paper. In §3, we present our model. In §4, we present the consumer market equilibrium and the analysis of liability policies in the short-run setting. In §5, we analyze the long-run case, where the vendor can make security investments, and we present our comparisons of zero-day loss liability, patch liability, and security standards policies. We also explore the benefits of a joint policy approach to software security. In §6, we summarize our policy recommendations and discuss their application. Section 7 offers our concluding remarks. All proofs are provided in the online appendix, which is available in the e-companion.[1]

2. Literature Review

Our work contributes to an expanding body of knowledge that examines the various facets of how security vulnerabilities can be managed to increase the value of software that runs in networked environments to society. Ransbotham and Mitra (2009) develop a conceptual model of the process that information security compromises tend to follow. With a unique data set from a managed security service provider, they find empirical evidence to support the model's notion that the compromise process can follow either opportunistic or deliberate paths, which converge. To inhibit progression along the process and avoid attacks, they suggest using vulnerability controls such as patching at early stages. Chen et al. (2011) study how a diversification strategy can help alleviate the negative externalities associated with security interdependence. For such a strategy to be effective, software markets should be characterized by multiple products with few shared vulnerabilities. Zhou and Johnson (2010)

[1] An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

explore the role of professional information security ratings on service providers, customers, and social welfare. They find that implementing such ratings do not always benefit service providers and customers—even those customers who demand highly secure business partners. On the other hand, they find that adopting such ratings benefits social welfare and therefore suggests that information security ratings should be encouraged.

Our work is closely related to economics literature on liability. Starting with McKean (1970b), which initiated a debate on the economics of product liability (see, e.g., Calabresi and Bass 1970, Dorfman 1970, Gilmore 1970, McKean 1970a), a large number of papers analyze product liability in a variety of settings in the economics literature. In a model of product safety, Oi (1973) demonstrates that shifting liability from consumers to producers can actually lead to increased production of the riskier product grade. Spence (1977) analyzes the effect of liability when consumers have misperceptions on probabilistic characteristics of the good such as safety. Epple and Raviv (1978) clarify how optimal liability rules hinge on the information sets of consumers and the availability and type of insurance. Polinsky (1980) compares the efficiency implications of strict liability versus negligence. Shavell (1982) studies the relationship between liability and insurance (both first-party and liability). Kolstad et al. (1990) study the efficiency of ex ante policies such as Pigouvian taxes and safety standards as well as ex post policies such as tort liability. In an empirical study, Viscusi and Moore (1993) inquire whether liability costs indeed provide incentives for introducing safer products or, instead, stifle innovation. They find that liability often increases product research and development intensity, but, at high levels of liability cost, it can reduce innovative activity.

Just as for the safety of traditional products, the use of liability in software security is heavily disputed. Ryan (2003) argues for liability particularly in the case of security software products, whereas Kaner (1997) agrees liability should be a part of the cure. Both papers point to schemes that hold the entity in the best position to fundamentally fix the security problem liable. In contrast, Heckman (2003) argues against liability for software, claiming that, given a wide range of user expertise and unpredictable consequences, imposing liability for software will be very expensive and stifle innovation. Similarly, Armour and Humphrey (1993) suggest that improving the quality of software is the best recourse of action because the large costs of enforcing legal liability would be best put toward these software improvements.

A few papers have begun to explore the important area that lies at the intersection of liability and software security. Kim et al. (2010b) examine the impact of vendor liability for software security losses. They find that software liability is effective at making vendors increase security quality when consumers are heterogeneous in their sensitivities to quality, whereas it is ineffective if consumers are homogeneous. Under full liability, they demonstrate that software quality increases, total security losses decrease, and usage goes down. If consumers have imperfect information concerning losses, then liability can be effective even in the homogeneous case. In a related work, Kim et al. (2010a) investigate how risk sharing of security losses between software vendors and their customers improves overall software quality. They establish that government imposition of vendor risk sharing can lead to more secure, higher-quality software products. Our paper complements the work of Kim et al. (2010a, b) by analyzing liability in a network setting characterized by interdependent risks. In this case, consumers impose negative externalities on one another, which alters the effectiveness of liability schemes. Furthermore, in our model we study consumers' ability to shield themselves from risk as well as reduce the externality they impose on others by maintaining their own systems' security, i.e., patching, in the presence of software security vulnerabilities.

Cavusoglu et al. (2008) develop a model to investigate optimal time-driven patch management. They demonstrate that both cost sharing and loss liability are useful coordination schemes, although the former is more robust, by coordinating the frequencies of vendor and user patching cycles. Studying loss and patch liability mechanisms as well as regulated security standards, our modeling approach is similar to those of August and Tunca (2006, 2008), who examine mechanisms for a software vendor or a social planner to improve network security and increase expected profit and social welfare when consumers independently make patching decisions given the network security risk. August and Tunca (2006) contains a base model of software purchasing and patching in the presence of negative security externalities and patching costs. Thus, in this work, we extend their modeling approach on several important dimensions. First, we introduce the possibility of zero-day attacks in addition to the more traditional attacks on patchable vulnerabilities. Their presence adds an additional negative network externality into the system, which is affected by the entire purchasing population. Hence, its effect is quite distinct from the externality stemming only from unpatched users, and, as a result, interesting changes to consumer market structures arise in a zero-day setting. Second, we endogenize the vendor's security investment decision as it affects the likelihood of vulnerabilities in the software. By incorporating the cost of security investment and its optimal selection, we can explore long-run settings and

better understand how liability can influence security through investment choices. In this aspect, our work is also related to studies on both firm and user investment in security and vulnerability disclosure (see, e.g., Cavusoglu et al. 2007, Arora et al. 2008, Png and Wang 2009, Choi et al. 2010, Ransbotham et al. 2011), but we focus on how the presence of a zero-day security risk landscape modifies behavior under negative security externalities when these investments in security can be undertaken. Third, the focus of the current work is to study the impact of several liability policies: patch liability, loss liability, and security standards. The latter two have never been studied in such a setting, and, although August and Tunca (2006) examine some aspects of patch liability in the absence of zero-day risk and security investment, by including their presence, we obtain new findings related to patch liability and can contrast them with extant results. Finally, we additionally explore joint welfare optimization problems in this work to explore the impact of coupling liability policies.

The use of software in networks with interdependent security risks is a core component of the overall security landscape of the Internet; thus, building an understanding of how to *appropriately* use policy to manage that security risk is critical to protecting our cyber infrastructure. Our study develops a formal approach to study the disputed question on what role liability should assume in these networked environments. In an environment where users protect themselves from security risk by patching and impose negative externalities on the network by not patching, liability can impact incentives for consumer usage and vendor investments in a substantially distinct manner. Furthermore, we examine how unavoidable risk in the form of zero-day attacks affects the efficiency of liability rules. Our goal is to provide insights into how liability and its form will affect the social value associated with network goods as policy makers are forced to take positions on these difficult topics in the face of an increasingly interdependent and risky Internet-based communications environment.

3. Model Description and Equilibrium

Our base model builds on August and Tunca (2006) with the introduction of zero-day attacks. A software product is offered by a vendor to a continuum of consumers whose valuations are uniformly distributed on $\mathcal{V} = [0, 1]$. The software is used in a network setting, thereby exposing purchasing consumers to additional risk associated with its use. There are two distinct types of attacks, one being completely avoidable whereas the other is not. Both types arise because of inherent security vulnerabilities in the software, but they are differentiated based on who discovers the vulnerability, namely, a malevolent hacker or a benevolent "white hat."[2] If a white hat discovers the vulnerability, it is announced to the general public with a software patch being made available by the vendor. Users can therefore shield themselves completely from this type of risk by patching. If a consumer elects not to patch, she is exposed to the risk of an attack by a hacker on this publicly known security vulnerability.

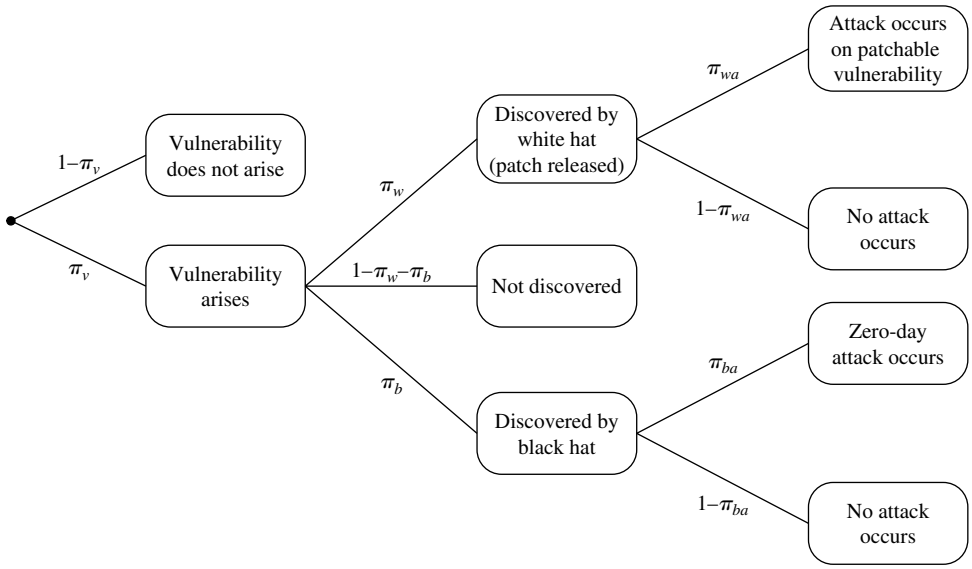
We denote the probability that a security vulnerability arises in the software with $\pi_v > 0$. Given that a vulnerability arises, we denote the probability that a white hat discovers it with $\pi_w > 0$ and the conditional probability that an attack targeting the vulnerability then occurs on the network with $\pi_{wa} > 0$. The total probability of an avoidable attack is thus given by $\pi_a \triangleq \pi_v \pi_w \pi_{wa}$. On the other hand, a hacker may be first to discover the vulnerability and execute a zero-day attack on the user population. Given that a vulnerability arises, we denote the probability that a black hat discovers it with $\pi_b > 0$ and the conditional probability that an attack targeting the vulnerability then occurs on the network with $\pi_{ba} > 0$. Similarly, we denote the total probability of a zero-day attack with $\pi_z \triangleq \pi_v \pi_b \pi_{ba}$. Because such an attack is unavoidable, the entire buying population incurs an expected loss associated with this risk. Figure 1 illustrates the relationship between the events that occur leading up to each type of security attack.[3]

Suppose the software product has a vulnerability. If a user gets struck by a security attack, then one would expect she suffers a loss positively correlated with her valuation. That is, consumers with high valuations will suffer higher losses than consumers with lower valuations because of higher opportunity costs, higher criticality of data, and loss of business. Specifically, the loss that a consumer with valuation v suffers if she is hit by an attack is αv, where $\alpha > 0$ is a constant. It is important to note that α typically takes on high values because, in addition to losses resulting from the inability to use software in case of an attack, there are enormous potential losses for the attacked institution associated with loss

[2] A "white hat" is one who identifies security vulnerabilities in systems or software and notifies system owners and/or vendors of such vulnerabilities to help prevent exploitation by malicious hackers.

[3] Throughout the paper, when we refer to zero-day risk or zero-day attack likelihood, we refer to the absolute probability of a zero-day attack occurring, i.e., π_z, rather than the relative probability of a zero-day attack with respect to a patchable case. Note that, because $\pi_z = \pi_v \pi_b \pi_{ba}$, zero-day risk is then determined by the probability of a vulnerability arising (π_v) as well as the probability of discovery of the vulnerability by a black hat conditional on the existence of the vulnerability (π_b), and the probability that an attack occurs conditional on the first two events (π_{ba}).

Figure 1 Relationship Between Vulnerability, Discovery, and Attack Events



of goodwill and reputation, trust, and future loss of business. Moreover, when an attack occurs and the systems of a company are compromised, a significant amount of work, money, and resources are typically required to bring these systems back up (e.g, many tedious tasks such as software reinstallation, reconfiguration, and data recovery must be undertaken). In some cases, this system reinstitution may even require the hiring of outside experts and consultants. What is more, permanent data loss can also occur, which is extremely costly. Finally, sensitive firm or consumer data can be compromised, and this may actually be the costliest part. (See, e.g., D'Amico 2000, Garg 2003, and Timms et al. 2004 for details on these cost components.) As a result, losses from being hit by an attack often go well beyond the direct loss of usage and can indeed reach extremely high levels (also see, e.g., *Computer Economics* 2004, Cavusoglu et al. 2004).

If the vulnerability is patchable and a consumer chooses to bear the risk, then, denoting the mass of the unpatched population in the network as u, the unconditional probability that she will be hit is $\pi_a u$. However, if a malevolent hacker discovers the vulnerability, we are in the case of unavoidable risk; one cannot typically patch a vulnerability that is unknown to the general public and for which no software patch is currently available. If the mass of the total purchasing (buying) user population is b, then the unconditional probability an unsuspecting user will be hit by an attack is $\pi_z b$.

In the base case, there are three decision periods. In the first period, the vendor sets a price $p > 0$ for a single-server license. In the second period, given the price and risk characteristics associated with the product, each consumer makes a decision whether or not to purchase the software. In the third period, if a security vulnerability has been discovered by either the vendor or another white hat, a patch is made available, and each purchasing consumer makes a decision whether or not to patch her software. If a consumer chooses to patch the software when these opportunities arise, she will incur an expected cost of patching denoted $c_p \triangleq \pi_v \pi_w c_p^\circ$, where $c_p^\circ > 0$ accounts for the money and effort that she must exert in order to verify, test, and roll out patched versions of existing systems. Patch management software is often utilized and can be helpful, but there always exist patching costs as part of a sound patch management strategy (Bloor 2003, Beres and Griffin 2009).[4] Systematic testing and deployment of patches also enables companies to avoid potential patching failures such as the recent McAfee antivirus update incident that caused XP machines to crash and enter a reboot cycle (McCullagh 2010). Finally, after the third stage, an attack may realize in which case consumers incur losses.

