MIS Quarterly

RESEARCH ARTICLE

CORRELATED FAILURES, DIVERSIFICATION, AND INFORMATION SECURITY RISK MANAGEMENT[1]

Pei-yu Chen
Department of Management Information Systems, Fox School of Business and Management, Temple University,
1801 N. Broad Street, Philadelphia, PA 19122 U.S.A. {pychen@temple.edu}

Gaurav Kataria
Booz & Co., 127 Public Squire, Suite 5300, Cleveland, OH 44114 U.S.A. {gkataria@google.com}

Ramayya Krishnan
School of Information Systems and Management, The Heinz College, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213 U.S.A. {rk2x@cmu.edu}

The increasing dependence on information networks for business operations has focused managerial attention on managing risks posed by failure of these networks. In this paper, we develop models to assess the risk of failure on the availability of an information network due to attacks that exploit software vulnerabilities. Software vulnerabilities arise from software installed on the nodes of the network. When the same software stack is installed on multiple nodes on the network, software vulnerabilities are shared among them. These shared vulnerabilities can result in correlated failure of multiple nodes resulting in longer repair times and greater loss of availability of the network. Considering positive network effects (e.g., compatibility) alone without taking the risks of correlated failure and the resulting downtime into account would lead to overinvestment in homogeneous software deployment. Exploiting characteristics unique to information networks, we present a queuing model that allows us to quantify downtime loss faced by a rm as a function of (1) investment in security technologies to avert attacks, (2) software diversification to limit the risk of correlated failure under attacks, and (3) investment in IT resources to repair failures due to attacks. The novelty of this method is that we endogenize the failure distribution and the node correlation distribution, and show how the diversification strategy and other security measures/investments may impact these two distributions, which in turn determine the security loss faced by the firm. We analyze and discuss the effectiveness of diversification strategy under different operating conditions and in the presence of changing vulnerabilities. We also take into account the benefits and costs of a diversification strategy. Our analysis provides conditions under which diversification strategy is advantageous.

Keywords: Security, diversification, downtime loss, software allocation, network effects, risk management, correlated failures

[1]H. Raghav Rao was the accepting senior editor for this paper. Ram Gopal served as the associate editor.

The appendix for this paper is located in the "Online Supplements" section of the *MIS Quarterly*'s website (http://www.misq.org).

Introduction

Network effects, arising from the need for compatibility, have been the driving force underlying a firm's decision on technology adoption (i.e., whether to adopt, what to adopt, and when to adopt) (Brynjolfsson and Kemerer 1996; Katz and Shapiro 1985, 1986). In the case of software adoption, firms often find it more valuable to adopt software with a larger market share. By making a choice that is compatible both internally as well as with their partners, firms enjoy positive network effects stemming from greater benefits of compatibility and interoperability. As a result, markets with network effects are usually "tippy" (i.e., tipping in favor of one product) (Farrell and Klemperer 2001; Katz and Shapiro 1985, 1994). The rise of Microsoft Windows as the most popular choice for desktop operating system is largely attributed to this very fact (Economides 2001).

However, the negative network externality associated with a consuming software with large market share is often ignored. First, more attacks target popular software (Honeynet Project 2004; Symantec Corporation 2006). Second, by using popular software to interact with many partners, firms risk being attacked and affected by breaches at their partners' (Kunreuther and Heal 2003; Ogut et al. 2005). Third, maintaining networks that are internally homogeneous (i.e., all computers on the network share the same vulnerability) substantially increases the possibility of concurrent failure of multiple systems given an exploit. Widely replicating worms and viruses have highlighted this very threat as seen in case of MS-Blaster (CERT CA-2003-20) and SQL Slammer (CERT CA-2003-04). Since viruses, worms, and other network exploits can easily seek and target any vulnerable computer connected to the network, networks that are internally homogeneous are likely to observe high correlation in failure of their devices and services due to such attacks (Bain et al. 2001).

