Title

Cloud Implications on Software Network Structure and Security Risks

Journal

Information Systems Research, 25(3)

ISSN

1047-7047

Authors

August, Terrence
Niculescu, Marius Florin
Shin, Hyoduk

Publication Date

2014-09-01

DOI

10.1287/isre.2014.0527

Cloud Implications on Software Network Structure and Security Risks

Terrence August*
Rady School of Management
University of California, San Diego
Korea University Business School

Marius Florin Niculescu†
Scheller College of Business
Georgia Institute of Technology

Hyoduk Shin‡
Rady School of Management
University of California, San Diego

Apr 16, 2014

Abstract

By software vendors offering, via the cloud, software as a service (SaaS) versions of traditionally on-premises application software, security risks associated with usage become more diversified which can greatly increase the value associated with the software. In an environment where negative security externalities are present and users make complex consumption and patching decisions, we construct a model that clarifies whether and how SaaS versions should be offered by vendors. We find that the existence of version-specific security externalities is sufficient to warrant a versioned outcome, which has been shown to be suboptimal in the absence of security risks. In high security-loss environments, we find that SaaS should be geared to the middle tier of the consumer market if patching costs and the quality of the SaaS offering are high, and geared to the lower tier otherwise. In the former case, it is noteworthy that strategic interactions between the vendor and consumers can lead a lower inherent quality product to actually be preferred by a higher tier customer segment when security risk associated with each version is endogenously determined by consumption choices. Relative to on-premises benchmarks, we find that software diversification indeed leads to lower average security losses for users when patching costs are high. However, when patching costs are low, surprisingly, average security losses can actually increase as a result of SaaS offerings and lead to lower consumer surplus. We also investigate the vendor's security investment decision and establish that the vendor tends to increase investments in an on-premises version and decrease investments in a SaaS version as the market becomes riskier. On the other hand, in low security-loss environments, we find that SaaS is optimally targeted to a lower tier of the consumer market, average security losses decrease, and consumer surplus increases as a result. Security investments increase for both software versions as risk increases in these environments.

Key words: cloud computing; Software-as-a-Service; network economics; security; versioning; on-premises software.

*Rady School of Management, University of California, San Diego, La Jolla, CA 92093-0553. Visiting IlJin Professor, Korea University Business School, Seoul, Korea, 136-701. e-mail: taugust@ucsd.edu

†Scheller College of Business, Georgia Institute of Technology, Atlanta, GA 30308. e-mail: marius.niculescu@scheller.gatech.edu

‡Rady School of Management, University of California, San Diego, La Jolla, CA 92093-0553. e-mail: hdshin@ucsd.edu

1 Introduction

With broadband access becoming faster and more pervasive, there has been a shift back toward models where computing is centralized and accessed via thin clients. Both firms and governments are starting to implement cloud-based systems to support business processes and increase operational efficiency. For example, the U.S. government, which has an $80 billion federal IT budget, has championed a Federal Cloud Computing Initiative to encourage agencies to move toward cloud computing solutions, supporting this transition with Apps.gov (Claburn 2009).[1] Gartner estimates that the cloud computing industry will grow to $149 billion by 2015 (Kundra 2011). Vivek Kundra, former chief information officer of the U.S., also suggested cloud computing will help increase productivity in health care, financial services, and education, pointing out that a one percent productivity increase in health care over the next 10 years represents $300 billion in value (e.g., shifting electronic medical records to the cloud).

Cloud computing is not likely to be an "end all" solution. Rather, for many firms, it will become an increasingly important component of an overall IT strategy, augmenting the traditional use models currently employed (O'Neill 2011). Among the many opportunities for engagement, cloud computing can be leveraged across diverse service models: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Of particular interest to application software companies and the consumers of their products is the SaaS model.[2] This model has emerged both out of economic efficiency and to satisfy users' additional preferences for their data and applications to be ubiquitous. Over the last two decades, consumers have harnessed SaaS applications for personal e-mail, online gaming, photo sharing, and social networking. Businesses are also utilizing SaaS versions of productivity software, enterprise resource planning software, and more; according to a recent survey by InformationWeek, three quarters of the "companies using SaaS consider application services extremely or critically important to their organizations" (Biddick 2010). For example, to address these market needs, Microsoft has added Microsoft Office 365, a SaaS variant of its well known Microsoft Office suite which offers enterprise browser-based Office

[1] NIST defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell and Grance 2011).

[2] NIST also characterizes the SaaS service model as follows: "The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings" (Mell and Grance 2011).

Web Apps, to its product portfolio (El Akkad 2011). Similarly, SAP offers SAP Business ByDesign, targeting this version to small businesses (Farber 2007).

When software vendors like Microsoft and SAP offer SaaS versions of traditionally locally hosted software (often called *on-premises* because such software is installed on premises at the customer's location), there are significant security risk implications. This versioning strategy greatly affects the incentives of users, which in turn determines how aggregate usage gets spread out across the versions and to what extent users engage in secure behaviors such as patching. To better understand the impact of versioning on security risk, it is useful to first discuss the nature of attacks faced by application software products. Security attacks can be broadly categorized into two classes: *directed* (targeted) and *undirected* (non-targeted).[3] A directed attack occurs when a malicious actor expends effort in an attempt to compromise a specific target (Villeneuve 2011). In contrast, an undirected attack typically involves self-propagating malware (e.g., a computer worm) designed to spread to and infect many vulnerable hosts, which are often running a common software application that contains an exploitable vulnerability.

Users and organizations are generally exposed to a wide range of directed and undirected attacks due to flaws in hardware and software technologies, poor configurations, and weaknesses in individual decision making. However, in order to isolate the differential impact on security risk associated with a software vendor offering distinct on-premises and SaaS versions of a software product, it is preferable to focus attention only on the attacks that specifically exploit vulnerabilities in the code of one of these two versions (essentially holding all other attacks unrelated to vulnerabilities in the variants of this product constant). With this lens, a directed attack refers to an attack explicitly directed toward a *particular system* running one of these versions and exploiting a vulnerability specific to the version. Similarly, an undirected attack refers to an attack that indiscriminately exploits one of these versions on *any system* running a copy of the version's software.

Focusing on a single software product and the idiosyncratic security risk stemming from each of its versions, the on-premises version has relatively higher undirected risk and the SaaS version has relatively higher directed risk. To see why, let us suppose SAP Business One (on-premises) contains vulnerability V1, SAP Business ByDesign (SaaS) contains vulnerability V2, and a firm is considering the risks associated with each version. If the firm, along with many other companies with similar needs, chooses to use SAP Business ByDesign, then the firm is exposed to considerable directed

[3]The security attack landscape is quite broad, including phishing, brute-force, social engineering, packet sniffing, SQL injection, keylogging, and more. However, most attacks can be grouped into these two classes, directed and undirected attacks.

risk. In particular, a directed attack on SAP's systems (running Business ByDesign) that exploits vulnerability V2 enables a malicious attacker to affect *many* organizations all at once. On the other hand, if the firm uses SAP Business One instead, its exposure to directed risk is much lower. This is because an attacker's incentive to exploit vulnerability V1 on the firm's system (running Business One) is reduced as it would only be affecting an *individual* organization. Instead, because on-premises versions like Business One are often widely deployed on independent but interconnected systems each containing vulnerability V1, a malicious hacker could inflict significantly more damage by using the same vulnerability to attack the entire network in one broad, undirected attack, affecting *many* users.

Because of the large number of installations associated with on-premises software, a software vendor usually manages vulnerabilities by creating and distributing software patches to users. However, it has been historically difficult to incentivize users to patch their own installations when security patches are released. Thus, for on-premises offerings, the user network is characterized by a large number of widespread nodes where individual instances of the software are running with many remaining unpatched (Lemos 2004, Keizer 2008). Even six years after the Conficker worm first struck, the malicious program is still infecting computers (Robertson 2014). Poor user patching behavior increases the risk faced by other users and reduces the value of the software vendor's product as a result. Code Red, SQL Slammer, Sasser, and Conficker are all examples of malware that spread across vulnerable software networks and caused sizable economic damages (Moore et al. 2002, Lemos 2003, Keizer 2004, Markoff 2009).[4] Microsoft's webserver software, Internet Information Services (IIS), and its database management system, SQL Server, are examples of software products being compromised by these undirected attacks.

Because systems servicing large populations of consumers are attractive targets to hackers incentivized by economic and/or political motivations, SaaS versions have considerable exposure to directed risk as discussed above. Recently Adallom, a SaaS security company, observed an exploited vulnerability in Office 365's token management that enables an attacker to steal a user's token and thus the user's access to the software (Messmer 2013, Liran 2013). Similarly, Office 365 also had a cross-site scripting vulnerability that would enable an attacker to gain administrator access to an organization's Office 365 account and configuration (Lee 2014). In both of these cases, directed attacks on Microsoft's systems to exploit vulnerabilities in the implementation of Office 365 can

[4]Notably, in all of these examples, the malware exploited known vulnerabilities for which a security patch had already been made available to users before the attacks occurred.

cause its users to incur substantial economic losses.

When a SaaS variant of an on-premises product is introduced into the market, the aggregate security risk can be affected in several ways. First, as discussed above, users of the SaaS version are exposed to significantly less *product-specific* undirected risk. Second, an increase in SaaS usage helps reduce total risk by diversifying exposure across undirected and directed attacks and limiting the sizes of populations that malware can effectively target; this, in turn, may indirectly reduce the incentives of malware developers to target diversified software (Bain et al. 2002 and Kannan et al. 2013). Third, a software vendor who expands its product line to offer distinct SaaS and on-premises versions may further expand usage through pricing which indirectly affects security risk. Finally, such a product line expansion to include SaaS may act as a substitute to investments in making on-premises software products more secure.

In recent years, software companies have invested substantial resources in (i) developing SaaS versions of their existing, on-premises software products, and (ii) increasing the security of their on-premises products (Charney 2012). It is important for companies like Microsoft to better understand how versioning and security are intimately related because of the inherent risk interdependencies. For example, when Microsoft brought its SaaS service Office 365 to the market, it faced a challenging question of how to manage both Office 365 and its on-premises counterpart, Office, so as to both cater to consumers' heterogeneous preferences and also harvest the risk diversification benefits that stem from having two versions with separate user populations. One question is which consumer segments should be the targeted users of the SaaS and on-premises versions. Given the strong market for Office 365, Microsoft also may have reconsidered whether its substantial investments in security are still necessitated when its product line expands to achieve security through diversity.

Similar challenges are being faced by providers of enterprise application server software (e.g., Oracle WebLogic Server, IBM WebSphere, etc.), database management systems (e.g., Oracle Database, SQL Server, MySQL, etc.), and customer relationship management software (e.g., Microsoft Dynamics, SAP 360, Oracle CRM, etc.).[5] Not every provider of on-premises software will find it economical to offer SaaS alternatives in addition to their traditional on-premises offerings because of differences in fixed costs of development, business strategy, and individual expertise.

[5]Any class of technologies where one version can be deployed on the vendor's systems, accessed by consumers remotely, and utilized by consumers in a manner where they still derive value similar to the other version which is deployed on local, internal systems, is amenable to analysis by our model. The two distinct deployment paradigms used by the vendor-managed version and consumer-managed version lead to different usage network structures which are characterized by idiosyncratic security risks.

However, for the software vendors who are either already or in the process of providing both on-premises and SaaS solutions, better product design, consumer segment targeting, and pricing decisions can be made with an improved understanding of the security implications of this type of versioning.

In this paper, we study the security impact of an application software vendor's versioning strategy to include a SaaS variant of its on-premises product. We have several research goals that drive our study. First, when both SaaS and on-premises offerings are available, we want to better understand how consumers separate across product variants in order to diversify security risk. Thus, we account for how directed and undirected attacks generate distinct security externalities under each paradigm, and build a model that captures consumer incentives to use either variant cognizant that (i) users of the on-premises alternative who choose not to patch impose undirected risk on other users, and (ii) directed risk associated with the SaaS alternative is proportional to the size of the total consumer population who opts for the SaaS variant. With this model of consumer behavior, we fully characterize the equilibrium consumer market structure in which the idiosyncratic security externalities are endogenously determined by consumption choices.

Using this equilibrium characterization, a second research goal is to develop an understanding of how a software vendor approaches the versioning problem when the vendor has the potential to offer both SaaS and on-premises versions. The first question to answer is whether a vendor should version his software or simply offer a single on-premises product. Second, we aim to comment on which consumer market segments should be the intended users of each version if the vendor finds it optimal to pursue a versioning strategy. Because of the distinct security externalities, we study whether the vendor's strategic behavior can yield unexpected strategies in the software vendor's product differentiation and pricing problem.

A final research goal is to quantify the risk diversification benefits of a versioning strategy in this context. Using our framework, we compare measures of profitability, security losses, consumer surplus and social welfare against benchmark measures stemming from a model of a single, on-premises product that faces only undirected security risk. With these comparisons, we can clarify under which risk environments versioning has greater potential, and we can also show how a software vendor's security investment decision interacts with the versioning strategy being employed.