Each consumer makes a purchasing decision to either buy, B, or not buy, NB, the software product. Similarly, if a vulnerability arises in the software, each user makes a decision to either patch, P, or not patch, NP, her own system. We denote the consumer action space by $S = \{B, NB\} \times \{P, NP\} - (NB, P)$, the

[4] The magnitude of patching costs is related to the class of software in question. For example, patching systems running enterprise software such as IBM WebSphere, Microsoft SQL Server, and Oracle tends to be more costly (high c_p) than testing lightweight client applications (low c_p). In the latter case, users may not use as rigorous of a patching process as described above and often followed by system administrators in enterprise settings. See §7 for a more extensive discussion of patching costs.

last exclusion stemming from (NB, P) clearly being infeasible. Given a fixed price, in a consumer market equilibrium, each consumer maximizes her expected utility given the equilibrium strategies for all consumers. For a strategy profile σ: $\mathcal{V} \to S$, if a security vulnerability arises in the software, the expected cost faced by the consumer with valuation v is then defined by

$$\Psi(v, \sigma) \triangleq \begin{cases} \pi_a \alpha u(\sigma) v + \pi_z \alpha b(\sigma) v & \text{if } \sigma(v) = (B, NP), \\ c_p + \pi_z \alpha b(\sigma) v & \text{if } \sigma(v) = (B, P), \\ 0 & \text{if } \sigma(v) = (NB, NP), \end{cases} \tag{1}$$

where

$$u(\sigma) = \int_{\mathcal{V}} \mathbf{1}_{\{\sigma(v) = (B, NP)\}} \, dv, \tag{2}$$

and

$$b(\sigma) = \int_{\mathcal{V}} \mathbf{1}_{\{\sigma(v) \in \{(B, NP), (B, P)\}\}} \, dv. \tag{3}$$

Consumers who buy but do not patch cause a negative externality on all users by decreasing the safety of the network and the software. Furthermore, because of the risk of zero-day attacks, even consumers who would patch if a patch were available cause a negative externality on all users by increasing the unavoidable part of the risk due to increased usage. Both of these externalities are reflected in (1), where $\Psi(v, \sigma)$ is increasing in both $u(\sigma)$ and $b(\sigma)$. Given the degrees of freedom in the attack model, we can reduce the exogenous parameter space of the base model to α, $c_p \in (0, 1)$ and π_a, $\pi_z \in (0, 1)$, noting that because of the natural dependence between the two types of security risk, $\pi_a + \pi_z < 1$ must be satisfied.[5] For ease of exposition, we mostly refer to reduced form parameters going forward. In practice, for a particular setting, the reduced form parameters can be computed from the underlying primitives as defined above.

3.1. Consumer Market Equilibrium

Taking the security characteristics of the software along with its price as fixed, we first investigate how consumers' usage and patching decisions give rise to the consumer market equilibrium. We denote the equilibrium strategy profile with σ^*. Thus, for the consumer with valuation v, the action $\sigma^*(v)$ provides a greater expected payoff than any other possible action from S when fixing all other consumers' actions to that prescribed by σ^*. The characterization of σ^* is provided in the following lemma. When we analyze security investment and liability policies in later sections, their impact to consumer trade-offs and hence the equilibrium strategy profile appear through adjustments to the *effective parameters* seen in the lemma. How each policy specifically affects these parameters will be addressed in their respective sections.

Lemma 1. *Given α, the consumer price $p \in (0, 1]$, and effective parameters $\tilde{\pi}_a$, $\tilde{\pi}_z$, and $\tilde{c}_p$, there exists a unique equilibrium in the consumer market. The equilibrium consumer strategy profile is characterized by v_b, $v_p \in [0, 1]$ and $v_b \le v_p$ such that, for $v \in \mathcal{V}$,*

$$\sigma^*(v) = \begin{cases} (NB, NP) & \text{if } 0 \le v < v_b, \\ (B, NP) & \text{if } v_b \le v < v_p, \\ (B, P) & \text{if } v_p \le v \le 1. \end{cases} \tag{4}$$

Let $\bar{p} \triangleq (1 - \tilde{c}_p(1 + \tilde{\pi}_z/\tilde{\pi}_a))^+ (1 - \tilde{c}_p/(\tilde{\pi}_a \alpha))^+$. Given (4), the patching behavior is characterized by two regions in the parameter space:

Region I: *If $\tilde{c}_p \le \min(\tilde{\pi}_a \alpha, \tilde{\pi}_a/(\tilde{\pi}_a + \tilde{\pi}_z))$ and $p < \bar{p}$, then $p < v_b < v_p = \tilde{c}_p v_b/(v_b - p - \tilde{\pi}_z \alpha (1 - v_b) v_b) < 1$.*

Region II: *If $\tilde{c}_p > \min(\tilde{\pi}_a \alpha, \tilde{\pi}_a/(\tilde{\pi}_a + \tilde{\pi}_z))$ or both $\tilde{c}_p \le \min(\tilde{\pi}_a \alpha, \tilde{\pi}_a/(\tilde{\pi}_a + \tilde{\pi}_z))$ and $p \ge \bar{p}$, then $0 < p < v_b < v_p = 1$.*

Lemma 1 establishes that low effective patching costs and zero-day likelihood combined with a high security-loss factor drives consumers toward patching in equilibrium. Furthermore, Lemma 1 also establishes that under broad conditions, patching can be cost inefficient and not a well-represented behavior in equilibrium. Particularly, as zero-day attacks become increasingly likely, more consumers optimally remain unpatched, which can cause greater security risk stemming from the patchable vulnerabilities.

4. Vendor Liability

To reduce Internet security risk and improve overall welfare, security advocates argue that policies that hold software vendors liable for some or all of the costs incurred by users as a result of poor inherent software security can improve the status quo (Ryan 2003, Schneier 2004, Schneier 2008). We study two policies of vendor liability placed on the costs incurred by consumers.

First, we examine patch liability, which is an incentive scheme that holds the vendor responsible for compensating consumers who incur patching costs to protect themselves from risk when a patchable security vulnerability arises in the software. Specifically, each consumer that patches receives a transfer payment of $\lambda_p c_p$ from the vendor, where $\lambda_p \in [0, 1]$ is the effective share of patching costs for which the

[5] Particularly, given any α, c_p, π_a, and π_z satisfying these conditions, there exist a set of underlying parameters: π_v, π_w, π_{wa}, π_b, π_{ba}, and c_p°, which gives rise to the reduced form parameters.

vendor is liable. A planner sees the following trade-off: the higher the vendor's patch liability, the larger the population of consumers who patch. A larger patching population effectively increases the security of the software because of interdependence, which reduces the riskiness of the software and, in turn, may increase the size of the population of users and the social value generated. On the other hand, holding the vendor liable for patching costs may drive higher prices under profit maximization, which can ultimately depress usage. Under patch liability, the effective patching costs faced by any user is reduced by the vendor's liable portion to $\tilde{c}_p = (1-\lambda_p)c_p$.

Second, we examine loss liability on zero-day security attacks. In the case of a zero-day attack, a security vulnerability for which the vendor has not yet issued a patch is exploited. We analyze a liability scheme that holds the vendor liable for a fraction $\lambda_z \in [0, 1]$ of the consumer losses associated with a zero-day attack. The reduction in user risk exposure to these losses is reflected in the consumer market equilibrium by decreasing the effective zero-day likelihood parameter by the liable portion, i.e., $\tilde{\pi}_z = (1-\lambda_z)\pi_z$.

We start by studying the *short-run* capability of optimal liability schemes to coordinate incentives as it relates to the trade-offs discussed above; our aim is to provide insight into where the efforts of policy should be focused for existing products exhibiting security interdependence and risk-shielding consumers. Another important question is what will be the impact of these liability schemes in a longer time horizon where vendors have the opportunity to invest in the security of their products. In §5, we endogenize the vendor's security investment decision and discuss the relevant trade-offs and implications.

4.1. Profit and Social Welfare Maximization

For a fixed liability scheme, we can formulate the vendor's profit-maximization problem with minor reparameterization of the consumer market equilibrium. The vendor's profit function is

$$\Pi(p, \lambda_\tau) \triangleq p(1-v_b) - \lambda_\tau L_\tau, \tag{5}$$

where

$$L_\tau \triangleq \begin{cases} \displaystyle\int_{v_p}^{1} c_p \, dv & \text{if } \tau = p, \\ \displaystyle\int_{v_b}^{1} \pi_z \alpha (1-v_b) v \, dv & \text{if } \tau = z, \end{cases} \tag{6}$$

noting that L_p measures the aggregate consumer losses associating with patching and L_z measures the aggregate losses associated with zero-day attacks. We use a subscript $\tau \in \{z, p\}$ to denote the specific liability policy under consideration; $\tau = z$ refers to a zero-day loss liability policy, and $\tau = p$ refers to a patch liability policy. The vendor's profit-maximization problem can be written as

$$\begin{aligned} \max_{p \in [0,1]} \ & \Pi(p, \lambda_\tau) \\ \text{s.t.} \ & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p, \lambda_\tau), \end{aligned} \tag{7}$$

where $\sigma^*(\cdot \mid p, \lambda_\tau)$ indicates that λ_τ affects the equilibrium strategy profile (through $\tilde{c}_p$ and $\tilde{\pi}_z$ as discussed above). Subtracting the security losses for which the vendor may be held liable from the net value derived from usage of software with patchable risk, our measure of social welfare is given by

$$\begin{aligned} W(\lambda_\tau) &\triangleq \int_{v_b}^{1} v(1 - \pi_a \alpha (v_p - v_b) \mathbf{1}_{\{v \le v_p\}}) \, dv - L_p - L_z \\ &= \tfrac{1}{2}\big((1-v_b^2)(1-\pi_z\alpha(1-v_b)) - 2c_p(1-v_p) \\ &\qquad - \pi_a \alpha (v_p - v_b)(v_p^2 - v_b^2)\big), \end{aligned} \tag{8}$$

noting again that λ_τ affects v_b and v_p, which are governed by σ^*. The social planner's problem is that of determining the liability share that maximizes welfare, and, mathematically, the planner's problem can be formulated

$$\begin{aligned} \max_{\lambda_\tau \in [0,1]} \ & W(\lambda_\tau) \\ \text{s.t.} \ & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p^*(\lambda_\tau), \lambda_\tau) \\ & p^*(\lambda_\tau) \text{ solves (7)}. \end{aligned} \tag{9}$$

4.2. Short-Run Liability Analysis

In the short run, liability policies can benefit welfare substantially if they are effective at inducing greater usage or patching populations. Notably, zero-day loss liability and patch liability engender usage in distinct ways. A patching liability policy aims to reduce the size of the unpatched population, which, in turn, reduces the negative security externality thereby providing incentives for more consumers to become users. A zero-day loss liability policy directly impacts the loss side of a consumer's trade-off while indirectly affecting patching behavior. In this section, we explore whether liability can help coordinate a system involving network security risk toward better outcomes, and, from a welfare perspective, we clarify the optimal structure of effective liability schemes.

Our study focuses on two broad security environments, namely, those characterized by low and high zero-day attack risk. In the first regime, the threat of zero-day attacks is low, hence security externalities stemming from unpatched users on the network play the predominant role. In contrast, in the second regime, the threat of zero-day attacks is high, hence security externalities from *all* usage become significant as well. In addition, we focus on high potential

security loss environments, i.e., environments where the parameter α is high. We do this because, first, such environments more accurately reflect the reality of network software security. Second, it can be shown that liability policies are unnecessary and ineffective in most low potential security loss environments.[6]

In the following proposition, we demonstrate that zero-day loss liability is not effective in either regime during the short run where the vendor cannot invest to improve security in his software product.

Proposition 1. *In the short run, welfare is strictly decreasing in zero-day loss liability in both regimes.*[7]

By placing some zero-day loss liability on the vendor ($\lambda_z L_z$), the consumer with valuation v will use the software product if and only if $v \geq p + (1-\lambda_z)\cdot \pi_z \alpha b(\sigma^*)v + \pi_a \alpha u(\sigma^*)v$. As seen in the right side of this inequality, the direct effect of zero-day loss liability is to reduce the purchasing threshold v_b (i.e., increase usage), which can increase welfare. However, a reduction in v_b will cause $b(\sigma^*)$ and $u(\sigma^*)$ to increase, which creates larger security externalities on the network and hurts welfare. Furthermore, forcing the vendor to bear a λ_z share of zero-day losses directly affects his profit-maximization problem in (7) and leads to increased prices. As Proposition 1 states, in the short run, the net effect of such a policy is that it reduces welfare. Regardless of regime, we find that, in response to zero-day liability, the vendor aggressively raises his price to restrict usage. Therefore, a social planner's attempt to increase welfare by stimulating usage (which tends to be lower in high potential security loss environments) via the imposition of vendor zero-day liability ($\lambda_z > 0$) backfires, leading to a reduction in software usage. In this case, the purchasing threshold increases to an extent that welfare decreases despite any security benefits from indirect network effects associated with a reduced user population. In §5.2, we will contrast this result with the efficacy of zero-day liability when it can be used to target the *investment* incentives of a software vendor.

Unlike zero-day loss liability, the use of patch liability in a short-run setting is often helpful at substantially increasing social surplus. As the following proposition establishes, one surprising finding is that whether patch liability is effective under high or low patching costs ultimately hinges on the magnitude of zero-day likelihood, and its effect yields opposing outcomes.

[6] An analysis of low potential security loss environments and the demonstration of ineffectiveness of liability are available from the authors upon request.

[7] A technical statement of this proposition can be found in the online appendix. For the remainder of the paper, technical details can be found in the online appendix whenever they are omitted from the body of the paper.

Proposition 2. *For high zero-day risk environments, patch liability is effective under low patching costs satisfying $c_p < \pi_a/(\pi_a + \pi_z)$, and the optimal liability share, i.e., λ_p^*, tends toward $1/2$ for sufficiently large α. For low zero-day risk environments, patch liability is only effective under high patching costs satisfying $c_p > 6 - \sqrt{33}$, and the optimal liability share tends toward $(12c_p - 3 - c_p^2)/(16c_p)$.*

Proposition 2 establishes that when zero-day likelihood is high, patch liability is effective if patching costs are low. When potential security loss and zero-day likelihood are high, the entire purchasing population, $b(\sigma^*)$, causes a significant negative externality on all users of the software. Even those users who are properly patched incur security losses because, unlike risk associated with patchable vulnerabilities, zero-day risk cannot easily be shed. In equilibrium, the consumer with valuation v will patch if and only if $(1-\lambda_p)c_p \leq \pi_a \alpha u(\sigma^*)v$ is satisfied, hence there are several effects. When patching costs are low (smaller c_p), there exist clear incentives for users to patch because the above inequality is satisfied for more consumers, all else being equal. On the other hand, because of zero-day risk, which users cannot protect themselves from by patching, the aggregate purchasing population tends to be small and consists mostly of high valuation users. A smaller user population tends to be associated with a smaller unpatched population in equilibrium, which, in turn, reduces incentives for users to patch systems because expected security losses associated with patchable vulnerabilities is proportional to $u(\sigma^*)$. When the latter effect is substantial, the *proportion* of individuals patching in a high zero-day risk world can be much smaller than that seen in a low zero-day risk world. In this case, liability on patching costs can be quite effective at inducing some of the unpatched proportion of users to patch. In contrast, when patching costs are large, users tend not to patch because it is expensive and offers no protection from the relatively high zero-day risk. To induce some of these users to start patching, the liability shares to be covered by the vendor must be large. However, too much exposure to users' patching costs negatively affects the vendor's profits, and, therefore, he will raise his price to serve a smaller portion of high valuation consumers. Because, in this case, the indirect effects of a liability policy cause raised prices and smaller user populations, its net effect on welfare is negative.