It may be that in considering positive network externalities alone and disregarding negative externalities firms have overinvested in homogeneous systems. An influential report titled "Cyber*In*security: The Cost of Monopoly" has argued that this has led to market failure in the case of the operating systems market (Geer et al. 2003). Inspired by biology research showing that the human race has survived deadly attacks that have plagued our history because of diversity due to heterogeneity, Sharman et al. (2004) proposed a new way to look at security, which they term *functionality defense by heterogeneity*. The idea is that rather than trying to defend a single system, we should defend functionality which can be achieved through heterogeneity (i.e., by having different hardware and software in place); this is consistent with what we propose in this paper. However, diversity comes with the costs of compromised network effects and the lack of economies of scale in maintaining heterogeneous software environments. Therefore, it is not clear whether and under what conditions maintaining a diversity of systems is a better strategy for firms.

The first goal of this research is, therefore, to provide a formal analysis to examine whether and when a firm can benefit from internally maintaining a diversity of systems by moving away from the dominant strategy of maintaining homogeneous systems that allow firms to take advantage of positive network effects and economies of scale. To address this question, we need to quantify the benefits and costs of a diversification strategy, and, therefore, our second goal is to develop a risk management framework to derive the optimal level of diversity at the firm level, taking into account the benefits and costs of a diversification strategy. Our approach uses network downtime resulting in unavailability of service/operations as a measure of security loss since downtime has been directly linked to financial loss. We develop an approach based on queuing theory for determining network downtime in the face of correlated failures. The software diversification strategy proposed in this study can be used to reduce shared vulnerabilities across nodes and therefore downtime loss by installing different software stacks on different nodes. In practice, software diversity can also be incorporated at a modular level by deploying *n* version software, which have some common and some diverse components (Avizienis 1995; Deswarte et al. 1998), and techniques like stack randomization are useful in introducing compile-time and run-time diversity (Barrantes et al. 2003; Kc et al. 2003). We show that mixing software with a different level of security (in the sense of receiving different number of attacks) can make a firm better off than using just the most secure software alone (unless the software has no vulnerability and receives zero attacks). This is the advantage of diversity: even though the firm may face more attacks due to more distinct vulnerabilities when employing heterogeneous software, the consequence of each attack would be of smaller magnitude and more manageable. By reducing shared vulnerabilities and, therefore, the variance of attack magnitude, the firm greatly reduces the possibility of drastic loss.

The operation of security involves three levels: prevention, detection, and response (White et al. 2004). Our risk management model incorporates all three levels of security operation: Software diversification can be used to help prevent drastic loss, while our model also accommodates other security instruments, such as antivirus software and firewalls, which help in detecting and curtailing attacks. Our model incor-

porates response by the repair time needed to bring nodes or the system back to a usable state. A firm may invest in response by increasing IT service capability, which helps to reduce repair time. This holistic and practice oriented framework allows us to compare the effectiveness of software diversification to the effectiveness of other security instruments in reducing loss and to study how software diversification can be used together with these other security measures to reduce security loss. We also note that while security technologies such as firewalls, intrusion detection systems (IDS), and antivirus software can reduce the likelihood of a vulnerability being successfully exploited, they cannot eliminate the vulnerabilities present in deployed systems. In a homogeneous software deployment, the vulnerabilities are shared by all computers on the network (i.e., every computer is vulnerable to an exploit targeting the particular vulnerability), regardless of whether these computers are directly connected or not. Software diversification, on the other hand, can isolate these vulnerabilities such that opportunistic attacks (exploiting a particular vulnerability) are effective against only a smaller subset of computers than under homogeneous software deployment, thereby reinforcing the effectiveness of other security measures, such as firewalls and IDS.

The concept of using diversification to limit correlated risks has been used in other settings, for example insurance and finance (Varian 1992). However, there are some fundamental differences between these settings and information systems. One of the most notable differences is that, in information systems, there is significant positive network effect associated with homogeneity, which is not present in finance and insurance. The potential lack of network effects together with possible loss of economies of scale in maintaining heterogeneous software represent major costs of a diversification strategy. Therefore, a diversification strategy in IS needs to take into account these costs. Our analysis suggests that expected security loss, which is often ignored in the decision-making process, is as important as network effect and economies of scale in formulating a firm's system acquisition and deployment strategy. We show that under general conditions, optimal diversity level is an interior solution, and diversity strategy is especially effective when other security measures are imperfect. In the case that guaranteed level of service is required, the result would, in most cases, involve a diversification strategy.