Briefly, we summarize the main findings in our study. One important implication highlighted by our model is that a software vendor may set prices to target an inherently lower quality SaaS product to consumers with higher valuations ("types") than those who consume a higher quality

on-premises version. In a standard vertical product differentiation setting, the lower quality product typically serves consumers with low types. We demonstrate that the strategic interactions between the software vendor and the consumers can drive the opposite result where the vendor prices the SaaS product high so as to create a small SaaS user population which will limit the directed security risk associated with this product. By offering a lower inherent quality product but with better, endogenously-determined security properties at a higher price, the vendor makes its consumption incentive compatible for the higher type consumers who neither find it cost effective to consume the higher inherent quality on-premises product in a patched state nor want to be exposed to the significant undirected risk associated with the on-premises product in an unpatched state.

By studying how the software vendor sets prices of SaaS and on-premises offerings and the consumer market structures induced as a result, our findings contribute to a better understanding of versioning problems of digital goods. We establish that even with uniformly distributed types and zero marginal costs (a case where the literature has established that vendors do not offer separate versions), a software vendor will find it optimal to version his product as long as each version has some idiosyncratic risk. Our finding remains valid even as the level of risk becomes small. Because the argument only requires each version to have minimal idiosyncratic risk (which merely requires subtle differences between software versions), our finding offers some explanatory power of the many versions being offered by most software producers.

When comparing economic measures between the versioned (SaaS and on-premises) outcomes with those of a benchmark scenario (on-premises only), we highlight several interesting implications. First, we formally establish that average security losses per user can actually increase when a software vendor introduces a SaaS version in high security-loss environments. The risk diversification benefits are outweighed by the vendor's pricing behavior with regard to inducing risky consumption. We also show that consumer surplus can decrease in these cases if the inherent quality of the SaaS alternative is sufficiently high. We demonstrate that these effects only exist in high security-loss environments, proving that in low security-loss environments, security losses always decrease and consumer surplus always increases as a result of the vendor's versioning decision. We also establish that the potential gains in profitability and social welfare relative to benchmarks stemming from a versioning strategy are much larger for high security-loss environments. In fact, in such environments, because of the software vendor's strategic behavior with regard to how it targets its SaaS version to the various consumer market segments, we highlight an opportunity to increase social welfare if the vendor can be encouraged to target SaaS to higher consumer types in

certain cases where he prefers his SaaS version to serve the lower tier of the consumer market.

2 Literature Review

Our paper bridges together three distinct research areas found within the literature on the economics of information systems, economics of information security, and computer science. In particular, these three areas are versioning of information goods, interdependent security risk, and software diversification, respectively. In this section, we describe the current research landscape in each of these areas and discuss in detail the contribution of our paper, which ties the areas together and advances our understanding of the interaction between SaaS versioning and security. We also discuss several papers that are connected to our work and study security and SaaS business models.

Versioning of information goods: Product differentiation is an important topic studied in economics, and similarly the versioning of information goods is a topic actively studied by the information systems community. Given the ease with which information goods can be versioned and reproduced, digital content owners typically offer several versions of their content which can readily be seen in the software, movie, and music industries. The questions of when and how to version as well as which consumer segments to target each version with pricing are all relevant concerns. There is a rich stream of literature on these topics, examining how the versioning decision relates to cost-to-quality ratios, consumer heterogeneity, positive network effects, competition, asymmetric information, group tastes, and free disposal concerns (see, e.g., Bhargava and Choudhary 2001, 2008, Johnson and Myatt 2003, Jing 2007, Jones and Mendelson 2011, Wei and Nault 2011, 2014, Chellappa and Jia 2011, Chellappa and Mehra 2013, Niculescu and Wu 2014).

In this literature, a common point of concern is that when consumers are heterogeneous in their taste for quality and this taste parameter is uniformly distributed, a software vendor will not find it optimal to version its product. This holds true for information goods where the marginal costs of reproduction are zero. Because such a result is not readily observed in practice, the papers listed above demonstrate conditions under which versioning is optimal by introducing realistic adaptations from this base model. We contribute to this understanding of versioning by examining how the software vendor reacts if each version of his product carries some *idiosyncratic* security risk that endogenously arises as a result of pricing and consumption behaviors in equilibrium. Using our model, we formally establish that even with uniformly distributed tastes and zero marginal costs, a software vendor will always find it optimal to version his product provided each version has some

idiosyncratic risk, even as this risk becomes negligible. Our finding is consistent with the nature of versions being offered by software providers which typically differ on various functionalities and satisfies this criteria of having minimal idiosyncratic risk. In this sense, one contribution of our paper is to demonstrate that security differences in product versions can swing the versioning decision.

Software diversity: Beyond this contribution to the versioning literature, one of the main goals of our paper is to show that for software exposed to security threats, versioning can have substantial implications on the equilibrium security levels faced by consumers. In this vein, our work connects to the literature on software diversity. There is considerable work examining the risks of having an IT monoculture and determining methods that can achieve diversity. IT monoculture refers to deploying similar systems running similar software (Lala and Schneider 2009). Both within and across organizations, a monoculture strategy reduces the cost of learning, management, configuration, and maintenance. However, similar systems share common vulnerabilities which puts entire networks of systems running common software at risk from large-scale attacks. In this literature, researchers have explored how to introduce artificial diversity via memory randomization (Forrest et al. 1997, Xu et al. 2003, Schneider and Birman 2009), additional redundancy with N-variant systems (Cox et al. 2006, Weatherwax et al. 2009, Gherbi et al. 2011), and, more recently, how diversity results from equilibrium actions in game-theoretic settings (Neti et al. 2012). In Chen et al. (2011), the authors are the first to construct a model that explores the trade-offs among increased security through software diversity, lost network effects, and economies of scale. They find that a firm can benefit more from diversification as software begins to use more standardized interfaces and when adapters and middleware are available to keep applications compatible.

An important consequence of the movement toward cloud-based SaaS offerings is it can indirectly introduce diversity. For example, Microsoft Office 365 includes enterprise Office Web Apps which is a virtual version of the most common Office tools such as Word, Excel and PowerPoint. By providing both a typical on-premises version and a SaaS one, Microsoft achieves software diversity in its Office suite as a consequence of naturally catering to its heterogeneous customer preferences. With the current movement, there is a fitting opportunity to add to the discussion on software diversity by examining settings where consumer demands have driven the need for a particular type of diversity, which can then be leveraged as an opportunity to simultaneously improve security. Our paper contributes to this stream of literature by formally studying how software diversity

stemming from SaaS offerings impacts the security risk properties of the network of users, as driven by benefits from usage becoming diversified across versions and changes in security behaviors (e.g., patching and security investment). In our paper, we quantify these benefits by comparing measures of profitability, security losses, consumer surplus and social welfare to analogous measures in benchmark scenarios absent of software diversity.

Interdependent security risk: A third stream of research closely related to our work centers on security interdependence. One particularly relevant type of interdependence relates to how users of software running on interconnected networks impose security externalities on one another through their usage choices and patching decisions. We examine the diversification benefits associated with having two separate types of risk, directed and undirected, noting that with undirected risk users typically make decisions on whether or not to patch their individual systems. Therefore, it is important for us to build upon prior work that focuses specifically on the trade-off between patching and being exposed to undirected risk. In particular, we build upon the foundational model in August and Tunca (2006) which captures how risk faced by unpatched users is related to the number of users who choose to be unpatched in equilibrium. August and Tunca (2006) focus on examining how patching rebates, mandates, and taxes can improve software security, whereas our research goals focus on security risk diversification through versioning. However, by extending the model in August and Tunca (2006), we can compare security properties of the risk-diversified network when *jointly* offering SaaS and on-premises versions to benchmarks from the base model.

While there are many other studies also examining phenomena that involve interdependent security risks (e.g., Kunreuther and Heal 2003, Heal and Kunreuther 2007, August and Tunca 2008, Choi et al. 2010, August and Tunca 2011, Hui et al. 2013, Nochenson et al. 2014), we contribute to this literature by investigating how risk interdependence can be mitigated by designing product substitutes with separate, idiosyncratic risks and allowing users to make consumption decisions that endogenously determine the aggregate security risk on the network. Given the scale of economic damages associated with security attacks and the substantial investments in security being made by software providers (Judge 2002, Lewis and Baker 2013), our paper provides insights into the value software versioning strategies have on overall security. These insights are useful to software managers who (i) are making decisions whether to offer SaaS variants of traditional software packages, and (ii) have also traditionally determined investment levels in product security of on-premises products, which can act as substitutes to versioning strategies for risk diversification.

Thus, our research goals are uniquely different than extant research, but clearly complement this body of knowledge which aims to put forth a better understanding of managing security risk in the presence of security externalities.

Our paper is related to the broad area studying the economics of information security.[6] We complement research streams on piracy (August and Tunca 2008, Lahiri 2012, Kannan et al. 2013), software liability (Cavusoglu et al. 2008, Kim et al. 2010, 2011, August and Tunca 2011), vulnerability disclosure (Cavusoglu et al. 2007, Arora et al. 2008, Choi et al. 2010), and markets for security (Kannan and Telang 2005, Dey et al. 2012, Ransbotham et al. 2012, Lee et al. 2013), which, similar to our work, all study particular facets of the security problem and recommend strategies to manage risk and improve the value derived from software.[7] Png and Wang (2009) examine the role of government in facilitating end-user precautions and enforcing laws against attackers, considering both directed and undirected attacks. In our model, we examine how a diversification strategy (releasing both SaaS and on-premises versions) affects directed and undirected attacks on *product-specific* vulnerabilities in order to analyze its aggregate impact on security risk.

Lastly, our work is related to several papers that study various aspects of SaaS versus on-premises business models. Choudhary (2007) examines how SaaS versus perpetual licensing affect a software vendor's incentives to invest in quality. In a two-period model, he establishes that the vendor tends to invest more in quality under a SaaS scheme and that both profits and welfare increase as a result. Zhang and Seidmann (2010) study the licensing problem under network effects and quality uncertainty. They demonstrate that under strong network effects, hybrid models are favorable; in our work, we establish a similar result driven by security risk diversification benefits in contrast to multi-period dynamics. Huang and Sundararajan (2005) take a more general approach to pricing to characterize optimal non-linear prices of on-demand computing, while Ma and Seidmann (2014) study competition between various software providers. In our model, we simplify the structure of the SaaS and on-premises alternatives, using a static model that abstracts away from upgrade cycles and multi-period pricing, in order to elegantly capture software security risk concerns which are the focus of our paper.

[6] Anderson (2001), Gordon and Loeb (2002), Anderson and Moore (2006), Grossklags et al. (2008), and Johnson (2008) together provide a comprehensive introduction to important themes in this research area.

[7] We abstract away from specific timing issues related to vulnerability disclosure for which there is a relatively well-developed literature.

3 Model Description

A vendor produces software and offers it to a continuum of consumers. The software can be made available in two formats: (i) *as a product* to be installed at the consumer's location (on-premises), and (ii) *as a service* which is installed only on the vendor's systems and accessible by users over the Internet (SaaS). SaaS versions of common software products (e.g., SAP Business ByDesign, Microsoft Office 365, and Microsoft Dynamics CRM On-Demand) typically are streamlined to be easier to use but include less functionality, require less setup, and offer less integration. In other words, a consumer of a SaaS version tends to forgo some flexibility in integration with business systems, ability to control data, and manage upgrading (Chow et al. 2009). On the other hand, consumers can derive greater value through a more comprehensive integration with an on-premises version of software that is installed internally and can connect with other systems. Therefore, we model consumer valuations for the on-premises version to be uniformly distributed on $\mathcal{V} = [0, 1]$ and assume that if a given consumer has valuation $v \in \mathcal{V}$ for the on-premises version, she has valuation δv for the SaaS version where $0 < \delta < 1$.

We assume that the software is used in a network setting, thereby exposing purchasing consumers to additional risk associated with the software's use. This risk comes in the form of directed and undirected security attacks, which are both described in Section 1. We denote the probability that a directed attack occurs on the network with $0 < \pi_d < 1$ (we use d to signify *directed*). Conditional on a directed attack having occurred on the network, we assume that the likelihood any given network location (node) is victimized is proportional to the mass of consumers at that node (Greenemeier and Hoover 2007, Roy 2011). Therefore, the total likelihood of a node that services d consumers being attacked is $\pi_d d$. Similarly, we denote the probability that a patchable security vulnerability arises in the software and that an undirected attack on that vulnerability occurs with $0 < \pi_u < 1$ (we use u to signify *undirected*). Given the spreading mechanics of undirected security attacks such as worms, if the mass of the unpatched population in the network is u, the unconditional probability that the worm will attack an unpatched user's system is given by $\pi_u u$.

Because the SaaS version is only installed at one node (on the vendor's system), under the above model specification users of the SaaS version are primarily exposed to directed risk, which increases with the size of the total SaaS user population. In contrast, the on-premises version carries minimal directed risk because a given node is negligible in comparison to the size of the total number of on-premises nodes (a subset of the continuum $\mathcal{V}$). However, because of these

many widespread nodes running the on-premises product, this version is exposed to considerable undirected risk which is proportional to the size of this user base that remains unpatched. In our model specification, we have made an effort to maintain the simplest structure where the SaaS and on-premises versions face idiosyncratic security risks, each having a security externality dependent on user behavior. Such a structure will permit us to analytically explore how versioning impacts security and produce clear insights.