In contrast, when zero-day likelihood is low, Proposition 2 states that patch liability is only useful if patching costs tend to be large. In this case, under high potential security loss and patching costs as well as the vendor's optimal pricing, the equilibrium patching population is also small, but, in contrast to the case just discussed above, the purchasing population is now much larger because of the lower zero-day

risk. Because there is also a large unpatched population under these conditions, a patch liability policy can force the vendor to assume a portion of the patching costs, which helps to increase the size of the patching population and benefit social welfare. Thus, by comparing these two outcomes, we find that regimes with high potential security loss behave fundamentally different when zero-day attacks become a larger portion of that risk. Proposition 2 demonstrates this result and establishes that even the same policy (patch liability in this case) must be strategically tailored to the environment being addressed.[8]

5. Vendor Investment in Security

In the last section, we observed that, in a network environment and short-run horizon, loss liability is ineffective at increasing welfare whereas liability on patching costs can help increase software security and the social benefit obtained from software under certain market conditions. A common argument in favor of imposing liability on software vendors for security losses is that liability can induce them to increase their investment in the security of software, specifically toward debugging, testing, and fixing security holes (Schneier 2004, Werth 2009). What are the effects of different liability schemes on security investments and consequently social welfare in a long-run analysis with network security externalities is an open question. To explore this issue, we first introduce a vendor's investment in the security of his software product. We then examine how he optimally invests in security given the risk profile of the software and network environment as well as the consumer market equilibrium stemming from this risk profile. Finally, we analyze the impact of liability schemes on security investment and welfare and assess the desirability of these schemes.

By investing in security, a software vendor can discover and fix security holes, hence reducing the likelihood that a security vulnerability arises in his product. To reduce the probability of a security vulnerability occurring in the software by $\beta \in [0, 1]$, the firm must invest $C(\beta)$, where C: $[0, 1) \to \mathbb{R}_+$, $C(0) = 0$, $C'(\cdot) \geq 0$, $C''(\cdot) \geq 0$, $C'''(\cdot) \geq 0$, and $\lim_{\xi \to 1} C(\xi) = \infty$, which are common cost structure assumptions used in many settings (see, e.g., Laffont and Tirole 1993). To avoid trivialities in the analysis and maintain conciseness, we also assume that $C'(0) < c_p(1-c_p)/2$. A security investment of $C(\beta)$ directly reduces the probability that a security vulnerability arises from π_v to $\pi_v(1-\beta)$. Thus, by the definition of π_z and π_a, the effective likelihood of a zero-day attack and an attack on a patchable vulnerability are, in turn, reduced to $\pi_z(1-\beta)$ and $\pi_a(1-\beta)$, respectively. Similarly, such an investment reduces the effective likelihood a patchable vulnerability appears later, hence, by the definition of c_p, the expected patching cost for a consumer at the time of purchase is reduced to $c_p(1-\beta)$.

Taking into account the effect of this reduction on the resulting consumer market equilibrium, the vendor makes an investment in β and, further, determines the optimal β that maximizes his expected profit. In the *base case* with no liability imposed, the vendor's problem can be written as

$$\begin{aligned} \max_{p,\,\beta \in [0,1]} \quad & \Pi(p, \beta) \triangleq p(1 - v_b) - C(\beta) \\ \text{s.t.} \quad & v_b \text{ satisfies } \sigma^*(\cdot \mid p, \beta), \end{aligned} \tag{10}$$

where $v_b = v_b(p, \beta)$ is the purchasing threshold that emerges in the consumer market equilibrium under the investment induced β and software price p. In the remainder of this section, we will explore the effectiveness of the liability mechanisms under consideration and characterize how each of them should optimally be deployed. Next, we compare performance across the mechanisms to establish the conditions under which each type of liability is preferable.

5.1. Long-Run Liability Analysis

Incorporating security investment and extending the formulation in (8), for analysis in a long-run setting, social welfare is given by

$$\begin{aligned} & W(\lambda_\tau, \beta) \\ & \quad \triangleq \int_{v_b}^{1} v\big(1 - (1-\beta)\pi_a \alpha (v_p - v_b)\mathbf{1}_{\{v \leq v_p\}}\big)\, dv \\ & \qquad - (1-\beta)(L_p + L_z) - C(\beta) \\ & \quad = \tfrac{1}{2}\big((1 - v_b^2)(1 - (1-\beta)\pi_z \alpha (1 - v_b)) \\ & \qquad\quad - 2(1-\beta)c_p(1 - v_p)\big) \\ & \qquad - \tfrac{1}{2}\big((1-\beta)\pi_a \alpha (v_p - v_b)(v_p^2 - v_b^2)\big) - C(\beta). \end{aligned} \tag{11}$$

Under a fixed liability scheme, λ_τ, $\tau \in \{p, z\}$, the vendor's profit function is adapted to

$$\Pi(p, \beta, \lambda_\tau) \triangleq p(1 - v_b) - \lambda_\tau L_\tau - C(\beta), \tag{12}$$

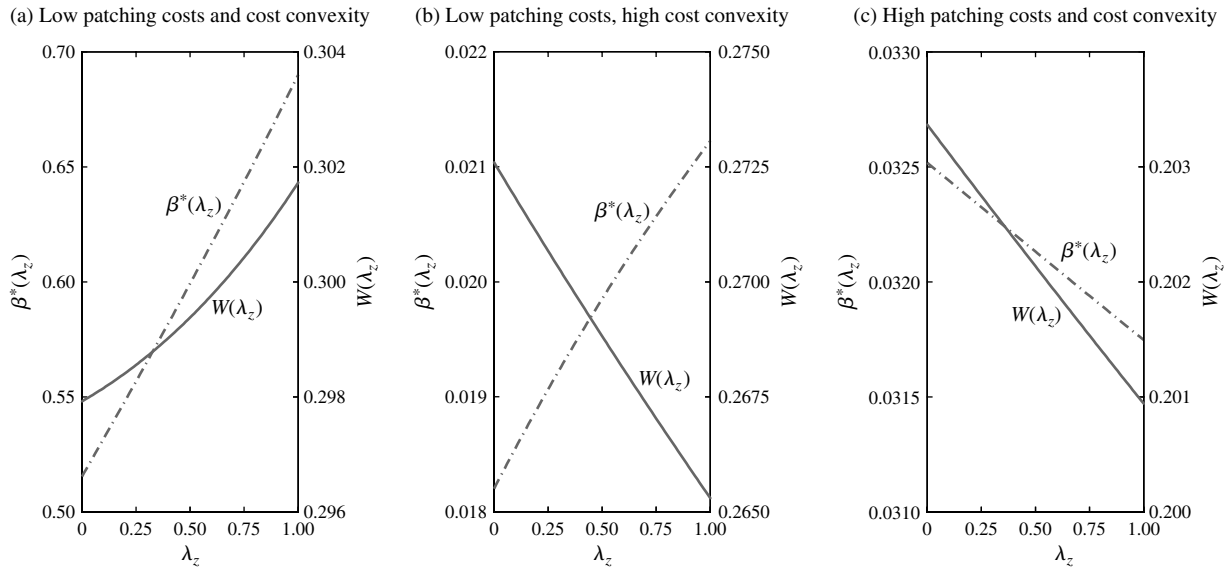
and his profit-maximization problem becomes

$$\begin{aligned} \max_{p,\,\beta \in [0,1]} \quad & \Pi(p, \beta, \lambda_\tau) \\ \text{s.t.} \quad & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p, \beta, \lambda_\tau). \end{aligned} \tag{13}$$

Thus, the social planner's liability share optimization problem now reflects how the software vendor

[8] In the next section, we study these same two policies (patch and loss liability) in addition to security standards and subsequently examine joint policies to see if there are benefits of combining them. For short-run settings, Proposition 1 establishes that loss liability is ineffective at any level of patching costs; thus, a joint approach cannot increase welfare beyond the use of a patch liability policy alone.

Figure 2 Welfare and the Vendor's Security Investment Under a Zero-Day Loss Liability Policy as a Function of the Vendor's Liability Share, λ_z



Notes. The security investment cost function for all panels is $\Gamma(\beta)$. For panel (a), $c_p = 0.14$ and $\delta = 0.02$. For panel (b), $c_p = 0.14$ and $\delta = 0.50$. For panel (c), $c_p = 0.50$ and $\delta = 0.50$. The remaining parameter values for all panels are $\alpha = 2.5$, $\pi_a = 0.30$, $\pi_z = 0.05$, $k = 0.04$, and $\hat{\beta} = 0.80$.

accounts for liability in both his pricing and investment, which is formally given by

$$\begin{aligned} \max_{\lambda_\tau \in [0,1]} \quad & W(\lambda_\tau, \beta^*(\lambda_\tau)) \\ \text{s.t.} \quad & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p^*(\lambda_\tau), \beta^*(\lambda_\tau), \lambda_\tau) \\ & (p^*(\lambda_\tau), \beta^*(\lambda_\tau)) \text{ solve (13).} \end{aligned} \tag{14}$$

Note that $W(0, \beta^*(0))$ corresponds to the welfare outcome in the benchmark no liability case, and $W(\lambda_\tau, 0)$ corresponds to the welfare outcome in the short run.[9]

5.2. Loss Liability

We begin by studying zero-day loss liability ($\tau = z$). Accounting for both investment and liability shares, the effective parameters for the consumer market equilibrium are given by $\tilde{\pi}_z = \pi_z(1-\beta)(1-\lambda_z)$, $\tilde{\pi}_a = \pi_a(1-\beta)$, and $\tilde{c}_p = c_p(1-\beta)$. For this case, we denote the optimal solution to (14) by λ_z^*, and we start by investigating the impact of a loss liability policy on the vendor's security investment and its effectiveness on improving social welfare for a low zero-day likelihood environment. The following proposition demonstrates that zero-day loss liability can improve welfare when the software vendor accordingly invests to improve the security of his software. However, increased investments alone are not sufficient.

[9] Note that $\beta^*(0)$, the vendor investment under no liability, typically will be strictly positive as the vendor would invest in security to improve the security of the product and keep its price high even without any zero-day or patch liability imposed on him.

Proposition 3. *Denote the unique solution to the equation*

$$c_p(1 - c_p(1-z)) - 2C'(z) = 0 \tag{15}$$

in $(0, 1)$ *by* z^*.

(i) *In low zero-day risk environments, if the rate of increase in the marginal cost of security investment is*

(a) *sufficiently low, i.e.,* $C'(z^*)/C''(z^*) \geq 3(1-z^*)$, *then both welfare and vendor investment in security increase with loss liability;*

(b) *sufficiently high, i.e.,* $C'(z^*)/C''(z^*) < 3(1-z^*)$, *then loss liability is ineffective. Furthermore, if either patching costs or the marginal costs of security investment are low, i.e.,* $c_p < 1/4$ *or* $C'(1 - 1/4c_p) < 3c_p/8$, *then vendor investment in security increases while welfare decreases in* λ_z. *Otherwise, both welfare and vendor investment in security decrease in* λ_z.

(ii) *In a high zero-day risk environment, with an increase in loss liability* (λ_z), *the vendor's security investment either decreases or stays constant, and welfare decreases, i.e.,* $\lambda_z^* = 0$.

Figure 2 illustrates part (i) of Proposition 3 for the cost function

$$\begin{aligned} C(\beta) &\triangleq \Gamma(\beta) \\ &= (k\beta + \delta\beta^2) \cdot \mathbf{1}_{\beta \in [0, \hat{\beta})} + \left(\frac{(k\beta + \delta\beta^2)(1-\hat{\beta})}{1-\beta} \right. \\ &\qquad \left. + \frac{(k\hat{\beta} + \delta\hat{\beta}^2)(\hat{\beta} - \beta)}{1-\hat{\beta}} \right) \cdot \mathbf{1}_{\beta \in [\hat{\beta}, 1)}, \end{aligned} \tag{16}$$

which satisfies all requirements stated in the description of the security investment model.[10] As we discussed above, a major argument in favor of imposing security liability on a software vendor is that liability can increase his incentives to invest in software security to create a more secure product, as the vendor would want to avoid payments to consumers in case of a security breach (Ryan 2003, Cusumano 2004, Schneier 2008). On the other side of the argument, opponents of vendor security liability argue that imposing liability on vendors reduces their upside potential for the software, thereby stifling software investment (Heckman 2003, Joyce 2005, Ho 2009). In settings where vendor investment in software security cannot be easily impacted, as in the short-run analysis presented in §4, we have shown that loss liability is not an effective tool to improve welfare. However, loss liability can be effective in long-run settings when applied judiciously under appropriate market conditions. Specifically, part (i)(a) of Proposition 3 shows that, when factoring in the vendor's security investment, *full* zero-day loss liability (i.e., $\lambda_z^* = 1$) can induce increased investment and lead to higher welfare. This result is also illustrated in panel (a) of Figure 2. Generally, for loss liability to be effective, the vendor's marginal cost of improving software should not increase too steeply.

On the other hand, as part (i)(b) of Proposition 3 establishes, vendor loss liability can also backfire in the long run as well. When the rate of increase in a vendor's marginal cost of improving software security is high, imposing loss liability can decrease welfare, and the optimal action becomes to impose no loss liability (i.e., $\lambda_z^* = 0$). Importantly, placing loss liability on the vendor can decrease social welfare *even if it induces increased vendor investment* in software security. As can be seen in panel (b) of Figure 2, when patching costs are low, increased liability can induce the vendor to increase security investments because he can transfer these security improvement costs relatively more easily to consumers. Nonetheless, the vendor's reflection of his investment and liability costs on the consumer price still draws down welfare. When patching costs are high, it becomes difficult for the vendor to pass the additional investment and liability costs to consumers. The profitability of the software for the vendor declines significantly; the upside of investing diminishes; as a consequence, both his investment in security and welfare decrease with imposed liability, as can be observed in panel (c) of Figure 2.