Overall, this research makes several unique contributions to the literature. First, it contributes to the IS and the economics of network effects and standardization literature by (1) demonstrating the importance and benefits of diversification in IT, (2) developing a general framework to help guide a firm's IT investments, and (3) exploring the conditions when the diversification strategy is effective. We also add to the security management literature by developing models to quantify security loss and by introducing the concepts of the *vulnerability matrix* and the *node failure correlation matrix* to help firms better manage security risks. While our model to estimate security loss due to unavailability (i.e., system downtime) is based on well-established queuing models, one innovation of our model is that the distribution from which the number of requests sent to the queue is drawn is endogenous to system variables, such as the level of diversity, the number of shared vulnerabilities, and the type of other security measures used. Therefore, by adequately setting the system variables, a firm can influence the performance of the queue and expected downtime loss. We believe that our model is not only of practical value to firms, but that our finding that diversification strategy can be an effective way to reduce security loss also provides important policy implications. Specifically, the required condition for diversification strategy to be viable is the existence of multiple compatible software with low shared vulnerability. Given such a condition, this suggests that the government has to ensure an environment where multiple software may exist and where the number of shared vulnerabilities among different software applications is minimized. The various policy levers that can be utilized by the government to promote diversity include public procurement, new technology grants, antitrust enforcement, and interoperability standards.

The rest of the paper is organized as follows. In the next section, we introduce system downtime as a metric for security loss and present a queuing model for estimating this loss. One innovation of our queuing model is that the distribution from which the number of requests sent to the queue is drawn is endogenous to system variables. In the subsequent section, we analyze the effectiveness of the diversification strategy in reducing expected downtime and show how optimal diversification can be determined when the goal is to minimize system downtime. We then discuss the benefits (i.e., lower downtime loss) and the costs of diversification (due to interoperability, integration, and support), and provide a framework that takes into account these benefits and costs of diversification and allows a firm to determine its optimal software deployment strategy to maximize its expected overall utility. We discuss conditions where diversity would be preferred. Moreover, since vulnerabilities evolve (old vulnerabilities are patched and new vulnerabilities are discovered), we also present an optimal allocation method to adapt the diversification strategy to changing vulnerabilities. Finally, we conclude by discussing our results and their implications and presenting ideas for future work.

A Model for Measuring Security Loss Due to Unavailability of Systems

As discussed earlier, security loss is usually not accounted for in deriving a firm's IT acquisition strategy in the literature or in practice, leading to over-investment in homogeneity. A reason for this is that it is difficult to quantify security loss. In this section, we introduce expected system downtime as a metric for quantifying loss and develop a model to calculate it. System downtime that results in the system's unavailability of service/operations has been directly linked to financial loss. For example, eBay's outage in 2000 has been linked to an unbelievable $5 billion loss (or about $225,000 per hour of downtime), while estimates are that the per hour downtime for brokerage operations can reach over $6 million; for package shipping services, it is about $150,000; for airline reservation services, it is $89,000; and for ATM services, it is $15,000 (Patterson 2002).

In the following, we assume that when an attacker successfully exploits a vulnerability through a worm, a virus, or a hack, the computer hosting the system is affected and will need to be "repaired" to be put back into service. System downtime is the time duration from when the successful exploit takes place to when the computer that is exploited is back in service. When multiple computers are affected under an attack, the systems affected in an attack have to wait while other affected systems are serviced. We note that system downtime is taken from the perspective of the firm. Firms and customers may have different views of downtime in the case that multiple computers are used to support operations. For example, when just one computer, among several that are used to support operations, is taken down, it reduces the operating capacity of the firm, and there is downtime from the firm's perspective, although there is no downtime time from the customers' perspective.

We model the sequential nature of diagnosis and repair of affected computers as a service queue (Figure 1). Note that while the focus of our analysis is on the shared vulnerabilities between applications and software systems, the approach is readily applicable to business processes. In the case of business processes, the system can be defined as a set of correlated or dependent business processes supported by information systems, and the downtime is the time duration from when the successful exploit takes place to when affected business processes that are exploited are able to function. When a process is down due to an attack to a node hosting systems that execute either steps within a process or the entire process, then other processes that depend (either through shared steps or input–output dependence) on the "downed" process may be non-functional even though they run on non-attached nodes. In this case of correlated processes, failure of one can cause unavailability of several processes or even the system as a whole. However, regardless of the level of analysis, downtime is still the time it takes to repair the failed node, no matter how many processes are affected.