If a user gets struck by either a directed or undirected security attack, then one would expect that she suffers a loss positively correlated with her valuation. That is, consumers with high valuations will incur greater losses than consumers with lower valuations due to higher opportunity costs, higher criticality of data and loss of business. For simplicity, we assume the correlation is of first order, i.e., the loss that a consumer with valuation v incurs if she is hit by the attack is either αv for on-premises or $\alpha \cdot \delta v$ for SaaS, where $\alpha > 0$ is a constant. Undirected attacks typically exploit known vulnerabilities for which a patch is already available, hence each consumer has an opportunity to patch in the face of this security risk.[8] If a consumer chooses to patch the software, she will incur an expected cost of patching denoted $0 < c_p < 1$, which accounts for the money and effort that a consumer must exert in order to verify, test, and roll-out patched versions of existing systems.

There are three decision periods. In the first period, the vendor determines which versions of its software to release and sets a product price $p > 0$ for a single server license for its on-premises version and a service price $p_s > 0$ for its SaaS version.[9] In the second period, given the price and security risk of each software offering, each consumer makes a decision whether to purchase the software as well as which version to purchase. Finally, in the third period, if a patchable security vulnerability has been discovered, each consumer who purchased the on-premises version determines whether or not to patch her own instance. Subsequent to these decision periods, both directed and undirected attacks may realize on the network and consumers incur losses.

Each consumer makes a purchasing decision to buy the on-premises product version, OP, buy

[8] Zero-day attacks on vulnerabilities that do not have a patch available yet can also occur (see, e.g., McBride 2005, IBM 2008), and are central to the debate on software liability because users cannot protect themselves from these risks. In this paper, we focus on patchable vulnerabilities and refer the reader to August and Tunca (2011) which helps build an understanding of how our insights will apply as zero-day attacks become more widespread.

[9] As mentioned in the literature review, we use a static model of product and service offerings in order to focus on security risk issues stemming from consumer usage and patching behavior. We consider each offering to provide value to the consumer for an equivalent fixed amount of time for a fixed price in order to abstract away from many ancillary issues, such as upgrade cycles and dynamic pricing in a dynamic model, while centering in on issues related to security risk diversification. Hence, the price for SaaS (p_s) should be carefully interpreted as the service price for the same period as the server license of the on-premises version.

the SaaS version, $SaaS$, or not buy either offering, N. Similarly, if a patchable vulnerability arises in the software, each user of the on-premises version makes a decision to either patch, P, or not patch, NP, her own system. If the consumer has chosen $SaaS$ or N, then she does not make a patching decision as in the on-premises case, which we denote by ND. We denote the consumer action space by $S = (\{OP\} \times \{P, NP\}) \cup (\{SaaS, N\} \times \{ND\})$. Given prices, in a consumer market equilibrium, each consumer maximizes her expected utility given the equilibrium strategies of all other consumers. For a strategy profile $\sigma : \mathcal{V} \to S$, the expected cost faced by the consumer with valuation v is then defined by

$$C(v,\sigma) \triangleq \begin{cases} c_p & if \quad \sigma(v) = (OP, P)\,; \\ \pi_u u(\sigma)\alpha v & if \quad \sigma(v) = (OP, NP)\,; \\ \pi_d d(\sigma)\alpha\delta v & if \quad \sigma(v) = (SaaS, ND)\,; \\ 0 & if \quad \sigma(v) = (N, ND)\,, \end{cases} \tag{1}$$

where the size of the unpatched user population of the on-premises version is given by

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

and the size of the user population of the SaaS version (most vulnerable to a *directed* attack) is given by

$$d(\sigma) \triangleq \int_{\mathcal{V}} \mathbb{1}_{\{\sigma(v) = (SaaS,ND)\}}\, dv\,, \tag{3}$$

where $\mathbb{1}_{\{.\}}$ is the indicator function. For expositional convenience, we also define the size of the patched population using the on-premises product to be

$$n(\sigma) \triangleq \int_{\mathcal{V}} \mathbb{1}_{\{\sigma(v) = (OP,P)\}}\, dv\,. \tag{4}$$

Our main research goal is to assess the security impact that stems from risk diversification benefits associated with a software vendor's versioning strategy as it extends into SaaS markets. In order to do so, we use the benchmark model from August and Tunca (2006) for comparison of security risk characteristics. In their model of undirected security risk, users make usage and patching decisions, and the vendor sets the product price. In this paper, we model the on-premises software product so it is consistent with August and Tunca (2006).[10] However, in order to study

[10] We modify notation slightly to differentiate between directed and undirected attack probabilities. Specifically,

risk diversification in the context of SaaS versioning, we also model a SaaS version with its own idiosyncratic risk. This directed risk reflects a security externality that is structurally different and unique from the externality stemming from unpatched usage of the on-premises version. By modeling both on-premises and SaaS versions with separate externalities, we can analyze how versioning in this manner affects pricing and user incentives, which then determine the security characteristics of this more complex software network. Finally, by being consistent with prior work, when $\delta = 0$, the SaaS version has no inherent value to users (in which case consumers only either purchase on-premises or remain out of the market), and then our model collapses to August and Tunca (2006) which we will refer to as the benchmark case, i.e., having only an on-premises offering.

4 Consumer Choice and Vendor Profit Maximization

4.1 Consumer Market Equilibrium

In order to study the software vendor's versioning problem and the security properties of the network that ensue as a consequence, we first develop an understanding of how consumers strategically determine whether to adopt an on-premises or SaaS solution. In this section, we take prices as given and study the choice problem faced by consumers who strategically interact due to version-specific security externalities associated with each alternative. Holding all other consumers' strategies fixed to σ_{-v}, the consumer with valuation v determines her optimal action by solving the following maximization problem

$$\max_{s \in S} \quad (v-p) \cdot \mathbb{1}_{\{s \in \{(OP,P),(OP,NP)\}} + (\delta v - p_s) \cdot \mathbb{1}_{\{s=(SaaS,ND)\}} - C(v,\sigma)\,, \tag{5}$$

where the strategy profile σ is composed of σ_{-v} (other consumers' strategies) and the choice being made, i.e., $\sigma(v) = s$. We denote her optimal action that solves (5) with $s^*(v)$. An equilibrium strategy profile σ^* must satisfy $\sigma^*(v) = s^*(v)$ for all $v \in \mathcal{V}$.

In the following lemma, we provide a full characterization of equilibrium consumer behavior for all prices and exogenous security and quality parameters in our model.

Lemma 1 *Given on-premises and SaaS prices, $p \in (0,1)$ and $p_s \in (0,\delta)$, respectively, and other*

we use π_u instead of π as in August and Tunca (2006) to signify undirected risk.

parameters c_p, π_d, π_u, and δ, a unique equilibrium in the consumer market exists.[11] *The equilibrium consumer strategy profile σ^* is characterized by thresholds v_d, v_u, $v_p \in [0,1]$ such that for $v \in \mathcal{V}$, it satisfies either*

$$\sigma^*(v) = \begin{cases} (OP,P) & if \quad v_p < v \leq 1\,; \\ (OP,NP) & if \quad v_u < v \leq v_p\,; \\ (SaaS,ND) & if \quad v_d < v \leq v_u\,; \\ (N,ND) & if \quad 0 \leq v \leq v_d\,, \end{cases} \tag{6}$$

or

$$\sigma^*(v) = \begin{cases} (OP,P) & if \quad v_p < v \leq 1\,; \\ (SaaS,ND) & if \quad v_d < v \leq v_p\,; \\ (OP,NP) & if \quad v_u < v \leq v_d\,; \\ (N,ND) & if \quad 0 \leq v \leq v_u\,. \end{cases} \tag{7}$$

Lemma 1 formally establishes that the consumer market exhibits a threshold structure.[12] Common to both possible equilibrium strategy profiles, as seen in (6) and (7), the consumers with highest valuations for the software choose the on-premises product and patch to avoid undirected security attacks in equilibrium. Thus, there is a patching threshold denoted v_p such that all consumers with valuations above this threshold value use this strategy, i.e., $\sigma^*(v) = (OP,P)$ for all $v \geq v_p$. As motivated before, the on-premises alternative carries the highest inherent quality which reflects the ability of a consumer to more fully integrate this product with her own systems and take advantage of greater functionality. As a result, we would expect the equilibrium outcome to reflect that the highest valuation users prefer on-premises and fully protect their value by patching.

However, one relevant consequence of consumers' strategic behavior that is highlighted by Lemma 1 concerns the effective quality ordering of the SaaS alternative and the on-premises software in an unpatched state. In particular, for the next consumer valuation interval directly below v_p, either unpatched on-premises users, who choose (OP,NP), or SaaS users, who choose $(SaaS,ND)$, compose the subsequent lower set of valuations, corresponding to the strategy profiles in (6) and (7), respectively. Given that consumers inherently prefer on-premises to SaaS (because $v > \delta v$), it is more natural to think that the SaaS version would be consumed by the lowest consumer segment

[11] Technically, when $p = p_s$, consumers can be indifferent between (OP,NP) and $(SaaS,ND)$ such that only the sizes of each population need to be maintained in equilibrium and multiple equilibria exist in that case. However, each consumer's utility and the vendor's profit are the same in all equilibria; hence, in this special case, without loss of generality, we focus on the threshold-type equilibrium presented in the lemma. A more detailed, technical discussion is provided in the proof in the Appendix.

[12] The actual characterization of the thresholds (v_d, v_u, and v_p) is presented in the proof of Lemma 1 as there are many different regions and region-specific equations which these thresholds satisfy.

remaining in the user population as in (6).

The fact that there can exist a segment of consumers choosing $(SaaS, ND)$ in equilibrium and having higher valuations than consumers in a segment choosing (OP, NP) as in (7) firmly demonstrates the role that idiosyncratic security externalities can play in shaping the equilibrium outcome. In this case, the effective quality of the SaaS version when adjusted for its exposure to directed security attacks, which is influenced by pricing and the number of users choosing the SaaS version, can actually be higher than the quality associated with using the on-premises version and not patching. For instance, higher valuation users may prefer slightly lower inherent quality software if it is used by very few users and is considerably more secure than higher inherent quality software with a large unpatched population and considerable undirected risk. Importantly, effective quality of each product is endogenously determined by consumer behavior so it is the strategic interactions driving the effective quality ordering found in (7). Without the on-premises and SaaS versions having unique exposures to different risks, the lower inherent quality product ordinarily would not be consumed by a higher consumer valuation segment.

Similar to our definition of the patching threshold v_p, the SaaS threshold v_d marks the valuation above which (up to the next higher threshold) consumers prefer to use SaaS and tolerate exposure to directed security risk. Lastly, we denote the on-premises purchasing threshold with v_u which marks the valuation above which (again, up to the next higher threshold) consumers prefer to use the on-premises product and not patch in equilibrium. By not patching, these consumers will be exposed to undirected security risk. Because of the threshold structure presented in Lemma 1, there are three distinct consumer market segments represented in equilibrium as characterized by these intervals in the consumer valuation space. For convenience in exposition, we will refer to these intervals as high tier, middle tier, and low tier which correspond to $(v_p, 1]$, $(v_u, v_p]$, and $(v_d, v_u]$, respectively for (6), and $(v_p, 1]$, $(v_d, v_p]$, and $(v_u, v_d]$, respectively, when the characterization in (7) arises in equilibrium. When only two consumer market segments arise in equilibrium (i.e., when $v_p = 1$ meaning no consumer prefers (OP, P) to the other alternatives), then we simply refer to the two, ordered segments as high tier and low tier.

4.2 Vendor Profit Maximization

Next, we formally present the software vendor's pricing problem and define measures of security losses, social welfare, and consumer surplus. Using the measures defined in (2), (3), and (4), the

vendor's profit function can be written

$$\Pi(p, p_s) \triangleq p[u(\sigma^*) + n(\sigma^*)] + p_s d(\sigma^*)\,, \tag{8}$$

where the size of each population depends on the equilibrium strategy profile which, in turn, is a function of prices, i.e., $\sigma^* = \sigma^*(\cdot \,|\, p, p_s)$.[13] The vendor's profit maximization problem can then be expressed

$$\begin{aligned} \max_{(p,p_s)\,\in\,[0,1]\times[0,\delta]} \quad & \Pi(p, p_s) \\ s.t. \quad & (v_d, v_u, v_p) \text{ satisfy } \sigma^*(\cdot \,|\, p, p_s)\,. \end{aligned} \tag{9}$$

In addition to characterizing the optimal prices in (9) and the corresponding equilibrium consumer market structures under these prices, we are also interested in examining measures of security risk, consumer surplus, and social welfare for these outcomes.