Part (ii) of Proposition 3 demonstrates that zero-day risk can have a significant impact on the effectiveness of loss liability schemes. Particularly, when zero-day risk is high, although the vendor has greater incentives to invest to cover losses from liability payments, the heightened level of those payments draws down the profitability of the software significantly and ultimately makes increasing investment in software security improvements not worthwhile. Rather, the reduced profitability of the software causes the vendor to spend even less on developing it. In addition, he reflects the liability losses through the consumer price. These two effects combine to result in a reduction of welfare with the introduction of vendor zero-day loss liability.

5.3. Patch Liability

In §5.2, we saw that loss liability is, in many cases, ineffective in improving welfare. In fact, contrary to the claims of some liability advocates, imposing liability on the vendor can reduce his incentives to invest in software security. In this section, we examine how patch liability impacts the vendor's incentives for security investments and whether, in the long run, it can be helpful in improving welfare.

Again, we begin with an analysis of low zero-day risk environments. For convenience, we first define

$$w_1 = \frac{c_p(1-c_p^2)}{2(3-c_p)} \quad \text{and} \quad w_2 = \frac{c_p(1+c_p)G(c_p)}{2H(c_p)}, \tag{17}$$

where $G(c_p) = 1 - 6c_p + c_p^2$ and $H(c_p) = 3 - 12c_p + c_p^2$.

Proposition 4. *In a low zero-day risk environment, when* $C''(0) > c_p^2/2$, *and*

(i) *if* $C''(0) \le w_1$, *then patch liability is only effective under low patching costs satisfying* $c_p \le 1/3$, *taking the form of full liability. Furthermore, under the optimal policy, the vendor's security investment increases relative to the outcome in the absence of patch liability, i.e.,* $\beta^*(\lambda_p^*) > \beta^*(0)$;

(ii) *if* $w_1 < C''(0) \le w_2$, *then patch liability should not be employed*;

(iii) *if* $C''(0) > w_2$, *then patch liability increases welfare only under high patching costs satisfying* $c_p > 6 - \sqrt{33}$. *The optimal policy implies partial patch liability, i.e.,* $\lambda_p^* = \lambda_0$ *for a* $\lambda_0 \in [0, 1)$ *satisfying*

$$\lambda_0 = \frac{2H(c_p(1-z^*))C''(z^*) - c_p((1-z^*)^{-1} + c_p)G(c_p(1-z^*))}{8(c_p^2(1+c_p(1-z^*)) - 4c_pC''(z^*))}, \tag{18}$$

where z^* *is defined in* (15). *Under the optimal policy, the vendor decreases his security investment compared to the case with no liability as a result, i.e.,* $\beta^*(\lambda_p^*) < \beta^*(0)$.

[10] Clearly, there are many functions that satisfy the commonly used requirements we listed for the investment cost function. Throughout the paper in all figures, we will use this particular function only because it is easy to handle numerically. None of our numerical demonstrations or our related observations are specific to this functional form in any way.

Table 1 Optimal λ_p and Its Effect on Vendor Security Investment for Low Zero-Day Risk Environments

	$C''(0) \le w_1$	$w_1 < C''(0) \le w_2$	$w_2 < C''(0)$
$0 \le c_p \le 6-\sqrt{33}$	$\lambda_p^* = 1, \quad \beta^* \uparrow$	$\lambda_p^* = 0$	$\lambda_p^* = 0$
$6-\sqrt{33} < c_p \le 1/3$	$\lambda_p^* = 1, \quad \beta^* \uparrow$	$\lambda_p^* = 0$	$\lambda_p^* = \lambda_0, \quad \beta^* \downarrow$
$1/3 < c_p \le 1$	$\lambda_p^* = 0$	$\lambda_p^* = 0$	$\lambda_p^* = \lambda_0, \quad \beta^* \downarrow$

Notes. Results are grouped into regions of c_p and $C''(0)$ as identified in Proposition 4. The notation $\beta^* \uparrow (\downarrow)$ indicates an increase (decrease) in the vendor investment in software security with the optimal policy compared to the vendor investment in security at the base case where $\lambda_p = 0$. Note that λ_0 is as defined in Proposition 4.

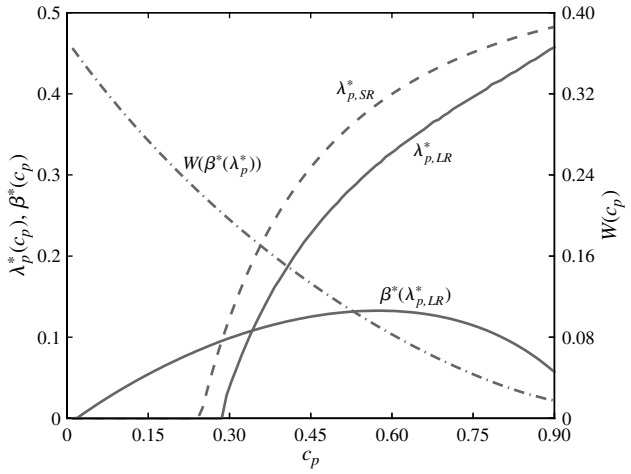
Table 1 summarizes the optimal patch liability policy given in Proposition 4.[11] As can also be seen in the table, the optimal patch liability policy varies significantly across the relevant parameter regions, and patch liability can be effective in very different forms and natures, in distinct regions. First, under low patching costs, when the initial rate of increase in the marginal cost of investment is small (i.e., $C''(0) \le w_1$), patch liability can be an effective tool to increase welfare, and, in fact, it can be optimal to impose full liability for patching (or *maintenance*) costs on the vendor, i.e., $\lambda_p^* = 1$. This is because, when patching costs are low, the vendor can recoup his investment in security from consumers relatively easier by adjusting price. Thus, if the marginal cost of improving software security is not very high, imposing patch liability on the vendor induces him to increase investment in software security compared to the case with no patch liability. As a result, welfare improves and dictates full patch liability on the vendor. On the other hand, when patching costs become too high (i.e., $c_p > 1/3$), the burden of patch liability on the vendor also becomes too high; it becomes harder to pass investment costs onto consumers for whom the total cost of software ownership is already substantial. Consequently, imposing liability reduces the vendor's upside for the software, which leads to depressed investments. In this case, it is best not to impose any patching cost liability on the vendor, i.e., $\lambda_p^* = 0$.

When the increase in the marginal cost of improving software security is moderate (i.e., $w_1 < C''(0) \le w_2$), patch liability becomes ineffective at improving software security and welfare. Under this cost structure, it is too difficult to induce increased investment in security by imposing patch liability without, as a consequence, also forcing the vendor to reflect this additional investment in the consumer price, thus reducing usage. In consideration of these effects, the optimal policy for a social planner is to place no patch liability on the vendor.

However, when marginal cost of software security improvement increases steeply (i.e., when $C''(0) > w_2$), the welfare-maximizing patch liability policy changes significantly in nature and effect. Specifically, the social planner now has to carefully consider the significant security improvement costs the vendor must undertake and how these costs are reflected in the consumer price, usage, and value generated by the software. Part (iii) of Proposition 4 states that the impact of patch liability in this region is quite different in nature than for where the rate of marginal cost increase is low. When patching costs are sufficiently high, imposing a patch liability can result in a *decrease* in security investments compared to the case with $\lambda_p = 0$, yet it can still improve social welfare. The underlying reason is that, from the vendor's perspective, software security investments and contributions to consumers' patching costs act as substitutes. When patching costs are high (i.e., $c_p > 6-\sqrt{33}$), the optimal patch liability policy imposes partial liability on the vendor. His covering of part of users' patching costs boosts their patching behavior and improves network security. This means that the vendor can reduce his investment in improving software security and still get a product that is more secure overall. In effect, the vendor shifts some of his investments from software improvement to sharing patching costs. The end result is reduced base security of the software but improved patch behavior and an increase in generated social value through increased usage. However, this trade-off is not effective in improving welfare when patching costs are low. In that case, i.e., when $c_p \le 6-\sqrt{33}$, consumers are easily induced to patch, and the vendor sets a high price of the software, selling to a higher valuation but smaller consumer group. When patch liability is imposed in this parameter region, because there is a high level of patching and the software is appealing to higher valuation customers, instead of reducing investment in security, the vendor can raise his price to recoup his patch liability expenses. This behavior results in reduced usage and decreased social welfare while also giving rise to a level of security investment that is excessive in terms of costs from the welfare point of view. As a result, imposing patch liability for this case reduces welfare, hence having no patch liability is optimal, i.e., $\lambda_p^* = 0$.

[11] The costs to improve security would be mostly driven by the current software labor market trends. The first derivative of the cost function, C', is driven by the current programmer wages in the industry. The second derivative, C'', represents how costly it is to hire new employees and outside resources to utilize in software development. For example, at a time of abundance of software developers and engineers (e.g., at a time of recession and/or low technology business activity), C'' has lower values. On the other hand, at a time of high software development activity (e.g., at a time of technology boom), qualified labor is scarce and expanding the work capacity by hiring outside software development labor and resources is expensive; this landscape corresponds to a higher C''.

Figure 3 **Optimal Vendor's Share of Patching Costs for Long-Run ($\lambda^*_{p,LR}$) and Short-Run ($\lambda^*_{p,SR}$) Settings, and Optimal Vendor Investment ($\beta^*(\lambda^*_p)$) and Welfare ($W(\beta^*(\lambda^*_p))$) in the Long-Run Setting**



Note. The parameter values are $\alpha = 100$, $\pi_a = 0.20$, $\pi_z = 0.0001$, and the security investment cost function is $\Gamma(\beta)$ with $\delta = 0.30$, $k = 0.01$, and $\hat{\beta} = 0.80$.

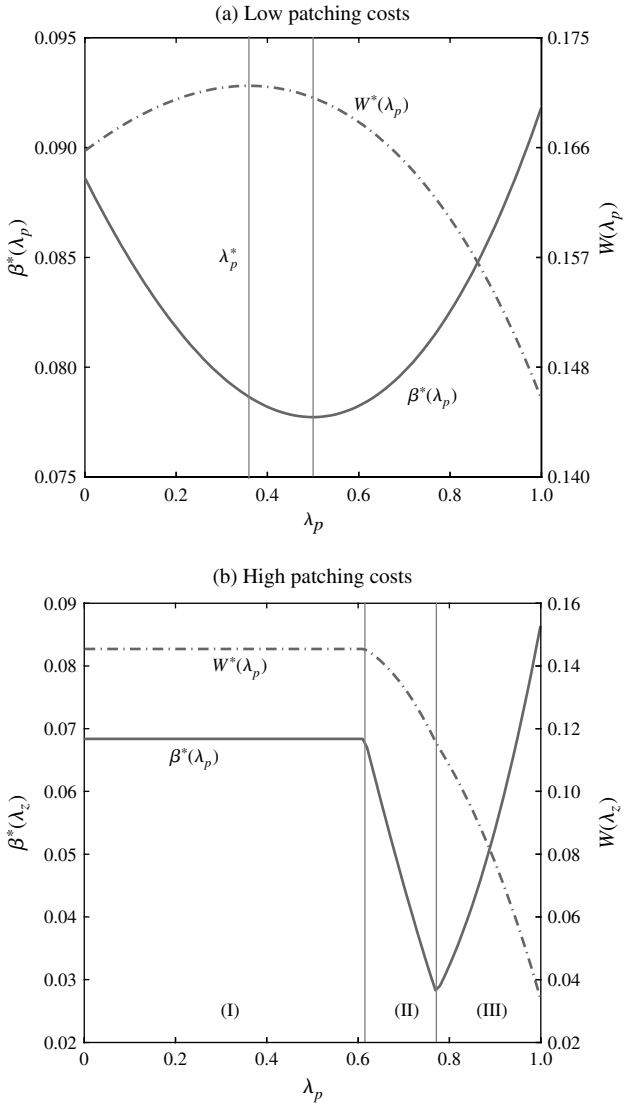
Figure 3 demonstrates the optimal patch policy and the vendor's corresponding investment as a function of patching costs for the low zero-day risk case when marginal security investment costs increase steeply. As demonstrated in Proposition 4, in this setting, when patching costs are low, the patch liability policy is ineffective, i.e., $\lambda^*_p = 0$. Beyond a certain threshold level of patching costs, however, patch liability becomes effective, and the vendor's share of patching costs in the optimal patch liability policy increases with patching costs but stays less than 50%. It is noteworthy that the optimal liability policy for the short-run case imposes a higher burden on the vendor than for the long-run case. This occurs because, in the long run, increased vendor liability has a negative impact on the vendor's security investment, which can, in turn, hurt welfare. As can be seen in the figure, the vendor's optimal security investment is nonmonotonic in patching costs however. The reasoning is that there is an opposite effect of increased patching costs: Higher patching costs reduce consumer patching incentives; thus, to maintain the security of software in a network environment, the vendor increases his security investments. Still, beyond a certain level, the effect of increased vendor patch liability dominates, and the vendor substitutes his software investment with his share of users' patching costs. On the other hand, welfare is monotonically decreasing with increased patching costs.

Next, we explore the efficacy of patch liability when the likelihood of zero-day attacks is high and the vendor has an opportunity to invest to improve security in response to this liability policy. As the following proposition demonstrates, patch liability takes on a similar role as in the short-run setting examined in §4.2.

Proposition 5. *In a high zero-day risk environment, similar to the short-run case, patch liability is effective under low patching costs satisfying* $c_p < \pi_a/(\pi_a + \pi_z)$, *and the vendor's optimal liability share, i.e.,* λ^*_p, *tends toward* $1/2$ *for sufficiently large* α. *Furthermore, the vendor's optimal security investment* $\beta^*(\lambda_p)$ *is initially weakly decreasing and then strictly increasing in patch liability* (λ_p).