Next, we present a queuing model to estimate expected downtime. To formally model the expected downtime, we model each arriving attack incident[2] as an event that sends a set of failed computers to the IT department service queue for repair. The number of computers failed in an incident is dependent on how vulnerable (in terms of probability of failure and how vulnerability is shared among nodes) the firm is to each attack, which in turn is determined by the type of software and the security measures used by the firm. We develop a model and provide several tools to estimate the number of computers failed in an incident. We then discuss how security incidents are modeled. For analytical tractability, we assume that the security incidents happen independently following a Poisson arrival process. However, we note that some incidents may be correlated themselves due to evolving attack tactics such as several variants of worms appearing frequently, but these dynamics should strengthen our results even more, as correlated incidents would cause more buildup in the queue, causing longer wait times.

Downtime Due to an Attack

We adopt the batch Poisson process with an $M^X/G/1$ queue to model system downtime due to security incidents (Kleinrock 1975). In an $M^X/G/1$ queuing system, requests for service arrive to the system according to a Poisson process with rate λ. The units within a batch are served one at a time by a single server, which corresponds to the firm's overall IS/IT department, according to some service time distribution. The single server and the sequential service on failed computers do not always hold because computers failed in the same attack may be fixed concurrently by dispatching the patches through the network instead of individually one by one. We would like to note that although many preventive measures such as installing software patches can be carried out in parallel across multiple computers, once a computer is infected, curative measures require individual attention to account for differences in user settings, permissions, data backups, and custom applications. In addition, we would like to point out that the insights derived from the single-server analysis would

[2] An incident can be defined as a collection of similar attacks spaced together in time (Howard 1997).

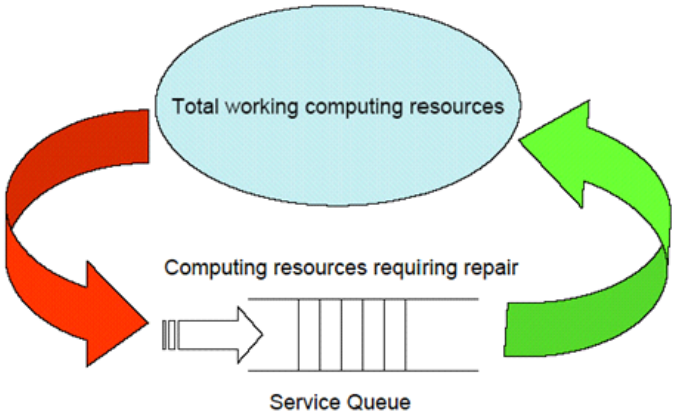


Figure 1. A Service Queue Repair Model for Failed Computers

still hold in the case of multiple servers as long as there is a repair queue. Since, in most cases, the repair capacity is less than the total resources in the firm, it is reasonable to expect the queue buildup in some cases.

In our context, each batch corresponds to a security incident, and the number of units in a batch corresponds to the number of computers (Y) affected in an incident, which depends on the type of attacks a firm faces and the susceptibility of its computers to those attacks. Once E[Y] is known, we can borrow from the literature to calculate the total downtime experienced in each incident. However, the challenge is that the distribution from which Y is drawn depends on many endogenous factors controlled by the firm, such as the type of software installed, system configurations, and other security measures used by the firm. One unique feature of our model that makes us different from the conventional M^X/G/1 models is that we endogenously determine E[Y] by identifying the factors that affect E[Y] and deriving E[Y] as a function of these factors (please see the next subsection for derivations). Table 1 summarizes the major notations used in our risk framework.