To facilitate the ensuing discussion, under a set of prices (p, p_s), we denote the total security losses with SL and define it as the sum of expected losses from undirected security attacks, directed security attacks, and patching costs under the equilibrium strategy profile $\sigma^*(\cdot \,|\, p, p_s)$, i.e.,

$$SL \triangleq \int_{\mathcal{V}} \mathbb{1}_{\{\sigma^*(v)\,=\,(OP,NP)\}} \pi_u u(\sigma^*) \alpha v \, dv + \int_{\mathcal{V}} \mathbb{1}_{\{\sigma^*(v)\,=\,(SaaS,ND)\}} \pi_d d(\sigma^*) \alpha \delta v \, dv + c_p n(\sigma^*)\,. \tag{10}$$

Social welfare can then be measured as

$$W \triangleq \int_{\mathcal{V}} \left[\mathbb{1}_{\{\sigma^*(v)\,\in\,\{(OP,P),\,(OP,NP)\}\}} v + \mathbb{1}_{\{\sigma^*(v)\,=\,(SaaS,ND)\}} \delta v \right] dv - SL\,, \tag{11}$$

which is the difference between the aggregate value derived from the software and these losses. Finally, consumer surplus is defined by

$$CS \triangleq W - \Pi(p, p_s)\,. \tag{12}$$

For the benefit of the reader, we briefly outline how we will structure our presentation of results going forward. Using the equilibrium consumer market characterization above, we next examine

[13] In order to focus on the security aspects of the network, we utilize a simplified cost structure with standard assumptions that the software development costs are sunk and the marginal cost of reproduction is sufficiently small to ignore. As for the SaaS alternative, again there is a fixed cost associated with setting up servers and infrastructure and nominal costs associated with servicing the marginal user. Including this type of cost structure will not add significantly to the insights we generate on how security risk can be managed. However, in Section 7, we do extend our analysis to examine security investments by the vendor.

high security-loss environments (high α) and *low* security-loss environments (low α) separately in the subsequent two sections. As noted above, because of the complexity of the general characterization of the equilibrium, it is too extensive to fully include in the exposition. However, when focused on either a high or low security-loss environment, this equilibrium characterization simplifies considerably. Thus, for each environment, we first present greater details on parameter boundaries of feasible regions. We then examine the vendor's pricing problem and study his versioning decision. In light of the vendor's optimal pricing and the associated consumer market outcome, we carefully examine how versioning affects security, consumer surplus and social welfare, as determined by patching costs (c_p), SaaS quality (δ), and the security loss factor (α). As part of our analysis, we compare these outcomes with those obtained when only offering an on-premises solution which we consider the benchmark case. Finally, we wrap up our study by examining how the software vendor's security investment decisions interact with his versioning choice which we present as an extension to our model.

5 High Security-Loss Environment

We begin by studying a high security-loss environment where consumers are subject to large economic losses if struck by security attacks. Specifically, in the loss model, the parameter α specifies the magnitude of loss correlation with valuation, and in this section we examine the vendor's problem when α is large. High security-loss environments are common and often reflect the reality of current network software security: some users are patching their on-premises software installations when vulnerabilities arise in order to prevent potential security breaches while other users do not because of costs associated with patching. Enterprise software is typically classified as being in a high security-loss environment which is why many organizations employ a planned and systematic approach for deploying patches on such systems (Bloor 2003, Boulton 2013, Kash 2013). Similarly, SaaS providers of enterprise software are diligently addressing vulnerabilities to protect the interests of their customers (Branscombe 2012, Microsoft 2013). Microsoft IIS, Microsoft Dynamics, and SAP Business One are examples of enterprise software that businesses utilize and rely on for their day-to-day operations. When enterprise systems such as the above are successfully attacked and compromised, the affected businesses often incur large economic losses associated with lost sales, customer goodwill, reputation, IT human resources, and information (Lewis and Baker 2013).

First, following Lemma 1, we present a characterization of the three regions that can arise in

the consumer market equilibrium when the security loss factor α becomes high.

Corollary 1 (Equilibrium under high α) *Given on-premises and SaaS prices, $p \in (0,\, 1-c_p)$ and $p_s \in (0,\, \delta)$, respectively, and other parameters c_p, π_d, π_u, and δ, the equilibrium consumer strategy profile σ^* satisfies:*

Region I (No SaaS): If $p_s > \delta c_p$, $p \le p_s/\delta - c_p$, and $\alpha \ge \alpha_B \triangleq c_p(1-c_p)/(\pi_u(1-c_p-p))$, then $p < v_u < v_p < 1$ and σ^* is given by either (6) with $v_d = v_u$ or (7) with $v_d = v_p$;

Region II (SaaS for low tier): If $p > \max(p_s/\delta - c_p, p_s)$ and $\alpha \ge \max(\alpha_E \triangleq \delta(p_s - p\delta)^2/(\pi_u p_s^2((p+c_p)\delta - p_s))$, $\hat{\alpha}_1)$, where $\hat{\alpha}_1$ is the unique root greater than c_p/π_u that satisfies $g_1(\alpha) = 0$ where

$$g_1(\alpha) \triangleq \delta + \frac{c_p\delta\pi_d}{\pi_u} + \sqrt{\delta\left(4p_s\alpha\pi_d + \frac{\delta(c_p\pi_d + \pi_u - \alpha\pi_d\pi_u)^2}{\pi_u^2}\right)} - 2(1-c_p) - \alpha\left(\delta\pi_d + \frac{2(p-p_s)\pi_u}{c_p - \alpha\pi_u}\right), \quad (13)$$

then $p_s < v_d < v_u < v_p < 1$ and σ^* is given by (6);

Region III (SaaS for middle tier): If $p_s < \delta c_p/(1-\delta)$, $p_s/\delta - c_p < p \le p_s$, and $\alpha \ge \max(\alpha_F \triangleq c_p(1-\delta)^2(p - p_s + c_p\delta)/(\pi_u(p-p_s+c_p)^2(\delta(p+c_p) - p_s))$, $\hat{\alpha}_2)$, where $\hat{\alpha}_2$ is the unique positive root that satisfies $g_2(\alpha) = 0$ where

$$g_2(\alpha) \triangleq 1 - 2(p + c_p - p_s) + \alpha\pi_u - \frac{2\delta\pi_d\alpha(p-p_s)}{\Phi - \alpha\delta\pi_d} - \frac{\Phi\pi_u}{\delta\pi_d} - \sqrt{4p\alpha\pi_u + \left(1 - \alpha\pi_u + \frac{\Phi\pi_u}{\delta\pi_d}\right)^2}, \quad (14)$$

and $\Phi \triangleq p - (1-c_p) + (\delta - p_s)$, then $p < v_u < v_d < v_p < 1$ and σ^* is given by (7).[14]

By Region II and III of Corollary 1, when usage of SaaS arises in equilibrium, the threshold valuations satisfy either $v_p > v_u > v_d$ or $v_p > v_d > v_u$, respectively. In the latter case, the SaaS alternative is preferred by the middle tier of the consumer market. Then, by (8), the vendor's profit function in Region III can be expressed

$$\Pi(p, p_s) = p\,(1 - v_p + v_d - v_u) + p_s(v_p - v_d)\,. \quad (15)$$

[14]For each of the three regions, a complete characterization of the threshold values v_p, v_u, and v_d is provided in the Appendix.

We will subsequently refer to the prices that maximize (15), subject to the constraint that they induce a middle-tier SaaS consumer market structure, with p^M and p_s^M. The corresponding profits are denoted by $\Pi^M \triangleq \Pi(p^M, p_s^M)$. Similarly, by (10), security losses are now simplified to

$$SL = \left[\alpha(\pi_u(v_d - v_u)(v_d^2 - v_u^2) + \delta\pi_d(v_p - v_d)(v_p^2 - v_d^2))\right]/2 + c_p(1 - v_p)\,, \tag{16}$$

and social welfare can be expressed

$$\begin{aligned} W = \big[1 - v_p^2 + v_d^2 - v_u^2 + \delta(v_p^2 - v_d^2) - \alpha(\pi_u(v_d - v_u)(v_d^2 - v_u^2) + \delta\pi_d(v_p - v_d)(v_p^2 - v_d^2))\big]/2 \\ - c_p(1 - v_p)\,. \end{aligned} \tag{17}$$

On the other hand, when the SaaS alternative is preferred by the lower tier of the market, i.e., $v_p > v_u > v_d$ as in Region II of Corollary 1, the vendor's profit function can be expressed

$$\Pi(p, p_s) = p(1 - v_u) + p_s(v_u - v_d)\,. \tag{18}$$

Analogously, p^L and p_s^L will denote the prices that maximize (18), constrained such that they induce a low-tier SaaS consumer market structure, and the respective profits will be denoted by $\Pi^L \triangleq \Pi(p^L, p_s^L)$. For this structure, the security losses and welfare are given by

$$SL = \left[\alpha(\pi_u(v_p - v_u)(v_p^2 - v_u^2) + \delta\pi_d(v_u - v_d)(v_u^2 - v_d^2))\right]/2 + c_p(1 - v_p)\,, \tag{19}$$

and

$$W = \left[1 - v_u^2 + \delta(v_u^2 - v_d^2) - \alpha(\pi_u(v_p - v_u)(v_p^2 - v_u^2) + \delta\pi_d(v_u - v_d)(v_u^2 - v_d^2))\right]/2 - c_p(1 - v_p)\,, \tag{20}$$

respectively.

5.1 Versioning Strategy

Proposition 1 *For high security-loss environments*[15]*, (i) when patching costs and the SaaS alternative's quality are both high, i.e., $c_p > 1/3$ and $\delta > \frac{2(1-c_p)}{1+c_p}$, a software vendor can maximize profits by setting prices such that the SaaS offering is preferred by the middle tier of the consumer market; (ii) otherwise, the vendor sets prices so that the SaaS alternative is geared for the lower tier of the consumer market.*

In high security-loss environments, there can be substantial consumer benefits associated with a reduction in the magnitude of security losses. When the consumer population is induced to separate usage across on-premises and SaaS offerings, these losses are mitigated through diversification. Proposition 1 establishes that in these environments the vendor should version by setting prices such that all three user populations (patched on-premises users, unpatched on-premises users, and SaaS users) are represented in equilibrium. By inducing a population of SaaS users, the vendor has removed a large mass of potentially unpatched hosts from the network; because of patching costs, in the absence of a SaaS version, many of these users would not patch as on-premises users. Thus, offering SaaS can help reduce the risk faced by remaining unpatched on-premises users because, by (1) and (2), the risk these users face is proportional to the size of unpatched users in equilibrium. Although the SaaS users do not bear undirected risk comparable to on-premises users, they now face considerable directed risk existing as a large node on the network. However, pricing the SaaS version to induce them to accept some directed risk helps diversify security risk within the network.

First, we discuss part (ii) of Proposition 1. When patching costs are small and security risk is large, consumers have strong incentives to patch when using on-premises software. In this case, the vendor can both charge a high price for its on-premises software and still keep the security risk faced by unpatched on-premises users low because they remain a relatively small population within the network which limits the impact of their security externality. Because of the vendor's pricing power associated with on-premises software, he should set the price of his SaaS offering to prevent cannibalization but still serve the lower tier of the consumer market. Part (ii) of Proposition 1 is illustrated in panels (a) and (b) of Figure 1. Because $\delta = 0.80$, the condition $\delta < 2(1 - c_p)/(1 + c_p)$ is satisfied whenever $c_p < 3/7$ as indicated by the area labeled A in the figure. Within this area, the optimal prices are given by p_s^L and p^L, satisfying $p_s^L < p^L$, which give rise to a low-tier SaaS structure characterized by $v_p > v_u > v_d$ as is illustrated in panel (b) of Figure 1.

In area A of panel (a), as patching costs (c_p) increase, the price of the on-premises version (p^L) decreases, but the price of SasS version (p_s^L) first decreases and then increases. To see why, first note that an increase in patching costs reduces the patching population of the on-premises version. Moreover, because of the negative security externalities associated with unpatched behavior, overall

[15] For all proposition statements covering high security-loss environments, we formally mean there exists $\underline{\alpha} > 0$ such that for all $\alpha > \underline{\alpha}$, the proposition statement holds. We employ asymptotic analysis because the exact lower bounds do not have closed form expressions due to the complexity of the model. Similar to other papers using asymptotic analysis techniques (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, Beil and Wan 2009), our goal is to identify the regions under which each result and implication is valid - these results are robust and satisfied for wide parameter regions.