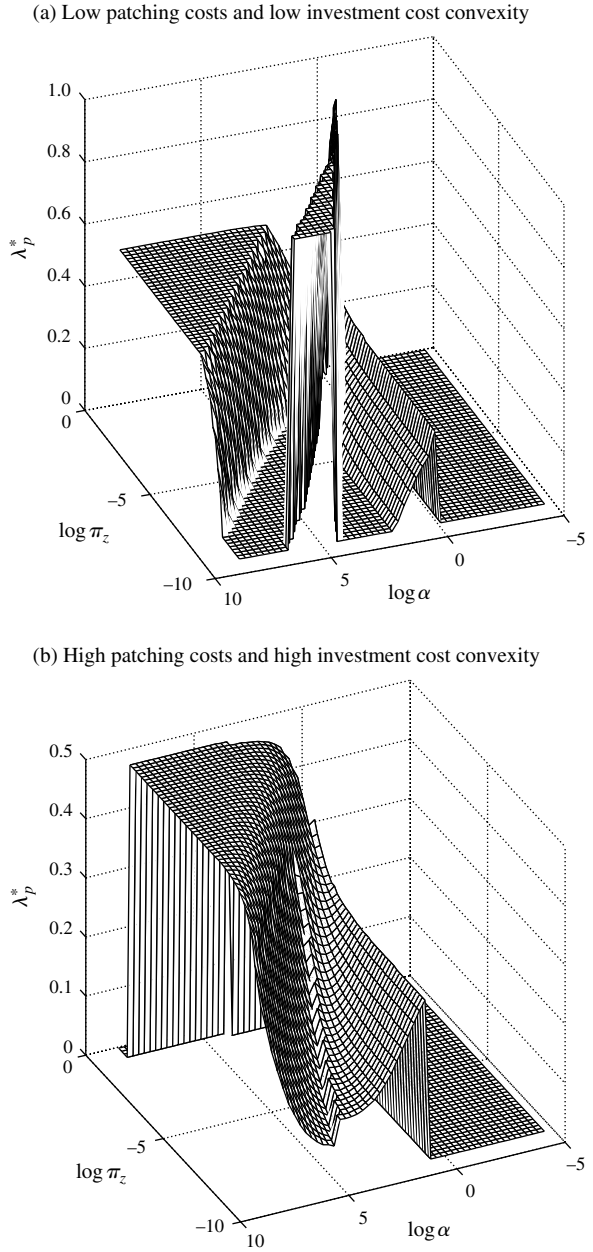
In contrast to the case where zero-day risk is low, the general characteristics of the optimal liability scheme in this long-run setting are similar to those found in a short-run analysis. Essentially, as we previously presented in Proposition 2, the optimal liability share of the vendor tends toward one-half for smaller patching costs, and no liability is optimal for high patching costs. Figure 4 illustrates the effect of patch liability on investment and welfare. As stated in Propositions 2 and 5, when c_p is low, for low λ_p values, increasing λ_p increases consumer patching, thus making the network safer. Because his share of patching costs is also low, the vendor does not have very strong incentives to invest in improving security; in equilibrium, he ends up recovering his losses from patch liability through reduced investments. That is, introducing patch liability can, in fact, make the vendor reduce his total investment in software security. However, in this region, improved consumer patching behavior more than compensates for the increased riskiness of the software product, thus increasing vendor liability actually increases social welfare. On the other hand, when λ_p becomes larger, the patching cost burden on the vendor increases, and, in addition to reflecting these costs on the consumer price, he has greater incentives to increase his security investment. Consequently, for large λ_p values, further increasing λ_p increases $\beta^*(\lambda_p)$ but decreases social welfare, as can be seen in panel (a) of Figure 4. For high c_p and low λ_p values, consumers lack incentives to patch. Therefore, up to a certain point, increasing λ_p affects neither the vendor investment nor welfare because it is ineffective at altering consumer patching behavior, as seen in segment (I) in panel (b) of Figure 4. Beyond a certain threshold level of liability, i.e., in segment (II), some consumers start patching, and the vendor can again reduce his investment in security of the software; the added security stems from consumer patching, which substitutes for the investment. However, when the vendor's share of liability further increases, to curb his losses from expected patching cost compensation to consumers, he increases his investment in security. Even so, welfare suffers from this inefficient increase in investment, as can be observed in

Figure 4 **Vendor Security Investment (β^*) and Welfare (W) as a Function of Vendor Patching Cost Share Under Patch Liability Policy**



Notes. Panel (a) illustrates a low patching cost case ($c_p = 0.24$) and panel (b) illustrates a high patching cost case ($c_p = 0.65$). Remaining parameter values for both panels are $\alpha = 3$, $\pi_a = 0.30$, $\pi_z = 0.19$, and $C(\beta) = \Gamma(\beta)$ with $k = 0$, $\delta = 0.60$, and $\hat{\beta} = 0.80$.

Figure 5 **Optimal λ_p as a Function of α and π_z Under the Security Investment Cost Function $\Gamma(\beta)$**



Note. The parameter values are $\pi_a = 0.20$, $c_p = 0.15$, $k = 0.064$, $\delta = 0.011$, and $\hat{\beta} = 0.60$ for panel (a), and $\pi_a = 0.20$, $c_p = 0.29$, $k = 0.103$, $\delta = 0.25$, and $\hat{\beta} = 0.60$ for panel (b).

segment (III) in the figure and stated in Proposition 5. As a consequence, for cases with high zero-day risk and high patching costs, it is best not to impose any patch liability on the vendor.

Figure 5 summarizes the main optimal patch liability policy findings of this section. Panel (a) illustrates the case for low patching costs (c_p) and slowly increasing marginal costs of investing in software security ($C''(0)$). For low potential security losses (i.e., low α values), it is best not to impose any patch liability on the vendor (i.e., $\lambda_p^* = 0$) because his paying for patching would encourage socially inefficient patching behavior by consumers in a relatively safe network environment. Furthermore, such behavior would lead to reduced usage by inducing the vendor to increase price. As α becomes large, vendor patch liability becomes an effective policy to increase social welfare with λ_p^* exhibiting different characteristics in different regions. For large α and low zero-day attack risk (π_z), full vendor liability is optimal, i.e., $\lambda_p^* = 1$, as we demonstrated in Proposition 4. When π_z is large, however, a partial vendor patch liability policy is optimal, which, in the limit, imposes 50% of patching costs

on the vendor, as can also be seen in the figure. On the other hand, when patching costs are high and the vendor's marginal costs of investing in security increase steeply, the optimal vendor patch liability policy has a different nature, as illustrated in panel (b) of Figure 5. In this case, if the zero-day attack risk is low, full vendor liability is not effective because it will not significantly boost his security investment, while partial liability is optimal. As zero-day risk becomes high, the optimal vendor patch liability again approaches one-half of the costs. However, when π_z is higher than a certain threshold level, the profitability of the software suffers, and, combined with high patching costs, imposing liability on the vendor becomes counterproductive; it serves to induce the vendor to heavily increase software price. As a consequence, placing no patch liability on the vendor becomes optimal (i.e., $\lambda_p^* = 0$), as formalized in Propositions 2 and 5 and seen in panel (b) of Figure 5 (see, e.g., $\log \pi_z > -1/2$).

5.4. Software Security Standards

So far, we have examined zero-day loss and patching cost liability schemes on the vendor to improve software security and the value generated by software. Another way to hold the vendor responsible for software security in a long-run setting is to ensure he spends enough effort and resources to achieve a certain security level prior to its release, i.e., setting and enforcing security standards. Similar quality and safety standards are set and enforced in many other products from automobiles, pharmaceuticals, and medical equipment to food products (U.S. Department of Transportation 2004, Joyce 2005, Weise 2009), and there have also been suggestions to enforce such standards for software security (Fisher 2002, Clarke 2003, Payne 2004, Feinman 2009). In this section, we examine the impact and performance of a software security standards policy as a form of cyber-security regulation.

Under security standards, a social planner imposes that the software vendor comply with a set of guidelines toward assuring his software is sufficiently secure. Essentially, such a requirement is equivalent to having a social planner set the risk reduction factor β_s. In this case, the vendor's profit-maximization problem becomes

$$\begin{aligned} &\max_{p\in[0,1]} \ \Pi(p, \beta_s) \\ &\text{s.t.} \ \ v_b \text{ satisfies } \sigma^*(\cdot \mid p, \beta_s), \end{aligned} \tag{19}$$

and the planner's problem can be written as

$$\begin{aligned} &\max_{\beta_s\in[0,1]} \ W(0, \beta_s) \\ &\text{s.t.} \ \ (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p^*(\beta_s), \beta_s) \\ &\qquad p^*(\beta_s) \text{ satisfies (19)}. \end{aligned} \tag{20}$$

The following proposition presents the solution to this problem.

Proposition 6. *Denote the unique solution of the equation*

$$3c_p(1 - c_p(1-\xi)) - 4C'(\xi) = 0 \tag{21}$$

in $(0, 1)$ *by* ξ^*. *In a security environment characterized by*

(i) *low zero-day security risk, the welfare-maximizing investment level approaches* ξ^* *for sufficiently large* α. *Furthermore,* (a) *under low patching costs, the prescribed investment level is increasing in* c_p; *whereas* (b) *under high patching costs, the investment level decreases in* c_p *whenever* $\xi^* < 1/2$;

(ii) *high zero-day security risk, the welfare-maximizing investment level tends toward zero. Furthermore, the prescribed investment level weakly decreases in* c_p.

When zero-day attack likelihood is low, there is a significantly larger purchasing population that can benefit from a more secure product than in a case where these attacks are more likely. Thus, social welfare can be improved by imposing stringent security standards that force the vendor to increase his investment substantially as established in part (i) of Proposition 6. In this case, if patching costs are low, much of the purchasing population is patching to protect themselves from potentially large security losses; thus, there can be significant social gains if consumers' *effective* patching costs are reduced through increases in β_s^*. On the other hand, when patching costs are large, the patching population is small, which, in turn, causes the total purchasing population to shrink due to negative externalities arising from a larger unpatched population. In this case, the vendor naturally already has strong incentives to invest in security, particularly if the marginal cost of increasing security does not increase too steeply. As c_p increases through this range, a planner can ease regulation since she no longer needs to control for the vendor's pricing behavior because his investment incentives become more closely aligned, as stated in part (i)(b) of the proposition.

On the other hand, when zero-day attacks are more common events, the social benefits with security investment regulation become depressed. Because the risk associated with this type of attack cannot be shed by proper patch maintenance, it tends to get managed by significant reductions in usage, which serve to control the network externalities. When usage is generally low, the social impact of security investment becomes marginalized. Thus, an optimal security standards policy calls for lower β_s^*, which decreases as patching cost increases, as seen in part (ii) of Proposition 6.

5.5. Policy Comparisons

Having examined how liability and security standards regulatory policies affect security investment and welfare, we now explore the relative effectiveness of these policies at improving social welfare. This analysis can help determine the most preferable security approaches for a social planner. We compare the performance of the three policies we studied, namely, loss liability, patch liability, and security standards policies; again, we focus on the two main security risk environments examined: high zero-day risk and low zero-day risk under high potential losses from a security breach.

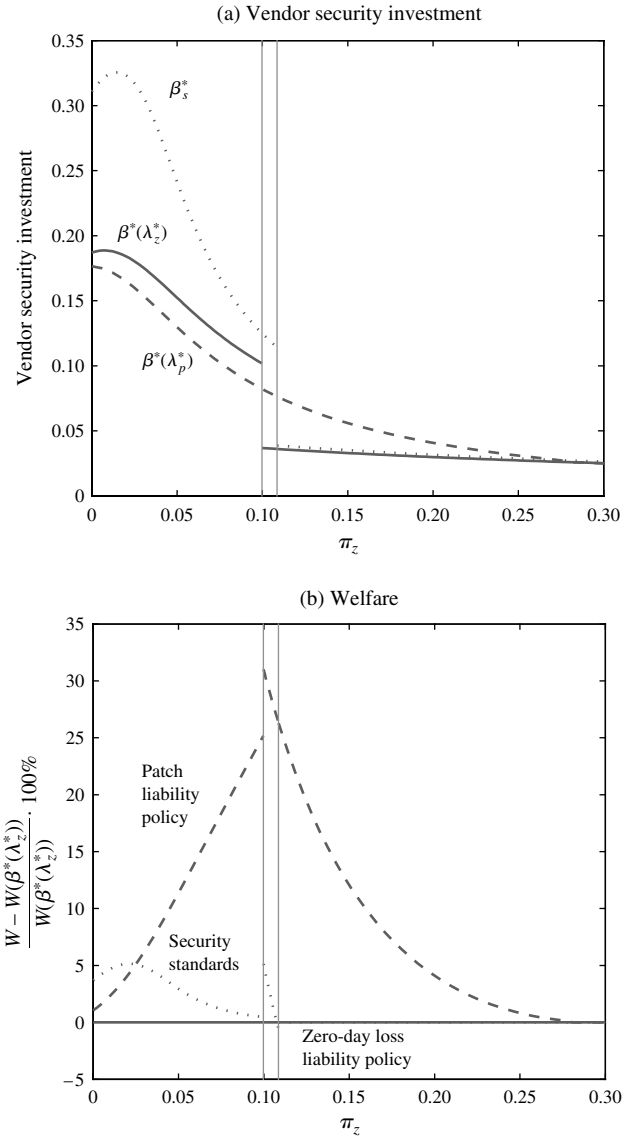
We start with the policy recommendation for low zero-day risk environments.

Proposition 7. *For low zero-day risk environments, a mandated security standards policy* (i) *generates the highest investment compared to zero-day loss and patch liability policies, i.e.,* $\beta_s^* > \max(\beta^*(\lambda_z^*), \beta^*(\lambda_p^*))$; *and* (ii) *achieves the highest welfare, i.e.,* $W(\beta_s^*) > \max(W(\lambda_z^*), W(\lambda_p^*))$.

Figure 6 illustrates the comparisons of security investment levels and welfare under the various policies. As part (i) of Proposition 7 states, for low zero-day risk environments, security investment is highest under the mandatory security standards case. Recalling from Proposition 6, when zero-day attack risk is low, a security standards policy enforces a relatively high level of security investment, β_s^*. On the other hand, as we discussed in §5.2, imposing zero-day loss liability hurts vendor profitability. Consequently, vendor investment in software security improvements declines in response, and, even for cases where loss liability increases the vendor's investment, a strong boost is unlikely. Similarly, as we demonstrated in §5.3, from the vendor's perspective, patch liability tends to serve as a substitute to investment in security. Therefore, instituting a vendor patch liability policy does not spur vendor security investments either and, in fact, can often reduce it. Therefore, under low zero-day risk environments, a mandated security standards policy ensures a significantly higher security level than either loss or patch liability policies, as can also be observed in panel (a) of Figure 6.