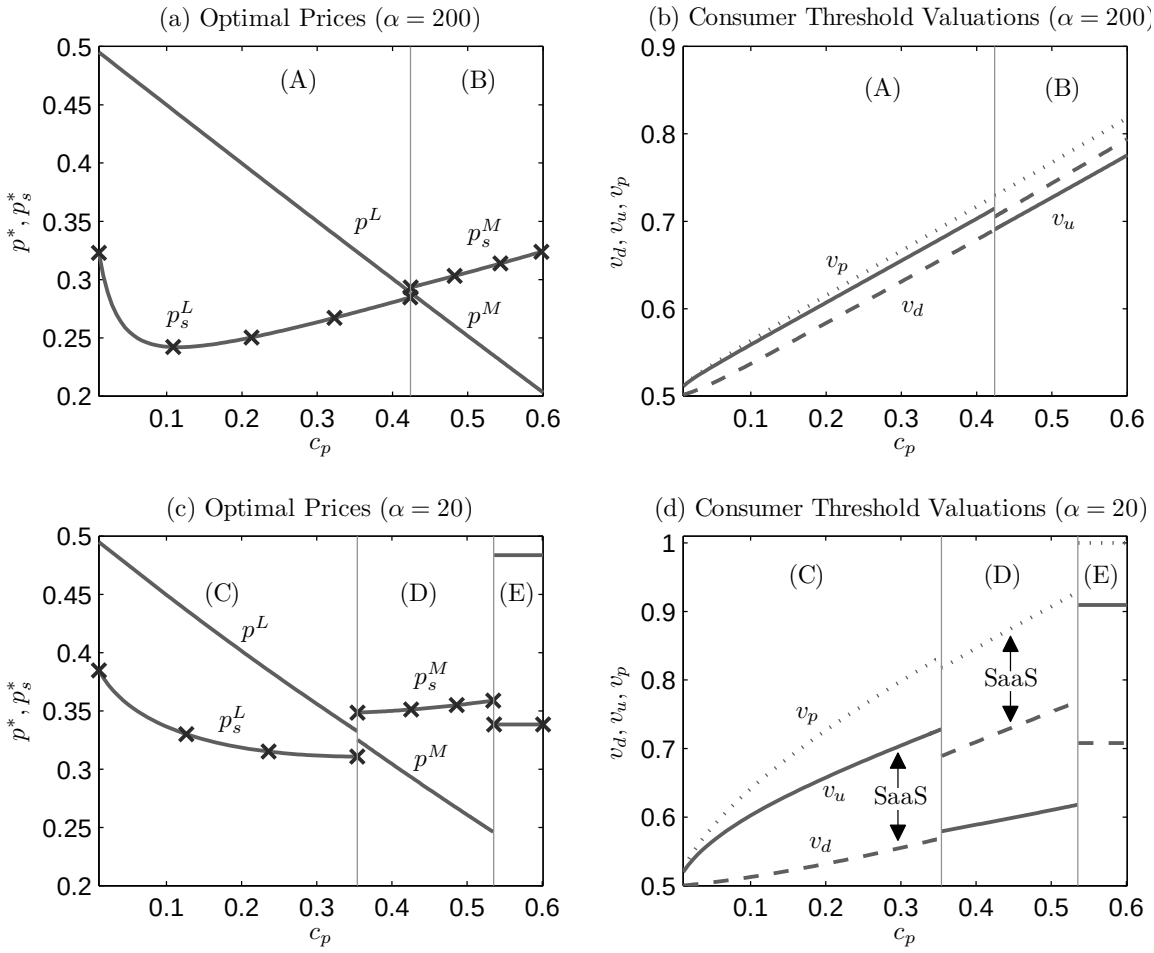

Figure 1: How patching costs affect pricing and on-premises versus SaaS usage. The other parameter values for all panels are $\delta = 0.80$, $\pi_u = 0.20$, and $\pi_d = 0.10$.

usage will also decline which reduces vendor profitability. In this case, the vendor must reduce p^L to help maintain a sizeable patching population, as well as to encourage unpatched on-premises users, who now face greater risk, to remain in the user population. However, under high security risk, the vendor also needs to throttle growth in the size of the risky populations as more users elect not to patch because of the increased patching costs. To achieve this objective, he must carefully adapt his SaaS price, p_s^L. If he lowers p_s^L, then more users at the lower end of the valuation space will become SaaS users which increases the risk associated with SaaS and, in turn, provides disincentives for members of the larger on-premises unpatched population to switch to SaaS. As a result, both of these risky populations could grow substantially. Instead, the vendor must raise p_s^L which prevents low valuation users from entering and leads to a comparatively smaller increase in the sizes of the risky populations, as in the right-hand portion of area A in Figure 1. However, when patching costs are low as in the left-hand portion of area A in panel (a) or when security risk is still high but

slightly lower as in panel (c) where $\alpha = 20$, the impact of a slightly larger unpatched, on-premises user population and SaaS user population is not as detrimental. As a result, the vendor prefers to reduce p_s^L to provide incentives for users to diversify risky usage across on-premises and SaaS versions.

Proposition 1 establishes that as patching costs become larger, there exists a point at which the vendor alters his strategy: jumping up his SaaS price from p_s^L to p_s^M, and jumping down his on-premises price from p^L to p^M. This can be seen in panel (a) of Figure 1 as a move from area A (Region II in Corollary 1) to B (Region III in Corollary 1) for a security loss factor of $\alpha = 200$. Similarly, in panel (c) of Figure 1, the same effect is shown as a shift from area C to D for a security loss factor one order smaller. At this point, due to the substantial patching costs, even though he lowers the on-premises product price, the vendor will face a larger unpatched population and reduced usage due to the negative externality these users impose. Although his SaaS product may have slightly lower base quality, when accounting for the security externalities, this may not be the case. This outcome is noteworthy because in typical product differentiation problems, the higher quality product is consumed by the higher value consumers, while the lower quality product is priced for the less quality-sensitive segment (see, e.g., Bhargava and Choudhary 2001, 2008, Johnson and Myatt 2003). Contributing to the versioning literature above, we establish a unique, inverse versioning result in the presence of two idiosyncratic security externalities; specifically, in our setting, the on-premises version is clearly assumed to be of higher quality (i.e., a type v consumer derives value v from on-premises and δv from SaaS where $\delta < 1$). However, the security risk associated with each version is endogenously determined in equilibrium, being affected by vendor pricing and strategic consumption behavior. Here, we see it is possible that the vendor will set prices to induce an outcome where the inherently lower quality SaaS version endogenously has higher effective quality. Specifically, as we cross the aforementioned boundary in patching costs, the vendor strategically prices its SaaS product at a higher level, i.e., he targets a smaller, higher-value population which is accompanied by a smaller directed risk. Because users of the SaaS option are exposed to negligible undirected risk, the vendor's pricing induces an outcome where medium valuation users will prefer the SaaS option over the lower value unpatched on-premises offering that faces considerable undirected security risk. In panel (b) of Figure 1, the area labeled by B shows how the thresholds induced by his pricing flip to $v_p > v_d > v_u$, leading to a middle-tier SaaS outcome; portion D of panel (d) has a similar nature.

One final point also illustrated in panels (c) and (d) of Figure 1 is another pricing strategy change

at the junction between areas D and E. When the magnitude of security losses is not too high (i.e., $\alpha = 20$) and patching costs increase to a larger level, the vendor has incentives to significantly increase his on-premises price to reduce the size of the purchasing on-premises population thus limiting the negative security externality to an extent that these consumers now have much reduced incentives to patch their products. Rather than continuing to cut his on-premises price to ensure a patching population exists to limit undirected security risk, a substantial price increase allows him to serve only the highest valuation market with his on-premises product. Complementing this strategic price increase is a drop in his SaaS price to capture more of the market at the lower end. However, for any level of patching costs, as the security loss factor grows high enough, area E as depicted in Figure 1 disappears due to the large losses users incur when being unpatched; this is the essence of Proposition 1.

Having developed an understanding of the conditions under which the vendor targets his SaaS product to the middle and lower tiers of the consumer market, we next study how his versioning strategy affects social welfare. We aim to highlight which of the two consumer market characterizations that arise under optimal vendor pricing, as fully described in Proposition 1, is socially preferable. Further, we also particularly identify regions in which welfare can be increased if the vendor were incentivized to induce the market outcome that goes against his preference.

Proposition 2 *For high security-loss environments, when patching costs are within an intermediate range and the SaaS alternative's quality is high, i.e.,* $\underline{c_p} < c_p < 1/3$ *and* $\delta > \underline{\eta}$*, social welfare can be increased if incentives are provided to encourage the software vendor to target his SaaS alternative to the middle tier rather than lower tier of the consumer market. However, for most other levels of patching costs and SaaS quality, the vendor-preferred outcome is also better for welfare. Technically, there exist* $\underline{c_p} > (17 - 4\sqrt{15})/7$ *and* $\underline{\eta} > \frac{2(7-14c_p+3c_p^2)}{(7-c_p)(1+c_p)}$ *such that*

(i) If $\underline{c_p} < c_p < 1/3$ *and* $\delta > \underline{\eta}$*, then* $W\big|_{p^*,p_s^*} < W\big|_{p^M,p_s^M}$*;*

(ii) If $\delta < \frac{2(7-14c_p+3c_p^2)}{(7-c_p)(1+c_p)}$*, or* $c_p > 1/3$ *and* $\delta > \frac{2(1-c_p)}{1+c_p}$*, then* $W\big|_{p^*,p_s^*} = \max\left(W^L, W^M\right)$*,*

where W^L *and* W^M *denote the welfare associated with equilibrium outcomes in Regions II and III, respectively.*[16]

Proposition 2 establishes that there exists an interval of patching costs where the vendor will prefer to induce a low-tier SaaS outcome characterized by $v_p > v_u > v_d$ (Region II of Corollary 1) with his

[16] Analytical expressions for W^L and W^M are provided in the Appendix.

pricing, whereas social welfare would be strictly higher if he priced at p^M and p_s^M to induce the middle-tier SaaS, $v_p > v_d > v_u$ (Region III of Corollary 1), consumer market outcome. The reasoning here is that when the vendor adapts his strategy to target the SaaS offering to the middle tier of the consumer market, he effectively increases the size of the patched population by dropping the on-premises price and restricts the size of the SaaS user population by increasing the SaaS price. Combining these effects, the total security losses on the network are smaller which, in aggregate, leads to higher welfare, despite the negative impact of restricted usage. Part (i) of Proposition 2 establishes that there exist intervals near the upper bound on patching costs and lower bound on the SaaS quality parameter where providing external incentives to the vendor and/or users may help encourage the socially-preferable outcome. However, part (ii) of the proposition also establishes that, in many cases, an outcome where the SaaS alternative is catered to the lower tier of the consumer market is also consistent with welfare considerations.

5.2 Comparison to Benchmark

In this section, we examine how a software vendor's decision to release SaaS versions of his traditionally on-premises software product (as detailed in Section 5.1) affects profitability, social welfare, and the security properties of the network relative to benchmark outcomes where only an on-premises offering is made. Additionally, we study the impact of introducing SaaS on consumer surplus. For convenience, we use the subscript "BM" to denote that the measure is under the *benchmark* outcome where the on-premises version is the sole offering.

Lemma 2 (Benchmark under high α) *For high security-loss environments without SaaS versioning (i.e., $\delta = 0$), under optimal pricing there always exist a positive mass of customers who prefer patching on-premises software (OP, P) and a positive mass of consumers who prefer to use on-premises software but not patch it (OP, NP). The equilibrium purchasing and patching thresholds satisfy $0 < v_u < v_p < 1$.*

Lemma 2 shows that when the quality of the SaaS offering goes to zero, or equivalently SaaS is not offered, the vendor will set the on-premises price to induce both patched and unpatched populations in equilibrium. In a high security-loss environment, the unpatched population is exposed to significant undirected security risk and shrinks to help limit the security externality. The case where $\delta = 0$ is a special case where our model converges to the model in August and Tunca (2006), hence the equilibrium pricing and market structure characterization are both consistent. This case

serves as an appropriate benchmark for comparison of economic and security measures when $\delta > 0$ and versioning occurs.[17]

First, we examine profitability and welfare comparisons with the benchmark solution. Proposition 1 establishes that for high security-loss environments, the software vendor will release both alternatives and target the SaaS version to the middle tier when patching costs (c_p) are high and the SaaS version has similar quality (i.e., high δ). In the following proposition, we demonstrate that a joint offering strategy increases both profits and welfare substantially. Further, we characterize how c_p, δ, and the likelihood of a directed attack on the SaaS offering (π_d) affect the extent to which the outcome of the joint offering improves these measures.

Proposition 3 *For high security-loss environments, both vendor profits and social welfare can increase substantially under a joint offering strategy. Both relative measures of improvement are increasing in patching costs and the quality of the SaaS version, but decreasing in the likelihood of directed attacks. Technically, there exists* $\underline{\omega}, \kappa > 0$ *such that for all* $\alpha > \underline{\omega}$,

$$\left| \frac{\Pi^* - \Pi_{BM}}{\Pi_{BM}} - \frac{c_p \delta}{\pi_d \alpha (1 - c_p)^2} \right| < \frac{\kappa}{\alpha^2} \tag{21}$$

and

$$\left| \frac{W^* - W_{BM}}{W_{BM}} - \frac{2 c_p \delta}{3 \pi_d \alpha (1 - c_p)^2} \right| < \frac{\kappa}{\alpha^2} \tag{22}$$

are satisfied.

Proposition 3 establishes that the introduction of a SaaS offering can result in substantial percentage increases in profits and social welfare. Examining the inequalities in (21) and (22), it is straightforward to see that both normalized measures decrease in π_d but increase in c_p and δ. A decrease in π_d corresponds to reduced directed security risk for consumers who use the SaaS alternative in equilibrium. In a similar vein, an increase in δ also reflects a higher quality SaaS offering which is beneficial to both vendor profitability and welfare. On the other hand, for c_p, the potential improvement associated with a SaaS release stems from consumers' patching behavior of the on-premises solution. In particular, as patching costs increase, consumers find it incentive compatible to bear more undirected security risk rather than incurring these patching costs. Given the negative externalities unpatched users can inflict on the network, under these circumstances,

[17] The expressions for the benchmark equilibrium outcome, including the thresholds (v_u and v_p) and the price, as well as the associated measures including profit, social welfare, consumer surplus and security losses per user are provided in the proof of Lemma 2 in the Appendix.

introducing the SaaS alternative can have an even stronger effect by inducing consumers to split usage across alternatives and diversify this security risk to include more directed risk (and less undirected risk).

Next, focusing our attention on high security-loss environments which have the greatest potential for improvement, we examine how introducing a SaaS alternative affects the security properties of the network as well as consumer surplus. By (10), we can define the average per-user security losses as

$$\hat{SL} \triangleq \frac{SL}{u(\sigma^*) + d(\sigma^*) + n(\sigma^*)}\,, \tag{23}$$

which simplifies to either $SL/(1-v_d)$ or $SL/(1-v_u)$ in Regions II and III of Corollary 1, respectively.

Proposition 4 *When a SaaS version is introduced in high security-loss environments:*

(i) The average security losses per user decrease under high patching costs, i.e. $c_p \geq \delta/(4-\delta)$*, but actually increase otherwise;*

(ii) Despite the substantial increase in welfare stemming from a SaaS release, when patching costs are low, i.e., $c_p \leq 1/3$*, and the SaaS offering quality parameter satisfies* $\delta > 2 - \frac{64c_p^2\pi_d(1-c_p(4-c_p))}{\pi_u(1+c_p)^4}$*, consumer surplus decreases in equilibrium.*

Part (i) of Proposition 4 brings forth an important insight: a vendor's diversification of software usage by offering both on-premises and SaaS versions can actually increase per-user security losses. One would expect that introducing a SaaS alternative would split the undirected risk being faced in the benchmark case into two smaller risks (undirected and directed) as a portion of the consumers adopt the SaaS alternative instead. However, part (i) of Proposition 4 establishes that a software vendor may influence usage and patching behavior through pricing in such a way that the average security losses per user is higher in the joint offering.

In high security-loss environments, when SaaS is introduced, some consumers who would have elected to buy the on-premises product and remain unpatched, i.e., (OP, NP), in the benchmark case now have incentives to switch to SaaS usage, i.e., $(SaaS, ND)$. Because this reduces the size of the unpatched population, consumers who were buying the on-premises product and patching, i.e., (OP, P), are now no longer facing as large a negative externality. Therefore, they have overall reduced incentives to patch, and some of these consumers will now elect to remain unpatched instead. Also, because introduction of SaaS splits risk into undirected and directed types, some consumers who had opted out in the benchmark case will now become users. Thus, in comparison

to the benchmark case, when both on-premises and SaaS versions are offered, overall usage increases while overall patching decreases.

Part (i) of Proposition 4 establishes that when patching costs are small, the aforementioned cumulative effect of increased usage and decreased patching associated with the introduction of SaaS results in higher average per-user security losses. The reason is that when patching costs are small, the consumer market structure is already characterized by a large patching population in the benchmark case. The population of unpatched on-premises users is in contrast relatively small. Hence, when SaaS is introduced, although patching slightly decreases, the proportional increase in either unpatched or SaaS usage is substantial. A relatively large increase in these two types of usage which are exposed to undirected and directed security risk, respectively, can lead to higher average security losses because of the accompanying negative externalities. On the other hand, when patching costs are large, the benchmark case is characterized by a small patching population and large unpatched population. In this case, aggregate SaaS and unpatched on-premises usage still increases while patching decreases. However, the reduction in patching behavior has a relatively minor negative effect on an already substantial unpatched population. In contrast to the case above, the diversification benefits of splitting security risk into undirected and directed types now outweigh the minor increase in the externality. Thus, for large patching costs, per-user security losses decrease when SaaS is made available.

One might expect that consumer surplus would increase when a software vendor offers a menu of differentiated products with idiosyncratic security risks, but that is not always the case as we establish in part (ii) of Proposition 4. Because, surprisingly, average per-user security losses can increase when a software vendor introduces a SaaS version of his on-premises product, from a consumer perspective such a release may not necessarily be beneficial. Part (ii) of Proposition 4 suggests that for software that has relatively lower patching costs (such as client applications), a vendor will release a SaaS version not for the benefits of reduced security risk but rather to expand his market at the lower end and price discriminate. The net effect of his joint offerings on consumer surplus is negative, which is partly driven by the increase in security losses formalized in part (i) of Proposition 4.

6 Low Security-Loss Environment

For additional insight into the overall security landscape, we next examine environments where consumers are subject to smaller economic losses associated with security attacks. In this case, we focus on a class where α is small and study software applications belonging to this class such as client applications that are less mission critical to business operations. Some examples include anti-virus client software, media players, document readers, and perhaps even productivity software such as Microsoft Office 365. In the following, we take a similar approach to Section 5 by further characterizing the consumer market equilibrium, which becomes simplified when considering only a low security-loss environment. Subsequently, we examine the vendor's versioning decision and then compare profitability, social welfare, security losses and consumer surplus to the benchmark measures for this case.

As the security loss factor α becomes small, users of on-premises software will find it better to assume undirected security risk than to incur patching costs. Thus, a patching population will not arise in equilibrium as is formalized in the following corollary.

Corollary 2 (Equilibrium under low α) *Given on-premises and SaaS prices, $p \in (0,\, 1)$ and $p_s \in (0,\, \delta)$, respectively, and other parameters c_p, π_d, π_u, and δ, the equilibrium consumer strategy profile σ^* satisfies:*

Region I (No SaaS): If $p \leq p_s/\delta$ and $\alpha \leq \min(\alpha_B,\, \alpha_A \triangleq \delta(p_s - p\delta)/(p_s\pi_u(\delta - p_s)),\, \alpha_C \triangleq (1 - p + p_s - \delta)(p - p_s + \delta)/(\pi_u(\delta - p_s))$, then $p < v_u < 1$ and σ^* is given by either (6) with $v_d = v_u$ and $v_p = 1$ or (7) with $v_d = v_p = 1$;

Region II (SaaS for low tier): If $p_s/\delta < p \leq 1 + p_s - \delta$ and $\alpha \leq \hat{\alpha}_1$, then $p_s < v_d < v_u < 1$ and σ^* is given by (6) with $v_p = 1$;

Region III (SaaS only): If $1 + p_s - \delta < p$ and $\alpha \leq \alpha_D \triangleq (1 - p + p_s)(p - 1 - p_s + \delta)/(\delta\pi_d(1 - p))$, then $p_s < v_d < 1$ and σ^* is given by either (6) with $v_u = v_p = 1$ or (7) with $v_u = v_d$ and $v_p = 1$.

As can be seen in Corollary 2, in equilibrium, there are three possibilities for the consumer market structure: the on-premises version is preferred by all users and not patched (Region I); higher valuation users prefer on-premises and do not patch while lower valuation users prefer SaaS (Region II); and finally all users prefer SaaS (Region III). Given this consumer market outcome, we next analyze the vendor's versioning problem.

6.1 Versioning Strategy

As presented in Proposition 1, in high security-loss environments, the vendor has strong incentives to release a SaaS offering to change the structure of the network and reduce security risk by splitting the user population; this helps to limit both directed and undirected security attacks which are influenced by total SaaS user and unpatched on-premises population sizes, respectively. With high security-loss environments, the margin for improvement is large. However, as the security loss factor becomes small, the benefit of risk diversification becomes more limited. An open question in the literature is whether versioning makes sense in the presence of security externalities as their impact becomes small.

Proposition 5 *For low security-loss environments*[18]*, a software vendor still prefers to offer both on-premises and SaaS versions of his software.*

In contrast to Proposition 1, one might expect that as the security risk associated with software becomes small a software vendor would shift toward a strategy where he prices his offerings in such a way that only the higher quality offering is consumed. This outcome would be consistent with the literature on versioning of information goods. Specifically, in the absence of unit costs and when consumer valuations (or quality sensitivities) are uniformly distributed, a monopolist with two different quality substitutes will price them such that only the high quality substitute is consumed in equilibrium. In other words, in such a case, he will not version his product. Because versioning is frequently observed with information goods, the literature on versioning of information goods has worked to reconcile this inconsistency.

Proposition 5 examines this versioning issue from a security perspective. Specifically, in the absence of the negative security externalities present in our model, because consumers have uniform valuations and the SaaS offering has a quality reduction factor, consistent with the versioning literature the vendor would optimally offer only the higher quality information good. However, when consumers of both versions are exposed to negative security externalities in the form of directed risk for the SaaS offering and undirected risk for the on-premises offering, Proposition 5 establishes that it is still optimal for the vendor to offer both versions even in low security-loss environments. In fact, as α diminishes, we analytically demonstrate that it is always profitable to still introduce the SaaS offering. In our case, the fact that each product is exposed to a unique

[18] For all proposition statements covering low security-loss environments, we formally mean there exists $\bar{\alpha} > 0$ such that for all $\alpha < \bar{\alpha}$, the proposition statement holds.

externality (directed or undirected risk) creates the separation necessary for versioning to become optimal. From a practical standpoint, this result provides another alternative explanation for the commonplace existence of multiple versions of software products: as long as the versions have some idiosyncratic risk stemming from their respective user populations, however small, then it is profit-maximizing to set prices such that both versions are consumed in equilibrium.

6.2 Comparison to Benchmark

Analogous to Section 5.2, next we examine how a software vendor's optimal decision to release a SaaS version of his product (as established in Proposition 5) affects profitability, social welfare, security losses, and consumer surplus in comparison to a benchmark. As before, for an appropriate benchmark, we use measures computed under the equilibrium solution to the case where $\delta = 0$ which is to say that the on-premises software is the sole offering. These benchmark measures then coincide with those computed in August and Tunca (2006).

Lemma 3 (Benchmark under low α) *For low security-loss environments without SaaS versioning (i.e., $\delta = 0$), under optimal pricing there is only a positive mass of customers who prefer not to patch on-premises software (OP, NP) in equilibrium. The equilibrium purchasing and patching thresholds satisfy $0 < v_u < v_p = 1$.*

The benchmark equilibrium characterized in Lemma 3 is similar to the outcome that unfolds from Corollary 2 and Proposition 5 in that no consumer will find it optimal to patch her on-premises software in equilibrium. Thus, all users face some degree of undirected security risk even though it is low, while under the versioning outcome in Proposition 5, risk is diversified as users separate into on-premises and SaaS user populations. Next, we turn our attention to comparing these two outcomes.[19]

For low security-loss environments, we characterize the relative benefit of introducing SaaS in the following proposition.

Proposition 6 *For low security-loss environments, introduction of a SaaS version will provide a limited increase in vendor profits and social welfare. Both relative measures of improvement are increasing in the quality of the SaaS version and the likelihood of undirected attacks. Technically,*

[19]The expressions for the benchmark equilibrium outcome, including the threshold (v_u) and the price, as well as the associated measures including profit, social welfare, consumer surplus and security losses per user are provided in the proof of Lemma 3 in the Appendix.

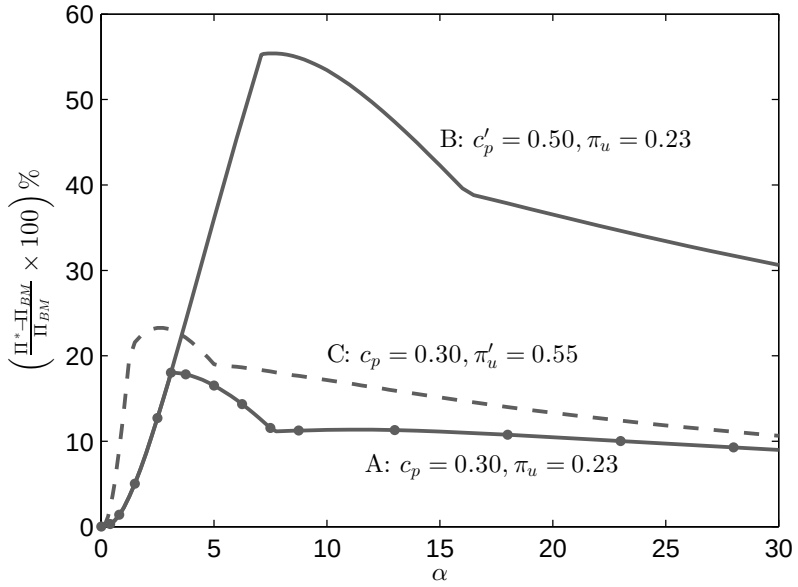

Figure 2: Percentage increase in vendor profitability when a SaaS version is offered in addition to an on-premises version of software. This percentage is plotted over a wide range of security loss environments. Values for patching costs and undirected attack probabilities are listed on the plot. The other parameter values are $\delta = 0.80$ and $\pi_d = 0.10$.

there exists $\overline{\omega}, \eta > 0$ *such that for all* $\alpha < \overline{\omega}$*,*

$$\left| \frac{\Pi^* - \Pi_{BM}}{\Pi_{BM}} - \frac{\delta \pi_u^2 \alpha^2}{16(1-\delta)} \right| < \eta \alpha^3 \tag{24}$$

and

$$\left| \frac{W^* - W_{BM}}{W_{BM}} - \frac{5\delta \pi_u^2 \alpha^2}{48(1-\delta)} \right| < \eta \alpha^3 \tag{25}$$

are satisfied.

Although Proposition 5 demonstrates that a vendor should optimally release both on-premises and SaaS versions of his product in low security-risk environments, Proposition 6 suggests that the benefits stemming from diversification are much more limited in these environments. In contrast to Proposition 3, by comparing (21) and (24), the percentage increase in profits associated with releasing the SaaS alternative is an order of magnitude smaller. In this sense, security concerns alone may not solely justify the additional costs of managing two, separate versions of the software.