In low zero-day risk environments, provided that the product is sufficiently secure, it is possible to achieve a significant user population and, hence, generate a lot of social value from the software. Thus, in such cases, enforcing a significant security investment level through the use of standards can pay off in terms of improved welfare. Liability policies, on the other hand, rely on the vendor to have the appropriate incentives to invest. The zero-day loss liability policy aims to shift the losses resulting from software security failures to the vendor, thereby inducing him to invest to eliminate such failures. However, as we have discussed above and in §5.2, the zero-day loss liability policy is ineffective in creating strong incentives to boost vendor security investments; in fact, it backfires and reduces these investments in many cases. In addition, the extra liability burden placed on the vendor can translate into higher prices and lower usage. The patch liability policy increases software security by encouraging user patching behavior. However, improved user patching acts as a substitute for vendor investment, and, even in cases where patch liability induces increased investment, the magnitude of the investment boost is small. Consequently, in a low zero-day risk environment, neither zero-day loss liability nor patch liability can generate the welfare level obtained from enforcing security standards. Thus, a

Figure 6 Policy Comparison of Security Investment and Welfare



Note. The parameter values for both panels are $\alpha = 20$, $\pi_a = 0.30$, $c_p = 0.50$, and $C(\beta) = \Gamma(\beta)$ with $k = 0$, $\delta = 0.40$, and $\hat{\beta} = 0.50$.

mandated standards policy is the best policy for this environment, as stated in part (ii) of Proposition 7 and seen in panel (b) of Figure 6.

Next, we present our policy comparisons for high zero-day risk environments.

PROPOSITION 8. *When $\pi_z > \pi_a(1-c_p)/c_p$, patch liability is not effective and a mandated security standards policy results in higher welfare than the patch liability policy. Otherwise, for high zero-day risk environments, when $\pi_z < \pi_a(1-c_p)/c_p$, a patch liability policy can induce a higher vendor security investment than set by the optimal mandated security standards policy and is the preferred policy for social welfare. Technically, there exist $\underline{\omega}$, $\underline{\eta} > 0$ such that if $\alpha > \underline{\omega}$, then*

(i) *if $\pi_z > \underline{\eta}$, then $W(\lambda_z^*) \le \min(W(\lambda_p^*), W(\beta_s^*))$;*

(ii) (a) *if $\underline{\eta} < \pi_z < \pi_a(1-c_p)^2/(c_p(2-c_p))$, then $\beta_s^* > \beta^*(\lambda_p^*)$; otherwise, if $\underline{\eta} < \pi_a(1-c_p)^2/(c_p(2-c_p)) < \pi_z < \pi_a(1-c_p)/c_p$, then $\beta_s^* < \beta^*(\lambda_p^*)$;*

(b) *$W(\lambda_p^*) > W(\beta_s^*)$ if and only if $\underline{\eta} < \pi_z < \pi_a(1-c_p)/c_p$.*

As we discussed in §5.2, the zero-day loss liability policy is ineffective in high zero-day risk environments, and, therefore, as also noted in part (i) of Proposition 8, it is dominated by the security standards and patch liability policies. When zero-day attack risk is high, by Proposition 6, the optimal investment level under a security standards policy is very low. In fact, as part (ii)(a) of Proposition 8 states and can be seen in panel (a) of Figure 6, the optimal security investment level under the standards policy is higher than the investment induced by the patch liability policy only if π_z is under a certain threshold. Under high zero-day risk, the limited security investment under the standards policy suggests that this policy can only make a modest improvement.

The other alternative to improve network security, namely, incentivizing users to patch their software by forcing the vendor to cover part of their patching costs, on the other hand, works effectively, as can be observed in panel (b) of Figure 6. In §5.3, we established that when zero-day risk is high, a patch liability policy reduces vendor investments because the vendor substitutes security investments with patching costs paid to the users. The overall influence of the patch liability policy turns out to be efficient because it increases network security while reducing ineffective security investments. As a result, for high zero-day risk environments, as part (ii)(b) of Proposition 8 states, the patch liability policy is recommended. However, this conclusion is valid only for cases where a patch liability policy can induce users to patch their software. Note that $\pi_z \le \pi_a(1-c_p)/c_p$ if and only if $c_p \le \pi_a/(\pi_z+\pi_a)$. As we have shown in Proposition 2, when $c_p > \pi_a/(\pi_z+\pi_a)$, patch liability policy is ineffective because no policy in this class can induce users to patch. Therefore, when $\pi_z > \pi_a(1-c_p)/c_p$, a standards policy becomes the recommended policy for social-welfare maximization. When π_a is sufficiently large and/or c_p is sufficiently small, however, $\pi_a(1-c_p)/c_p > 1$ holds; this means that when patchable attack risk is high or patching costs are low, the patch liability policy is the recommended policy for all sufficiently high π_z levels, even as π_z approaches 1.

5.6. Joint Policies

Propositions 7 and 8 establish that the most effective policy in low zero-day risk environments is the employment of security standards; the most effective policy in high zero-day risk environments is patch liability (when it is applicable), and zero-day loss liability tends to be the least effective. An important question is whether there exist substantial benefits from combining the effects of these separate policies to further positively affect software security and social welfare. In this section, we investigate the joint welfare maximization problem to gain insight on this matter.

We denote the liability shares and security investment level employed under a standards policy with $\mathbf{\Lambda} \triangleq (\lambda_z^c, \lambda_p^c, \beta_s^c)$. The vendor's profit function can then be represented as

$$\Pi(p, \mathbf{\Lambda}) \triangleq p(1-v_b) - \lambda_z^c L_z - \lambda_p^c L_p - C(\beta_s^c). \tag{22}$$

In this setting, for any fixed $\mathbf{\Lambda}$, the vendor's problem is given by

$$\begin{aligned} \max_{p\in[0,1]} \ & \Pi(p, \mathbf{\Lambda}) \\ \text{s.t.} \ & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p, \mathbf{\Lambda}). \end{aligned} \tag{23}$$

The welfare function is still given by the right-hand side of (11), where β_s^c is the investment level and the equilibrium consumer market threshold valuations are now computed under $\mathbf{\Lambda}$. Thus, the planner's problem is given by

$$\begin{aligned} \max_{\mathbf{\Lambda}\in[0,1]^3} \ & W(\mathbf{\Lambda}) \\ \text{s.t.} \ & (v_b, v_p) \text{ satisfy } \sigma^*(\cdot \mid p^*(\mathbf{\Lambda}), \mathbf{\Lambda}) \\ & p^*(\mathbf{\Lambda}) \text{ solves (23)}. \end{aligned} \tag{24}$$

In §5.5, we compared the independent effects of the three policies being studied and found that security standards and patch liability were most effective. Whether a joint policy can engender any welfare gains hinges on being in a security landscape where the underlying, independent effects of each policy can improve welfare.

Proposition 9. (i) *For a low zero-day risk environment*,

(a) *the welfare-maximizing joint policy can yield only a marginal improvement over the optimal single policy, security standards, if patching costs are large and satisfy* $c_p > 6 - \sqrt{33}$, *and investment costs increase steeply, i.e.,* $C'(1 - (6 - \sqrt{33})/c_p) > 3c_p(\sqrt{33} - 5)/4$;

(b) *otherwise, the joint policy and security standards are equivalent, i.e.,* $W(\mathbf{\Lambda}^*) = W(\beta_s^*)$;

(ii) *For a high zero-day risk environment, the optimal joint policy yields either no improvement over the optimal single policy or the difference quickly converges to zero as* α *increases*.

First, we examine low zero-day risk environments. Proposition 7 establishes that security standards are the most effective. Furthermore, as indicated by Proposition 4, patch liability tends to be effective under high patching costs and investment cost functions that have higher convexity, whereas, by Proposition 3, loss liability tends to be ineffective under such cost functions. Thus, under these conditions and as stated in part (i)(a) of Proposition 9, there is some potential for a joint approach to enhance welfare. However, we formally establish that this potential converges to zero as α grows large. Under low patching costs, patch liability can induce inefficient patching. Similarly, when the security investment cost function is relatively flat, the use of security standards tends to offset the positive effects from patch liability. In this case, security standards alone can maximize welfare as stated in part (i)(b) of Proposition 9.

In part (ii) of Proposition 9, we examine high zero-day risk environments under a joint policy approach to software security. Similar to part (i), whether one employs security standards or patch liability as prescribed by Proposition 8, the use of the single, optimal lever seems to be sufficient. For high zero-day risk environments, we established in Proposition 3 that loss liability is ineffective. Furthermore, by part (ii) of Proposition 6, the benefits stemming from the use of security standards are small. However, in high zero-day risk environments, patch liability can be quite effective, as established by Proposition 5. In this case, a joint policy would find it difficult to improve upon patch liability alone. Panel (b) of Figure 6 also illustrates this result in the right-hand portion of the plot, where π_z is high and only patch liability has a nonnegligible effect. Part (ii)(b) of Proposition 8 also establishes that when both zero-day risk and patching costs are high, patch liability may not be preferable. In this case, the joint policy would again be identical to a security standards policy.

Our findings suggest that it may not be recommended to take a joint policy approach toward security in order to increase the social surplus associated with software. Clearly, in general, a joint policy approach can always weakly improve social welfare. However, to justify the employment of a joint policy, the benefits must minimally offset the additional costs of implementation.[12] Because the purpose in our work is to focus on identifying the strengths of individual policies and characterizing regions where each policy is most effective, we have abstracted from the details of implementation. Nonetheless, Proposition 9 establishes that the benefits from a joint approach are marginal and, therefore, may not justify these implementation costs, which are likely to be nonnegligible.

6. Policy Summary

In §4, we studied the efficacy of patch liability and zero-day loss liability on improving welfare in short-run settings. Next, in §5, we explored how these policies perform when software vendors can respond by increasing or decreasing their investments in security. In the latter context, we also examined whether software security standards can improve welfare. Compiling all of these results together, we next present a summary of the policy implications of our analysis while highlighting how different recommendations apply to different security landscapes and investment settings. Furthermore, we also provide more insight into how to connect these policy implications to realistic classes of software and the potential implementation challenges that may arise.

In this paper, we focus on a region where α is sufficiently high that some users are patching their software installations when vulnerabilities arise; this setting is the most realistic one, as evidenced by actual users' patching behavior in the face of ongoing security attacks. Under reasonably high α, we can then divide our results into a 2×2 matrix of low/high patching costs and low/high zero-day risk. For the first categorization, client-side software applications tend to have relatively lower patching costs than enterprise software applications because the latter typically require significantly more effort.[13] For the second categorization, we leveraged publicly available vulnerability and attack information for various classes of software as well as specific packages to determine what types of software are exposed to higher *absolute* zero-day risk.[14] Using these available data and statistics,

[12] Implementation costs refer to the enforcement of software liability laws and security standards regulations.

[13] The testing and deployment of updates to enterprise software applications often involve a structured flow through local, staging, and production server environments for validating installations and testing application interactions rigorously before final deployment (see, e.g., Beres and Griffin 2009 for details on typical enterprise patching processes).

[14] Our policy implications are based on absolute zero-day risk rather than relative risk, hence the risk is influenced by the conditional probabilities that compose it, including the probability of simply a vulnerability arising.

Table 2 Examples of Software Products for Various Security Landscapes

Software applications	Client software	Enterprise software
	Low patching cost	High patching cost
Low zero-day risk	WinEdt, Symantec Anti-Virus, Microsoft Windows Media Player, Apple OSX	IBM Lotus Notes, Oracle E-Business Suite, SAP MaxDB, Novell Groupwise
High zero-day risk	Microsoft Internet Explorer, Microsoft Office, Sun JRE, Adobe Acrobat, and Reader	Microsoft IIS, Microsoft SQL Server, Oracle Secure Backup, Microsoft Windows Server

we have composed a list in Table 2 of the typical software packages that would fall into each of the four relevant categories presented in our policy summary (IBM 2008, Hewlett-Packard 2010).[15]

In Table 3, we summarize our policy recommendations for each of these different classes of software. As can be seen in the table, a main takeaway from our analysis is that it is advisable to social planners to refrain from imposing *loss liability* on vendors of software with interdependent risks. We find that whether or not software vendors have the opportunity to invest in security in response to loss liability makes little difference in the net impact of such a policy: welfare mostly decreases. On the surface, one may think that holding the vendor liable for a portion of consumers' losses would provide increased incentives for the vendor to improve security, and, in some cases, the vendor's optimal investment indeed rises. However, even if investments positively respond, the vendor tends to restrict usage through pricing to limit its exposure to liability on these losses. Hence, loss liability is an inefficient policy to apply in the software industry. On the other hand, we find that both *patch liability* and *security standards* should play an important role in software policy although, in certain cases, no liability is necessary.

Panel (a) of Table 3 presents our policy recommendations for short-run settings where investments are fixed. First, we consider software products that exhibit low patching costs and high zero-day risk (see Table 2). According to IBM X-Force, in recent years, the client software applications facing the highest zero-day risk include Web browsers, dynamic Web content enabling engines, and document readers (Cox et al. 2006, IBM 2008). Typical examples include products such as Microsoft Internet Explorer and Office. In this setting, we find that the application of patch liability

[15] We use data available at http://www.zerodayinitiative.com/advisories/published/ in part to categorize the software applications.

Table 3 Recommended Policy for Various Security Landscapes and Investment Scenarios

	(a) Short run		(b) Long run	
Optimal policy	Low patching cost	High patching cost	Low patching cost	High patching cost
Low zero-day risk	No liability	Patch liability	Standards	Standards
High zero-day risk	Patch liability	No liability	Patch liability	Standards

could benefit social welfare. That is, it can be helpful from the social-welfare standpoint to hold software vendors liable for a portion of the users' patching costs. On the other hand, when both patching costs and zero-day risk are low, we find that liability may be unnecessary because users and software vendors are likely to make decisions that are fairly welfare efficient. Examples of client-side software with low zero-day risk include text editors, antivirus tools, some media players, and lower profile operating system software. Listed in Table 2, we provide some examples of these software packages, including WinEdt, Symantec Anti-Virus, and Apple OSX.

Similarly, when both patching costs are high, as it is with enterprise software, and zero-day risk is substantial, then liability in a short-run setting is unlikely to be helpful. Data suggest that the more risky enterprise software packages are geared for the Web, including Web server software and some database management systems. Notable examples of software in this category include Microsoft IIS, SQL Server, and Oracle Secure Backup (Moore et al. 2002, Halfond et al. 2006, Hewlett-Packard 2010). Finally, when enterprise software faces low zero-day security risk, patch liability on software vendors is once again optimal. Products with less zero-day risk include many enterprise application software packages including IBM Lotus Notes, Oracle E-Business Suite, and Novell Groupwise. As is summarized in panel (a) of Table 3, in some cases, no additional software security responsibility should be imposed on the software vendor, and the users should be held responsible for their own security patching costs. However, in other cases, the vendor certainly needs to become more responsible, and this responsibility should take the form of patch liability.