The findings in Propositions 3 and 6 are illustrated in Figure 2. We depict three curves plotting the measure $\frac{\Pi^* - \Pi_{BM}}{\Pi_{BM}}$ computed numerically under parameter sets A: $c_p = 0.30, \pi_u = 0.23$, B: $c_p' = 0.50, \pi_u = 0.23$, and C: $c_p = 0.30, \pi_u' = 0.55$, respectively. As can be seen, near $\alpha = 30$, the percentage improvement in profits ranges from approximately 10-30% under these parameter sets;

however, near $\alpha = 1/30$, the percentage improvement is negligible. This is the essence of the two propositions - that diversification of security risk by offering SaaS has much greater potential for moderate to high security-loss environments. Comparing curve B to A, one can see that an increase in patching costs from $c_p = 0.30$ to $c_p' = 0.50$ can change the potential profit improvement of a SaaS release strategy substantially because of the poor patching behavior induced on the network at this higher cost level.[20] This characteristic is consistent with the results presented in Proposition 3, in particular from (21). Similarly, for lower α, Proposition 6 and specifically (24) suggests that an increase in the likelihood of an undirected attack (π_u) will also increase the potential benefit of a diversification strategy. In Figure 2, we can see this effect by comparing curve C to A within the lower range of α.

Next, we turn our attention to security losses and consumer surplus. In low security-loss environments, the effect of versioning on these two measures differs considerably from what we established for high security-loss environments.

Proposition 7 *When a SaaS version is introduced in low security-loss environments, the average security losses per user decrease and consumer surplus increases in equilibrium.*

For high security-loss environments, in Proposition 4, we established that when patching costs are low, the average security losses per user actually increase when the vendor versions by introducing a SaaS solution. When potential security losses become limited, Proposition 7 formally establishes that a similar type of outcome cannot happen; that is, by offering a SaaS version in addition to an on-premises version, the vendor is able to effectively diversify the undirected risk under the benchmark case into two smaller risks of undirected and directed variety, which reduces the average security losses. Because the potential security losses are inherently limited, the negative impact of a reduction in the patching population is smaller than the positive diversification effect. Moreover, the introduction of SaaS benefits consumers in terms of both security risk considerations and differentiated pricing.

7 Extension: Security Investment

In this section, we study a setting where the software vendor can invest to increase the security of his SaaS and on-premises offerings. We assume the firm can invest effort levels ϵ_u, $\epsilon_d \in [0, 1)$

[20] When the security loss factor (α) is within a lower range, curves A and B coincide because the equilibrium consumer market structure under optimal pricing dictates that users of the on-premises product are not patching, preferring to bear the low security risk.

to reduce the security risks associated with the on-premises and SaaS versions, respectively. An effort investment of ϵ_u yields a risk reduction from π_u to $(1-\epsilon_u)\pi_u$, and similarly ϵ_d being exerted reduces π_d to $(1-\epsilon_d)\pi_d$. Because ϵ_u reduces the likelihood a vulnerability arises in the on-premises product, it also will reduce the expected patching cost from an initial value of c_p to $(1-\epsilon_u)c_p$. The respective costs associated with effort investments to improve security are denoted $C_u(\epsilon_u)$ and $C_d(\epsilon_d)$, where both cost functions are twice-differentiable, convex, increasing, and satisfy $C_u(0) = C_d(0) = C'_u(0) = C'_d(0) = 0$. For technical reasons, we assume there exists a constant $\tau > 0$ such that $C''_u(\cdot), C''_d(\cdot) > \tau$.

As we have seen thus far, versioning is an effective way to achieve risk diversification, and hence pricing SaaS and on-premises versions can also be used as a tool by the software vendor to influence consumption decisions toward profitable consumer market structures. In this extension, we aim to understand how the vendor's additional ability to invest in the security of both products influences outcomes. In particular, we examine the interaction between his versioning decision and security investments. In the following proposition, we first explore how the vendor's security investments differ across high and low security-loss environments.

Proposition 8 *In low security-loss environments, the firm increases its security-improving investments for both on-premises and SaaS versions as risk increases in the market, i.e., $\epsilon_u^*(\alpha)$ and $\epsilon_d^*(\alpha)$ are increasing in α. However, in high security-loss environments, if the likelihood of an undirected attack relative to a directed attack, patching costs, and SaaS quality are all sufficiently high, then the vendor increases security-improving efforts for the on-premises version and decreases efforts for the SaaS version as risk increases in the market. Technically, there exists $\hat{r} > 0$ such that if $\pi_u/\pi_d > \hat{r}$, $c_p > 1/2, \tau > 1/2$, and $\delta > \bar{\delta} \triangleq 2(1-c_p)/(1+c_p-2c_p^2)$ are satisfied, then $\epsilon_u^*(\alpha)$ increases in α whereas $\epsilon_d^*(\alpha)$ decreases in α.*

Figure 3 illustrates the results in Proposition 8 and the underlying intuition. In low security-loss environments, consumers tend to prefer to bear a limited, undirected security risk rather than incurring the costs of patching. Because of this reason, the vendor does not need to increase security investment in the on-premises product with a goal of reducing the expected patching costs. However, he may invest effort in the on-premises product in order to slightly reduce the risk consumers face in equilibrium. His incentives to invest into SaaS security are similar, and both equilibrium effort levels are illustrated in panel (b) of Figure 3 in the area labeled A. As α increases, the vendor increases both of its investment levels to throttle equilibrium risk which can increase

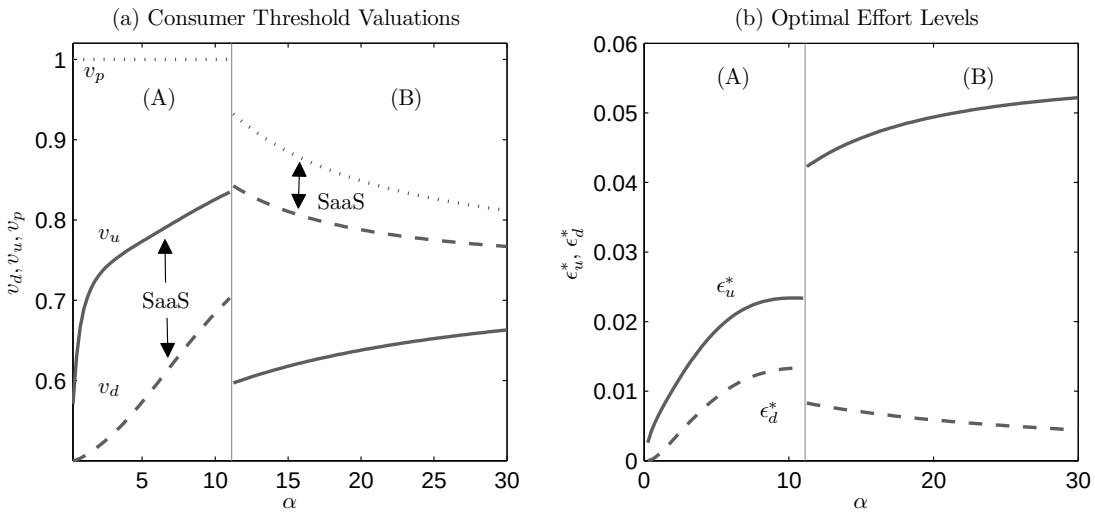

Figure 3: The impact of varying security-loss environments on security investment and the consumer market. The parameter values are $\pi_u = 0.2$, $\pi_d = 0.3$, $c_p = 0.5$, and $\delta = 0.9$. For this numerical illustration, we use cost functions $C_u(\cdot) = C_d(\cdot) = C(\epsilon) = \left(\frac{1}{1-\epsilon} - 1\right)^2$ which satisfy all technical conditions.

quickly because no one patches in equilibrium as is illustrated in panel (a) of Figure 3 in area A.

In high security-loss environments, the vendor's investment behavior differs substantially from that described above, and remarkably his security investments in on-premises and SaaS versions diverge. Under potentially high security losses, consumers have much stronger incentives to patch and protect themselves if using the on-premises product. In fact, the vendor also wants to reduce the expected patching costs by increasing his effort to improve on-premises security. As we saw earlier in Proposition 1, under high patching costs and high SaaS quality, the vendor pursues a strategy where SaaS is targeted to the middle tier of the consumer market. An essential element of this strategy is that the vendor must limit equilibrium directed security risk such that the SaaS version is consumed by this middle tier segment. The vendor can accomplish this in two ways: investing in security of the SaaS product directly, or limiting the SaaS population to indirectly achieve greater security. As α increases, because of his strong incentives to invest in the on-premises product to reduce patching costs, high type consumers shift toward patched, on-premises usage away from SaaS. Because the SaaS population becomes more limited and achieves greater security as a result, the vendor can decrease his investment in security of the SaaS product to reduce his costs. His investment behavior is illustrated in area B of panel (b). Area B of panel (a) shows how the patching population increases while the SaaS population shrinks. Connected to this discussion is the following result.

Proposition 9 *In high security-loss environments, when baseline consumer patching costs and SaaS quality are high, and the firm's security-improving costs are sufficiently convex, the vendor invests greater effort in addressing on-premises security than SaaS, i.e., ϵ_u^* is greater than ϵ_d^*, while also targeting SaaS to the middle tier of the consumer market in equilibrium.*

Proposition 9 formalizes that the software vendor may continue to target his lower inherent quality SaaS product to the middle tier even when he can invest to improve security instead. Importantly, his behavior in this case hinges on the convexity of his investment costs not being too low. In the alternative case, the vendor would have incentives to make his on-premises product extremely secure which would lead him to pursue a strategy where SaaS is geared to the lower tier of the market.

8 Concluding Remarks

In this paper, we explore how a vendor's offering of on-premises and SaaS versions of application software affects users' consumption incentives. In particular, we analytically demonstrate how users segment across products to manage the endogenously determined security externalities associated with unpatched on-premises behavior (undirected risk) and SaaS usage (directed risk). Using our characterization of equilibrium consumer behavior, we rigorously study the software vendor's versioning problem and reveal several interesting insights. First, the vendor sometimes possesses incentives to market its lower inherent quality SaaS version to a higher valuation consumer segment than it targets its higher inherent quality on-premises version. In this case, the vendor strategically sets a high SaaS price to reduce its usage and the associated level of directed security risk such that he offers a more secure product, albeit with less features, to a more quality-sensitive consumer segment. Panel (a) of Table 1 illustrates that this result obtains in high security-loss environments when patching costs are high and the inherent quality of the SaaS version is not too low, i.e., $\Pi^M > \Pi^L$ in the cell corresponding to High c_p/High δ. We also analytically establish that when patching costs are within an intermediate range and the inherent quality of the SaaS version is still reasonably close to its on-premises counterpart that the vendor will prefer to target its SaaS product to the low tier when it is advantageous to social welfare if he would instead target it to the middle tier; provision of additional incentives to the vendor can lead to welfare-superior outcomes in this region. This result can be seen in the cell in panel (a) of Table 1 corresponding to Medium c_p/High δ.

(a)

	High Security-Loss Environment		Low Security-Loss Environment
	Low δ	High δ	All δ
Low c_p	$\Pi^L > \Pi^M$ $W^L > W^M$	$\Pi^L > \Pi^M$ $W^L > W^M$	$\Pi^L > \Pi^H$ $W^L > W^H$
Medium c_p		$\Pi^L > \Pi^M$ $W^M > W^L$	
High c_p		$\Pi^M > \Pi^L$ $W^M > W^L$	

(b)

Low c_p	$\hat{SL} \uparrow\downarrow, CS \uparrow$	$\hat{SL} \uparrow, CS \downarrow$	$\hat{SL} \downarrow, CS \uparrow$
High c_p	$\hat{SL} \downarrow, CS \uparrow$		

Table 1: Summary of results on how SaaS versions are targeted to consumer segments and how measures of average per-user security losses and consumer surplus compare to benchmarks. For the high security-loss environment columns, panel (a) summarizes profit and social welfare comparisons dependent on whether the SaaS version is priced to serve the low tier (superscript L) or the middle tier (superscript M). For the low security-loss environment column, because there are only two consumer segments in equilibrium, the comparisons involve the low tier and the high tier (superscript H). Panel (b) indicates whether average per-user security losses and consumer surplus increase or decrease under SaaS versioning.

In the versioning literature, for uniformly distributed consumer types and zero marginal costs, the standard result is that a software vendor will find it optimal to only offer his higher quality product to consumers. In our study, we formally demonstrate that because of risk diversification benefits, the vendor will always offer both versions of his product as long as the risks associated with each version are idiosyncratic. Interestingly, this versioning result holds even as the security risk becomes negligible. In panel (a) of Table 1, we indicate that for low security-loss environments, the software vendor always gears his SaaS version to the lower tier of the market, pricing his higher quality on-premises version to serve the higher tier. Because a patching population does not exist in equilibrium, we use Π^H and W^H to refer to measures of profit and welfare if SaaS were targeted to the higher tier instead, which is shown to be a dominated strategy.