Because patching costs exist and furthermore because they are sufficiently large that a nontrivial portion of software users do not patch and incur losses when security attacks occur, it is important to understand the impact of policies that can help induce users to perform proper patch maintenance. The policy we study is for the vendor to assume a share of these costs. By holding the vendor responsible for part of the patching costs, a larger consumer population can be induced to patch, and the risk level of the network can decrease, particularly in the face of high zero-day risk.

There are many ways to implement the essence of such a policy. One way to implement it is to simply have software vendors offer customers who agree to adhere to an appropriate patching strategy with a price discount. In the context of our model, this mechanism is equivalent to having the vendor be liable for a portion of the patching costs. Furthermore, patching behavior is verifiable because, in an interconnected environment with risk, the vendor can use the interconnectedness for communicating patching status (see, e.g., August and Tunca 2006 for more on this point). For example, let us consider Microsoft Office. One effective implementation would be to have Microsoft offer a discounted version of Office with mandatory updates. Also, schemes such as rebates or other means that reduce well-behaved users' costs can all work to some extent. The primary challenge of implementing patch liability are privacy concerns. There may be a portion of the consumer base that prefers not to permit the software vendor with a means of verifying patching status because it requires the passing of information *out* of firewall protected servers. Implementation of this type of change to the security policies of a user firm may be against its current practices and, therefore, encounter organizational resistance. However, the net impact of patch liability on welfare is substantial, which can help drive change.

Turning attention to long-run settings where the vendor can adapt its investment in security, under high zero-day risk and low patching costs, we find that patch liability continues to be the most effective policy to improve welfare. Our results for this setting are summarized in panel (b) of Table 3. However, under low zero-day risk/low patching cost and high zero-day risk/high patching cost, social welfare can now benefit from the application of liability on the software vendor. For these two corners of the table, in contrast to the recommended absence of liability in the short run, we find that it is best to directly target the software vendor's investment through the use of software security standards. In the case of low zero-day risk and high patching costs, the recommended policy shifts from patch liability to security standards as well. The main implication here is that under the sets of conditions that span these three categories, there are insufficient incentives for software vendors to produce software that is sufficiently secure. Because this software runs in networked environments and its users generate negative externalities on one another, the government may need to ensure that producers are developing software that meets some predetermined minimum standards. Thus, in such cases, the entity who most needs to be responsible for software security is the software vendor, and the policy recommendation is direct regulation of security investment through standards.

Implementation of standards is a common function of the government. For instance, the U.S. Department of Labor's Occupational Safety and Health Administration (OSHA) agency enforces regulations to maintain workplace safety and health, and the U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) agency writes and enforces standards for motor vehicles (Bennett 2010, Berzon 2010). For the case of software causing security risk across networks, an agency such as the U.S. Department of Homeland Security (DHS) would need to work with security experts to generate guidelines for software producers to adhere to in developing software. For example, checking compliance to these guidelines would improve the social value associated with Oracle's E-Business Suite in the long run. Such guidelines could ensure that SQL injection, session hijacking, password decryption, and remote attack vulnerabilities are tested for and prevented during development. The primary challenge of implementation is that checking compliance and enforcing the standards can be quite costly and will require the support of both government and taxpayers. However, as software continues to become increasingly pervasive in all industries and drives critical functions (e.g., software now controls automobiles, planes, and even the smart grid), people's lives are becoming more at risk. Similar to OSHA and NHTSA, the enforcement of software security standards may become a necessity in the near future.

7. Discussion and Concluding Remarks

In this paper, we explored the effectiveness of software security liability policies that have been the subject of an ongoing debate in the software security community in recent years. We explored three separate liability mechanisms, namely, zero-day loss liability, patch liability, and security standards. We derived the optimal liability sharing under each mechanism and compared the effectiveness of each mechanism under short-term and long-term investment scenarios. Our analysis allowed us to derive policy implications for software security liability for different classes of software and market environments. There are a number of future research directions our current model and analysis can be extended to, which we discuss in this section.

Patching costs may increase with factors such as the relative size of a user firm, for instance, depending on the number of servers on which they run the software product. The main reason is that a larger number of users require a higher number of servers likely in different environments and configurations on which

to patch the software, driving up the patching costs proportional to the number of servers since the patching cost per server is approximately constant (see, e.g., Bloor 2003, Forbath et al. 2005, Beres and Griffin 2009). Our results continue to apply with consideration of this dimension: In our model, each host can be considered to be a single server and a large corporation can have a number of different servers from which they may potentially derive different valuations due to the different number of users that each server is intended to support. Mathematically, in our setting, a company's purchasing and patching decisions regarding each server can be treated independently as long as a company does not represent more than countably many points in the consumer valuation space.

Modeling the cost of patching as a constant is standard in the literature that examines topics in security related to patching costs, as can be seen in Beattie et al. (2002), August and Tunca (2006), Cavusoglu et al. (2008), and Choi et al. (2010). However, all of our results and implications are robust to the introduction of heterogeneous patching costs and will continue to hold over broad regions. For example, under high zero-day risk, Propositions 2 and 5 establish results on broad regions of patching costs; thus, even as heterogeneity is introduced, these results and their corresponding insights will continue to apply. One interesting extension is to explicitly characterize how, under low zero-day security risk, the optimal patch liability policy adjusts to heterogeneous patching costs. Particularly, because when the rate of increase in the marginal cost of security investment is either sufficiently low or high, the prescribed optimal liability shares are sensitive to patching costs, as can be seen in the middle row of Table 1. In this case, developing an understanding of how heterogeneous patching costs can tip the liability shares in either direction can yield interesting insights as an extension. However, in this paper, the main point and the essence of the patch policy is to simply understand the impact of having the vendor incur some costs to incentivize unpatched users to begin performing better patch maintenance. Our model gives sharp implications on when and how welfare can be enhanced in that regard.

There are several other important directions for future research related to this work on software liability and security standards. The first extension is to bridge the gap between research on vulnerability disclosure and software liability in a context where users can patch systems. Specifically, in the vulnerability disclosure literature, researchers study the socially optimal timing of disclosure as well as the release timing of patches. In our model, the focus of security investment is to reduce the likelihood a security vulnerability arises in the first place, i.e., decrease π_v. However, there are many factors that can influence whether it is more likely that a white hat or black hat is first to find a vulnerability, or perhaps that the vulnerability is never found. For example, choices studied in the vulnerability disclosure literature such as whether to disclose a vulnerability before a patch is available can have a significant effect on π_w and π_b in our model. Similarly, vulnerability markets where discovery information can be sold by individuals for profit can also have a large influence on these discovery probabilities. By supplementing models where users make patching decisions with these concerns related to vulnerability disclosure, we can start to develop an even richer understanding of these systems. In this context, we can also study what role liability and security standards policies would assume in an integrated approach toward software security.

Another future extension is studying insurance for damages from security breaches or vendor warranties as potential tools for improving value generated by vulnerable software. For instance, insurance can be offered to buyers of software by a third party entity or the government to cover part or all of the losses that are incurred in case of a zero-day attack. Users who choose to purchase this coverage would pay a premium on top of the purchase price. Such a mechanism can improve usage and social welfare, especially when users are risk averse. Alternatively, the vendor can also offer insurance coverage or it can offer warranties to cover a certain percentage of damages or patching costs. In fact, note that the zero-day loss liability and patch liability schemes we study in this paper correspond to partial (or full for $\lambda_z = 1$ or $\lambda_p = 1$) warranty coverage, where the warranty is bundled with the product; hence, its price is incorporated into the product price.

Our results in this paper are robust to the setting where the software vendor's investment in security is unobservable to users. In our model, both the investment and the security properties of the software are common knowledge. The reason is because, in many cases, consumers rationally form expectations that quality will be provided at such a level. This point relates to what is actually observed in reality; it is generally not the case that a vendor has incentives to significantly scale back security investment in order to exploit an advantage stemming from unobservability. Instead, a vendor who exhibits myopic behavior would be severely penalized going forward by consumers since, in reality, they have repeated interactions. If we extended our model to include unobservability in a repeated game context that captures these repeated interactions, we would essentially establish the same results; this fact lies at the heart of the well-known results involving repeated games and the Folk Theorem (see, e.g., Rubinstein 1979, Fudenberg

and Maskin 1986, Tirole 1988). Because the essence of our results and corresponding intuitions would be unchanged and few additional insights would emerge, we employ the simpler, static model with full observability to develop the understanding of the role for software liability and security standards that we provide in this work.

A related future extension of the model can include multiple software vendors who price their products competitively after investing in their own softwares' security (see, e.g., Spence 1977).[16] In such a model, the product price is forced to a level that leaves each firm with zero expected profits, taking their reliability investments into account. As a consequence, the consumers can perfectly refer to the firms' investments. That is, such a setup allows for signaling of unobserved reliability investment, but it takes away strategic vendor decisions on price and its interaction with strategic investment decisions in reliability/software security. Such decisions and their interactions do exist in real-world software markets, which are normally imperfectly competitive. However, an extension of the model with perfectly competitive vendors is an interesting future research direction that can shed light on the competitive implications of liability.

Our model focuses on undirected security attacks that exploit vulnerabilities over a network of interconnected users. This class of attacks typically spread or hit randomly through unpatched hosts. In such environments, for an unpatched user, the probability of being hit is increasing in the size of the unpatched population. This is caused by several factors such as the increased number of unpatched users causing easier spread of malicious code and the increased size of unpatched users (and buyers for zero-day attacks) also increasing the incentives of hackers to search and attack vulnerabilities in a given software product. In our model setup, the probabilities for white hat and black hat vulnerability discovery and the attack probabilities (π_w, π_b, π_{ba}, and π_{wa}) are assumed independent of the unpatched user population. However, this setup is essentially mathematically equivalent to the one that assumes that these probabilities are increasing in the unpatched user population. With such a modeling assumption, the probability of being hit by an attack for an unpatched user would depend on the size of the unpatched user population, effectively taking the form of a product of the effects of the size of this population on the probabilities of vulnerability discovery, attack upon a discovery, and being hit conditional on there being an attack. Therefore, the resulting effect would be the same as in the current model in that the end probability of being hit would be increasing in the size of the attack, and the general insights from our model would still hold.

[16] We thank an anonymous referee for suggesting this extension.

Some security attacks are more directed in nature and typically focus on compromising a particular high-profile host. For example, Google recently fell victim to a security attack in which the attackers gained access to the authentication software code (Worthen and Vascellaro 2010). The popular CRM software provider, Salesforce.com, also suffered a directed attack in which hackers gained access to a database that held private customer information for some of the businesses that use Salesforce.com's service (Krebs 2007). Similarly, Monster.com's resume database was breached by hackers in an effort to steal private information (Keizer 2007). Examining security environments and the role of policy when both directed and network based attacks are present is another interesting future research direction.

In our setup, the zero-day and patchable attack probabilities (π_z and π_a) do not depend on the vendor's investment in security (β). There are two aspects to this assumption. First, provided that both probabilities are affected by the vendor's investment in security in the same way, the independence of the attack probabilities from the vendor investment is, in terms of its effects on the equilibrium outcome and policy implications, mathematically equivalent to a recalibration of the model with adjustment of the vendor security investment cost function $C(\cdot)$ (within certain limits) to accommodate for these effects. A second issue is whether the vendor's security investments would affect the two probabilities differently. Although there is inherently no data available on this issue for analysis, we believe the effect of the vendor's investment on the two probabilities would not differ significantly. Furthermore, inclusion of such a differential effect would complicate the analysis significantly and hurt tractability. A potential future extension of the model could explore the case where the vendor's security investment affects the zero-day and patchable attack probabilities differently.

When considering the enforcement of security standards, there are multiple ways a regulator can detect vendor compliance. One way of doing so is with direct auditing. Regulators can perform random audits or regularly check before the release of each version of the software whether the required security measures and steps are being taken by the vendor to comply with the standards. Another method for detection is ex post monitoring of the number (and nature) of vulnerabilities that get realized. Based on the expected statistical properties of the number of security vulnerabilities that emerge, a punitive action on the vendor can be triggered if the number of vulnerabilities exceeds a certain threshold. A challenge of employing this method is that, in order for it to be used as a basis for legal enforcement, derivation of reliable vulnerability statistics is required. This may require long

periods of observation in changing software and security environments. Nevertheless, an interesting future extension could study this detection method and its effect on socially desirable software security liability policy.

In our study, we used asymptotic analysis, which is a commonly used method in microeconomic analysis. The use of asymptotic analysis can be expected here (as in many other studies with microeconomic models) because of the complexity of the problem and its solution characterization (see, e.g., Li et al. 1987; Laffont and Tirole 1988, MacLeod and Malcomson 1993; Pesendorfer and Swinkels 2000; Muller 2000; August and Tunca 2006, 2008; Vereshchagina and Hopenhayn 2009; Wan and Beil 2009; among many others). In this method of analysis, typically, the exact lower and upper bounds do not have closed-form expressions. However, the aim of the asymptotic analysis is not the closed-form expression for the bound but the existence of the bound itself and the uniform behavior of decision variables, performance measures, and equilibria beyond such a threshold. The goal is the characterization of the region under which the statement of the proposition holds. For instance, in our case, using asymptotic analysis techniques, we identify the regions under which each policy implication for liability is valid. Our results are robust and satisfied for wide parameter regions as can be observed in conjunction with the statements of the propositions and the figures (e.g., Figures 5 and 6).

In summary, our results in this paper identified liability policy recommendations and established the criticality of zero-day attack likelihood, patching costs, and the network environment in making vital software security regulation decisions. Our findings aim to help provide insights to regulators, policy makers, and the software community through recommendations on the issues of this important debate concerning vendor liability for software security. Many of these issues pose novel challenges in preventing large amounts of losses in money and resources every year in the community and the industry. Our work also helps pave the way for future studies that continue to explore the role of software vendor liability, its effects on vendor security investments and software development, and the social value generated by software in today's increasingly risky global network environment.

8. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

Acknowledgments

The authors thank Sandra Slaughter (the department editor), the associate editor, and two anonymous referees, as well as the seminar participants at University of California, San Diego, and Georgia Institute of Technology for valuable comments and suggestions.

References

Armour, J., W. S. Humphrey. 1993. Software product liability. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Arora, A., R. Telang, H. Xu. 2008. Optimal policy for software vulnerability disclosure. *Management Sci.* **54**(4) 642–656.

Asay, M. 2009. EC takes three steps back on software liability. *CNET News* (May 12), http://news.cnet.com/8301-13505_3-10238475-16.html.

August, T., T. I. Tunca. 2006. Network software security and user incentives. *Management Sci.* **52**(11) 1703–1720.

August, T., T. I. Tunca. 2008. Let the pirates patch? An economic analysis of software security patch restrictions. *Inform. Systems Res.* **19**(1) 48–70.

Beattie, S., S. Arnold, C. Cowan, P. Wagle, C. Wright, A. Shostack. 2002. Timing the application of security patches for optimal uptime. *Proc. 16th USENIX Conf. System Admin.*, USENIX Association, Berkeley, CA, 233–242.

Bennett, J. 2010. NHTSA to test Lexus SUV for rollover. *Wall Street Journal* (April 15), http://online.wsj.com/article/SB10001424052702304510004575185983714141448.html.

Beres, Y., J. Griffin. 2009. Optimizing network patching policy decisions. Technical Report HPL-2009-153, Hewlett-Packard Laboratories, Palo Alto, CA.

Berzon, A. 2010. OSHA boosts oversight of state safety agencies. *Wall Street Journal* (March 29), http://online.wsj.com/article/SB10001424052748034098045751439942004261 92.html.

Bloor, B. 2003. The patch problem: It's costing your business real dollars. http://www.netsense.info/downloads/PatchProblemReport_BaroudiBloor.pdf.

Calabresi, G., K. C. Bass, III. 1970. Right approach, wrong implications: A critique of McKean on products liability. *Univ. Chicago Law Rev.* **38**(1) 74–91.

Cavusoglu, H., H. Cavusoglu, S. Raghunathan. 2007. Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Trans. Software Engrg.* **33**(3) 171–185.

Cavusoglu, H., H. Cavusoglu, J. Zhang. 2008. Security patch management: Share the burden or share the damage? *Management Sci.* **54**(4) 657–670.

Cavusoglu, H., B. Mishra, S. Raghunathan. 2004. The effect of Internet security breach announcements on market value: Capital market reactions for breached firms and Internet security developers. *Internat. J. Electronic Commerce* **9**(1) 69–104.

Chen, P., G. Kataria, R. Krishnan. 2011. Correlated failures, diversification, and information security risk management. *MIS Quart.* Forthcoming.

Chertoff, M. 2008. Remarks by Homeland Security Secretary Michael Chertoff at the Chamber of Commerce on Cybersecurity. Office of the Press Secretary, Washington, DC. http://www.dhs.gov/xnews/releases/pr_1224091491881.shtm.

Choi, J. P., C. Fershtman, N. Gandal. 2010. Network security: Vulnerabilities and disclosure policy. *J. Indust. Econom.* **58**(4) 868–894.

Clarke, R. 2003. Interview: Frontline cyber war! *PBS* (March 18), http://www.pbs.org/wgbh/pages/frontline/shows/cyberwar/interviews/clarke.html.

Collins, B. 2007. Lords tell government to clean up "Wild West" Internet. *PC Pro* (August 10), http://www.pcpro.co.uk/news/broadband/122086/lords-tell-government-to-clean-up-wild-west-internet.

Computer Economics. 2004. The cost impact of major virus attacks since 1995. (February), http://www.computereconomics.com/article.cfm?id=936.

Cox, R. S., S. D. Gribble, H. M. Levy, J. G. Hansen. 2006. A safety-oriented platform for Web applications. *Proc. 2006 IEEE Sympos. Security Privacy*, IEEE Computer Society, Washington, DC, 350–364.

Cusumano, M. A. 2004. Who is liable for bugs and security flaws in software? *Commun. ACM* **47**(3) 25–27.

D'Amico, A. D. 2000. What does a computer security breach really cost? White paper, September 7, Secure Decisions, Applied Visions Inc., Northport, NY.

Dorfman, R. 1970. The economics of products liability: A reaction to McKean. *Univ. Chicago Law Rev.* **38**(1) 92–102.

Epple, D., A. Raviv. 1978. Product safety: Liability rules, market structure, and imperfect information. *Amer. Econom. Rev.* **68**(1) 80–95.

Feinman, J. 2009. OWASP sheds light on its security standards. *SDTimes* (May 13), http://www.sdtimes.com/link/33469.

Fisher, D. 2002. Software liability gaining attention. eWeek.com (January 14), http://mobile.eweek.com/c/a/Application-Development/Software-Liability-Gaining-Attention/.

Forbath, T., P. Kalaher, T. O'Grady. 2005. The total cost of security patch management. White paper, Wipro Technologies, Ltd., Bangalore, India.

Fudenberg, D., E. Maskin. 1986. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica* **54**(3) 533–554.

Garg, A. 2003. The cost of information security breaches. *CrossCurrents* (Spring) 8–11.

Ghosh, B. 2009. How vulnerable is the power grid? *TIME* (April 15), http://www.time.com/time/nation/article/0,8599,1891562,00.html.

Gilmore, G. 1970. Products liability: A commentary. *Univ. Chicago Law Rev.* **38**(1) 103–116.

Goodin, D. 2009. Webhost hack wipes out data for 100,000 sites. *The Register* (June 8), http://www.theregister.co.uk/2009/06/08/webhost_attack/.

Haase, G. 2004. Security liability laws are NOT the answer. OneFreeVoice.com (November 1), http://blog.onefreevoice.com/2004/11/01/security-liability-laws-are-not-the-answer/.

Halfond, W. G., J. J. Viegas, A. Orso. 2006. A classification of SQL-injection attacks and countermeasures. *Proc. IEEE Internat. Sympos. Secure Software Engrg.*, IEEE Computer Society, Los Alamitos, CA.

Heckman, C. 2003. Two views on security software liability: Using the right legal tools. *IEEE Security & Privacy* **1**(1) 73–75.

Hewlett-Packard. 2010. HP TippingPoint zero day initiative. http://www.zerodayinitiative.com/advisories/published/.

Ho, V. 2009. EU software liability law could divide open source. *CNET News* (June 11), http://news.cnet.com/8301-1001_3-10262503-92.html.

IBM. 2008. IBM Internet security systems X-Force® 2008 mid-year trend statistics. Report, IBM Global Technology Services, Armonk, NY.

Joyce, E. 2005. More regulation for the software industry? eSecurity Planet.com (February 16), http://www.esecurityplanet.com/trends/article.php/3483876/More-Regulation-For-The-Software-Industry.htm.

Kaner, C. 1997. Software liability. Unpublished paper, http://www.badsoftware.com/theories.htm.

Keizer, G. 2004. Sasser worm impacted businesses around the world. *TechWeb News* (May 7), http://www.networkcomputing.com/data-protection/sasser-worm-impacted-businesses-around-the-world.php.

Keizer, G. 2007. FAQ: The Monster.com mess. *Computerworld* (August 24), http://www.computerworld.com/s/article/9032518/FAQ_The_Monster.com_mess.

Keizer, G. 2009. Researchers wait for Downadup worm's second act. *Computerworld* (January 23), http://www.computerworld.com/s/article/9126691/Researchers_wait_for_Downadup_worm_s_second_act.

Keizer, G. 2010. Pwn2Own winner tells Apple, Microsoft to find their own bugs. *Computerworld* (March 25), http://www.computerworld.com/s/article/9174120/Pwn2Own_winner_tells_Apple_Microsoft_to_find_their_own_bugs.

Kim, B. C., P. Chen, T. Mukhopadhyay. 2010a. An economic analysis of the software market with a risk-sharing contract. *Internat. J. Electronic Commerce* **14**(2) 7–39.

Kim, B. C., P. Chen, T. Mukhopadhyay. 2010b. The effect of liability and patch release on software security: The monopoly case. *Production Oper. Management*, ePub ahead of print October 19, http://onlinelibrary.wiley.com/doi/10.1111/j.1937-5956.2010.01189.x/full.

Kolstad, C. D., T. S. Ulen, G. V. Johnson. 1990. Ex post liability for harm vs. ex ante safety regulation: Substitutes or complements? *Amer. Econom. Rev.* **80**(4) 888–901.

Krebs, B. 2003. Bush approves cybersecurity strategy. *Washington Post* (January 31), http://www.washingtonpost.com/.

Krebs, B. 2007. Should e-mail addresses be considered private data? *Washington Post* (October 19), http://voices.washingtonpost.com/securityfix/2007/10/database_theft_leads_to_target.html.

Laffont, J.-J., J. Tirole. 1988. The dynamics of incentive contracts. *Econometrica* **56**(5) 1153–1175.

Laffont, J.-J., J. Tirole. 1993. *A Theory of Incentives in Procurement and Regulation*. MIT Press, Cambridge, MA.

Lewis, J. 2009. Innovation and cybersecurity regulation. *CircleID* (March 31), http://www.circleid.com/posts/20090331_innovation_and_cybersecurity_regulation/.

Li, L., R. D. McKelvey, T. Page. 1987. Optimal research for cournot oligopolists. *J. Econom. Theory* **42**(1) 140–166.

MacLeod, W. B., J. M. Malcomson. 1993. Investments, holdup, and the form of market contracts. *Amer. Econom. Rev.* **83**(4) 811–837.

Markoff, J. 2009. Defying experts, rogue computer code still lurks. *New York Times* (August 26), http://www.nytimes.com/2009/08/27/technology/27compute.html.

McBride, S. 2005. Zero day attack imminent. *Computerworld* (February 3), http://www.computerworld.com.au/article/1535/zero_day_attack_imminent/.

McCullagh, D. 2010. Buggy McAfee update whacks Windows XP PCs. *CNET News* (April 21), http://news.cnet.com/8301-1009_3-20003074-83.html.

McKean, R. N. 1970a. Products liability: Implications of some changing property rights. *Quart. J. Econom.* **84**(4) 611–626.

McKean, R. N. 1970b. Products liability: Trends and implications. *Univ. Chicago Law Rev.* **38**(1) 3–63.

Moore, D., C. Shannon, J. Brown. 2002. Code-Red: A case study on the spread and victims of an Internet worm. *Proc. Second ACM SIGCOMM Workshop Internet Measurement*, ACM, New York, 273–284.

Muller, H. M. 2000. Asymptotic efficiency in dynamic principal-agent problems. *J. Econom. Theory* **91**(2) 292–301.

National Infrastructure Advisory Council. 2003. The national strategy to secure cyberspace. Report, U.S. Department of Homeland Security, Washington, DC, http://www.dhs.gov/xlibrary/assets/National_Cyberspace_Strategy.pdf.

Oi, W. Y. 1973. The economics of product safety. *Bell J. Econom. Management Sci.* **4**(1) 3–28.

Payne, J. E. 2004. Regulation and information security: Can Y2K lessons help us? *IEEE Security & Privacy* **2**(2) 32–35.

Pesendorfer, W., J. M. Swinkels. 2000. Efficiency and information aggregation in auctions. *Amer. Econom. Rev.* **90**(3) 499–525.

Png, I. P. L., Q.-H. Wang. 2009. Information security: User precautions, attacker efforts, and enforcement. *Proc. 42nd Hawaii Internat. Conf. System Sci.*, IEEE Computer Science, Los Alamitos, CA, 1–11.

Polinsky, A. M. 1980. Strict liability vs. negligence in a market setting. *Amer. Econom. Rev. Papers Proc.* **70**(2) 363–367.

Poulsen, K. 2003. Slammer worm crashed Ohio nuke plant network. *SecurityFocus* (August 19), http://www.securityfocus.com/news/6767.

Ransbotham, S., S. Mitra. 2009. Choice and chance: A conceptual model of paths to information security compromise. *Inform. Systems Res.* **20**(1) 121–139.

Ransbotham, S., S. Mitra, J. Ramsey. 2011. Are markets for vulnerabilities effective? *MIS Quart.* Forthcoming.

Rubinstein, A. 1979. Equilibrium in supergames with the overtaking criterion. *J. Econom. Theory* **21**(1) 1–9.

Ryan, D. J. 2003. Two views on security software liability: Let the legal system decide. *IEEE Security & Privacy* **1**(1) 70–72.

SANS Institute. 2009. The top cyber security risks. Report, SANS Institute, Bethesda, MD.

Schneider, F. B. 2005. It depends on what you pay. *IEEE Security & Privacy* **3**(3) 3.

Schneier, B. 2004. Information security: How liable should vendors be? *Computerworld* (October 28), http://www.computerworld.com/s/article/96948/Information_security_How_liable_should_vendors_be_.

Schneier, B. 2008. Software makers should take responsibility. *The Guardian* (July 17), http://www.guardian.co.uk/technology/2008/jul/17/internet.security.

Schweitzer, D. 2003. Emerging technology: Patch me if you can! NetworkMagazine.com. (August 5).

Shavell, S. 1982. On liability and insurance. *Bell J. Econom.* **13**(1) 120–132.

Singel, R. 2007. Zero-day Malware attacks you can't block. *PC World* (February 21), http://www.pcworld.com/article/129020/zeroday_malware_attacks_you_cant_block.html.

Spence, M. 1977. Consumer misperceptions, product failure and producer liability. *Rev. Econom. Stud.* **44**(3) 561–572.

Timms, S., C. Potter, A. Beard. 2004. Information security breaches survey 2004. UK Department of Trade and Industry, Apr.

Tirole, J. 1988. *The Theory of Industrial Organization*, 1st ed. MIT Press, Cambridge, MA.

U.S. Department of Homeland Security. 2007. National strategy for homeland security, October. Homeland Security Council, Washington, DC.

U.S. Department of Transportation. 2004. Federal motor vehicle safety standards and regulations. Report, National Highway Traffic Safety Administration, Washington, DC. http://www.nhtsa.gov/cars/rules/standards/FMVSS-Regs/index.htm.

Vereshchagina, G., H. A. Hopenhayn. 2009. Risk taking by entrepreneurs. *Amer. Econom. Rev.* **99**(5) 1808–1830.

Viscusi, W. K., M. J. Moore. 1993. Product liability, research and development, and innovation. *J. Political Econom.* **101**(1) 161–184.

Wan, Z., D. Beil. 2009. RFQ auctions with supplier qualification screening. *Oper. Res.* **57**(4) 934–949.

Weise, E. 2009. "All hands on deck" as FDA aims to fix food safety. *USA Today* (July 7), http://www.usatoday.com/news/health/2009-07-07-food-safety-guidelines_N.htm.

Werth, C. 2009. Software crackdown. *Newsweek* (July 25), http://www.newsweek.com/2009/07/25/software-crackdown.html.

Worthen, B., J. E. Vascellaro. 2010. Google attackers gained access to computer code. *Wall Street Journal* (April 20). http://online.wsj.com/article/SB10001424052748703757504575194873659727984.html.

Zhou, Z. Z., M. E. Johnson. 2010. The impact of professional information risk ratings on vendor competition. Working paper, Rady School of Management, University of California, San Diego, La Jolla.