We compare economic measures (profitability, social welfare, security losses, and consumer surplus) under a versioning strategy against analogous measures in a benchmark case where on-premises is solely offered to consumers. We demonstrate that the potential improvement to profits

and social welfare associated with a SaaS release and its corresponding security risk diversification are substantial in high security-loss environments, but more limited in low security-loss ones. In spirit, an introduction of a SaaS version which has a different exposure to directed and undirected attacks than the on-premises version can help reduce the overall security risk on the network because of these diversification benefits. As can be seen in panel (b) of Table 1, for low security-loss environments and for high security-loss environments when patching costs are high, average per user security losses indeed decline as a result of versioning. However, our study also highlights that because of opportunistic pricing by the vendor, the security risk diversification benefits can sometimes be outweighed by market expansion leading to higher average per user security losses. This result can be seen in the cell corresponding to Low c_p/High δ. In this case, introduction of a SaaS version can even lead to depressed consumer surplus as is indicated in the table. For high security-loss environments, when the inherent quality of the SaaS offering and patching costs are low, with SaaS versioning average per-user security losses can go in either direction as is established in Proposition 4.

In recent years, companies have invested millions of dollars to provide SaaS versions of on-premises application software. Our study focuses on the security benefits of SaaS offerings in terms of risk-mitigated versioning. Given our research goals, we abstract away from modeling concerns that are outside of our paper's scope. For example, in order to implement SaaS alternatives in addition to their traditional on-premises offerings, software vendors would necessarily need to incur additional costs in practice. These costs would have a fairly large fixed-variable cost ratio. One potential future research trajectory is to develop a better understanding of why we observe the existence of vendors who do not offer both SaaS and on-premises versions (e.g., Salesforce.com). By understanding the vendor's incentives outside of security concerns that underlie this type of behavior, we could also explore what leads to outcomes in a single market where competing firms each offer one version, SaaS or on-premises, and choose to differentiate themselves by specializing.

The benefits of cloud computing and, particularly, SaaS applications are driving businesses and governments to migrate many internally supported systems and software to the cloud. However, such a paradigm shift can have major consequences on security risks as consumers make choices on software deployment and protection. We hope that the model and insights presented in this paper provide a stepping stone toward a broader understanding of how security risk can be managed as cloud computing matures and becomes an integral part of IT strategies.

Acknowledgments

The authors thank Rahul Telang (the senior editor), the associate editor, and the anonymous reviewers for their helpful suggestions throughout the review process. The authors also thank Sridhar Narasimhan, D. J. Wu, Subodha Kumar, Ali Tafti, Ramnath Chellappa, the participants at the Conference on Information Systems and Technology 2012, INFORMS 2012 Annual Meeting, the Workshop on the Economics of Information Security 2013, as well as the participants in the research seminar at University of California Irvine for their helpful comments and discussions. This material is based upon work partially supported by the National Science Foundation under Grant No. CNS-0954234.

References

Anderson, R. (2001). Why information security is hard – an economic perspective. In *Proc. of the 17th Annual Computer Security Applications Conf.*, pp. 358–365. IEEE Computer Soc.

Anderson, R. and T. Moore (2006). The economics of information security. *Science 314*, 610–613.

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

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

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

August, T. and T. I. Tunca (2011). Who should be responsible for software security? A comparative analysis of liability policies in network environments. *Management Science 57*(5), 934–959.

Bain, C., D. B. Faatz, A. Fayad, and D. Williams (2002). Diversity as a defense strategy in information systems. Does evidence from previous events support such an approach? In *Proceedings of the IFIP TC11/WG11.5 Fourth Working Conference on Integrity, Internal Control and Security in Information Systems: Connecting Governance and Technology*, Deventer, The Netherlands, pp. 77–94. Kluwer, B.V.

Beil, D. and Z. Wan (2009). RFQ auctions with supplier qualification screening. *Operations Research 57*, 934–949.

Bhargava, H. K. and V. Choudhary (2001). Information goods and vertical differentiation. *Journal of Management Information Systems 18*(2), 89–106.

Bhargava, H. K. and V. Choudhary (2008). Research note: When is versioning optimal for information goods? *Management Science 54*(5), 1029–1035.

Biddick, M. (2010, Jan). Why you need a SaaS strategy. *InformationWeek*.

Bloor, B. (2003). The patch problem: It's costing your business real dollars. *Baroudi Bloor*.

Boulton, C. (2013, Apr). American Airlines outage likely caused by software quality issues. *The Wall Street Journal*.

Branscombe, M. (2012, Sep). Salesforce talks security: From passwords to animatronic ponies. *ZDNet*.

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

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

Charney, S. (2012, Feb). Trustworthy Computing Next. Microsoft, white paper.

Chellappa, R. K. and J. Jia (2011). Competition and versioning. Working Paper, Emory University and Hong Kong University of Science and Technology.

Chellappa, R. K. and A. Mehra (2013). Versioning 2.0: A product line and pricing model for information goods under usage constraints and with R&D costs. Working Paper, Emory University and Indian School of Business.

Chen, P., G. Kataria, and R. Krishnan (2011). Correlated failures, diversification, and information security risk management. *MIS Quarterly 35*(2), 397–422.

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

Choudhary, V. (2007). Comparison of software quality under perpetual licensing and software as a service. *Journal of Management Information Systems 24*(2), 141–165.

Chow, R., P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina (2009). Controlling data in the cloud: Outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, New York, NY, USA, pp. 85–90. ACM.

Claburn, T. (2009, Sep). Government embraces cloud computing, launches app store. *InformationWeek*.

Cox, B., D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser (2006). N-variant systems: a secretless framework for security through diversity. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA. USENIX Association.

Dey, D., A. Lahiri, and G. Zhang (2012). Hacker behavior, network effects, and the security software market. *Journal of Management Information Systems 29*(2), 77–108.

El Akkad, O. (2011, Jun). Microsoft wants you to rent an Office in the cloud. *The Globe and Mail*.

Farber, D. (2007, Sep). SAP's challenge to NetSuite, Workday and salesforce.com. *ZDNet*.

Forrest, S., A. Somayaji, and D. Ackley (1997). Building diverse computer systems. In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, HOTOS '97, Washington, DC, USA, pp. 67–72. IEEE Computer Society.

Gherbi, A., R. Charpentier, and M. Couture (2011, Sep/Oct). Software diversity for future systems security. *CrossTalk*.

Gordon, L. A. and M. P. Loeb (2002, Nov). The economics of information security investment. *ACM Trans. Inf. Syst. Secur. 5*, 438–457.

Greenemeier, L. and J. N. Hoover (2007, Feb). How does the hacker economy work? *InformationWeek*.

Grossklags, J., N. Christin, and J. Chuang (2008). Secure or insure?: A game-theoretic analysis of information security games. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, New York, NY, USA, pp. 209–218. ACM.

Heal, G. and H. Kunreuther (2007). Modeling interdependent risks. *Risk Analysis 27*(3), 621–634.

Huang, K. and A. Sundararajan (2005, Nov). Pricing models for on-demand computing. Working paper, National University of Singapore and New York University.

Hui, K. L., W. Hui, and T. Yue (2013). Information security outsourcing with system interdependency and mandatory security requirement. *Journal of Management Information Systems 29*(3), 117–156.

IBM (2008). IBM internet security systems X-Force 2008 mid-year trend statistics. *IBM Global Technology Services*.

Jing, B. (2007). Network externalities and market segmentation in a monopoly. *Economics Letters 95*, 7–13.

Johnson, J. P. and D. P. Myatt (2003, Jun). Multiproduct quality competition: Fighting brands and product line pruning. *The American Economic Review 93*(3), 748–774.

Johnson, M. E. (2008). *Managing Information Risk and the Economics of Security* (1st ed.). Springer Publishing Company, Incorporated.

Jones, R. and H. Mendelson (2011). Information goods vs. industrial goods: Cost structure and competition. *Management Science 57*(1), 164–176.

Judge, P. (2002, Jul). Microsoft security push cost $100m for .Net server alone. *ZDNet*.

Kannan, K., M. S. Rahman, and M. Tawarmalani (2013, Jan). Economic and policy implications of restricted patch distribution. Working paper, Purdue University and University of Calgary.

Kannan, K. and R. Telang (2005). Market for software vulnerabilities? Think again. *Management Science 51*(5), 726–740.

Kash, W. (2013, Sep). Software patches eat government IT's lunch. *InformationWeek*.

Keizer, G. (2004, May). Sasser worm impacted businesses around the world. *TechWeb News*.

Keizer, G. (2008, Dec). Windows users indifferent to Microsoft patch alarm, says researcher. *Computerworld*.

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

Kim, B. C., P. Chen, and T. Mukhopadhyay (2011). The effect of liability and patch release on software security: The monopoly case. *Production and Operations Management 20*(4), 603–617.

Kundra, V. (2011, Aug). Tight budget? Look to the 'cloud'. *The New York Times*.

Kunreuther, H. and G. Heal (2003). Interdependent security. *Journal of Risk and Uncertainty 26*(2-3), 231–249.

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

Lahiri, A. (2012). Revisiting the incentive to tolerate illegal distribution of software products. *Decision Support Systems 53*(2), 357–367.

Lala, J. and F. Schneider (2009). IT monoculture security risks and defenses. *IEEE Security & Privacy 7*(1), 12–13.

Lee, C. H., X. Geng, and S. Raghunathan (2013). Contracting information security in the presence of double moral hazard. *Information Systems Research 24*(2), 295–311.

Lee, M. (2014, Jan). Microsoft closes Office 365 admin access vulnerability. *ZDNet*.

Lemos, R. (2003, Feb). 'Slammer' attacks may become way of life for net. *CNET News.com*.

Lemos, R. (2004, Apr). MSBlast epidemic far larger than believed. *CNET News.com*.

Lewis, J. A. and S. Baker (2013, Jul). The economic impact of cybercrime and cyber espionage. Center for Strategic and International Studies, Washington, DC.

Li, L., R. D. McKelvey, and T. Page (1987). Optimal research for Cournot oligopolists. *Journal of Economic Theory 42*, 140–166.

Liran, N. (2013, Dec). Severe Office 365 token disclosure vulnerability - research and analysis. Adallom.

Ma, D. and A. Seidmann (2014, Feb). Analyzing Software-as-a-Service with per-transaction charges. Working paper, Singapore Management University and University of Rochester.

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

Markoff, J. (2009, Aug). Defying experts, rogue computer code still lurks. *The New York Times*.

McBride, S. (2005, Feb). Zero day attack imminent. *Computerworld*.

Mell, P. and T. Grance (2011, Sep). The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce.

Messmer, E. (2013, Dec). Identity-theft vulnerability fixed in Microsoft Office 365, says security firm. *NetworkWorld*.

Microsoft (2013). Security in Office 365. Whitepaper. Available at `http://www.microsoft.com/en-us/download/details.aspx?id=26552`.

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

Muller, H. M. (2000). Asymptotic efficiency in dynamic principal-agent problems. *Journal of Economic Theory 91*, 292–301.

Neti, S., A. Somayaji, and M. E. Locasto (2012). Software diversity: Security, entropy and game theory. In *Proceedings of the 7th USENIX conference on Hot Topics in Security*, HotSec'12, Berkeley, CA, USA, pp. 5–5. USENIX Association.

Niculescu, M. F. and D. J. Wu (2014). Economics of free under perpetual licensing: Implications for the software industry. *Information Systems Research 25*(1), 173–199.

Nochenson, A., J. Grossklags, and C. F. L. Heimann (2014). How loss profiles reveal behavioral biases in interdependent security decisions. Forthcoming in *International Journal of Internet Technology and Secured Transactions*.

O'Neill, S. (2011, Sep). Survey: Value of the cloud, telecommuting overstated. *CIO*.

Pesendorfer, W. and J. M. Swinkels (2000). Efficiency and information aggregation in auctions. *American Economic Review 90*(3), 499–525.

Png, I. P. and Q. Wang (2009). Information security: Facilitating user precautions vis-à-vis enforcement against attackers. *Journal of Management Information Systems 26*(2), 97–121.

Ransbotham, S., S. Mitra, and J. Ramsey (2012). Are markets for vulnerabilities effective? *MIS Quarterly 36*(1), 43–64.

Robertson, J. (2014, Apr). Heartbleed fixes taking longer as websites plug gaps. *Bloomberg.com*.

Roy, D. (2011, Aug). Data on sale. *CIO*.

Schneider, F. and K. Birman (2009). The monoculture risk put into context. *IEEE Security & Privacy 7*(1), 14–17.

Vereshchagina, G. and H. A. Hopenhayn (2009). Risk taking by entrepreneurs. *American Economic Review 99*(5), 1808–1830.

Villeneuve, N. (2011, Oct). Trends in targeted attacks. *Trend Micro*.

Weatherwax, E., J. Knight, and A. Nguyen-Tuong (2009). A model of secretless security in N-variant systems. In *Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '09. IEEE Computer Society.

Wei, X. and B. R. Nault (2011, Feb). Vertically differentiated information goods: Monopoly power through versioning. Working paper, Fudan University and University of Calgary.

Wei, X. and B. R. Nault (2014). Monopoly versioning of information goods when consumers have group tastes. Forthcoming in *Production and Operations Management*.

Xu, J., Z. Kalbarczyk, and R. Iyer (2003). Transparent runtime randomization for security. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS 2003)*, SRDS 03, pp. 260–269. IEEE Computer Society.

Zhang, J. J. and A. Seidmann (2010). Perpetual versus subscription licensing under quality uncertainty and network externality effects. *Journal of Management Information Systems 27*(1), 39–68.