Assuming, in the meantime, that E[Y] is given, the mean arrival rate of affected computers to the service queue (γ) is then a function of how many attack incidents occur per unit time (λ) and how many computers fail on average per such incident (i.e., E[Y]). Therefore,

$$\gamma = \lambda \times \mathrm{E}[Y] \tag{1}$$

The downtime (or sojourn time) for the computers affected in an incident is the sum of service time (S) plus the waiting time (W) computers have to wait before other affected computers are serviced. The expected downtime per computer is given by

$$E[T] = E[S] + E[W] \tag{2}$$

Sohraby (1989) has shown that for an M^X/G/1 queue with deterministic service time, d, per computer, where queue utilization $= \lambda d < 1$, the average wait time per computer will be

$$E[W] = d\left[\frac{Bat + \theta - 1}{2(1-\theta)}\right] \tag{3}$$

Where, Batchiness, Bat, of a batched Poisson process is given by (Eckberg 1985)

$$Bat = \frac{E[Y^2]}{E[Y]} \tag{4}$$

Substituting (3) in (2),

$$E[T] = d\left[\frac{1 + Bat - \theta}{2(1-\theta)}\right] \tag{5}$$

The overall economic loss due to unavailability of service experienced by the firm is equal to downtime per computer, E[T], times number of computers affected per unit time, γ. Therefore, the expected downtime loss is given by

$$DT = E[T] \times \gamma \tag{6}$$

In the next subsection, we focus on modeling the number of computers affected per incident (Y). As noted earlier, one major contribution of our model is that E[Y] is treated as an endogenous variable, and we also introduce tools to manage and identify factors that affect E[Y].

Table 1. Major Notations Used in the Risk Framework	
Y	Number of computers failed in an incident
N	Total number of nodes
λ	Arrival rate of attacks
γ	Arrival rate of computers to the service queue ($\gamma = \lambda \times E[Y]$)
S	Total service time
W	Total wait time before a computer can be serviced
d	Service time per computer
T	Total downtime per incident
DT; DT(x)	Total downtime (across all attacks) under diversity level x
θ	Queue utilization = λd
Bat	Defined as $E[Y^2]/E[Y]$
P_i	Probability of attacks node i receives
P_{ij}	Probability of attacks node i and j have in common
π	Unconditional probability of computer failure in an attack ($0 \le \pi \le 1$)
ρ	Correlation of failure across nodes: when $\rho = 0$, nodes fail independently; $\rho = 1$, all nodes fail together ($0 \le \rho \le 1$)
m	Relative number of attacks faced by another software configuration (e.g., when software 1 receives λ attacks, software 2 would receive mλ attacks)
c	Proportion of attacks that are common across software ($c \le \min\{1,m\}$)
x_i	The proportion of nodes allocated with software configuration i
b	Degree of interoperability
NE(x)	Network effects benefits under diversity level x
C_{NE}	Scaling constant of network effects benefits and security loss
MC(x)	Maintenance costs under diversity level x
C_{MC}	Scaling constant of maintenance costs and security loss
a_k	Allocation k, indicating the k^{th} software configuration

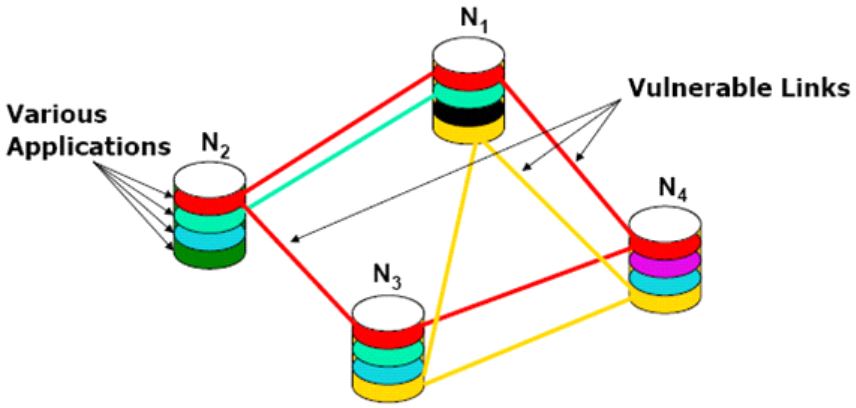
Number of Computers Affected per Incident (Y)

Firms usually have many computers on their network, each running several software applications (Figure 2). These software applications often have vulnerabilities in them, which, when exploited, cause correlated failure of nodes that run those vulnerable applications. This is because attacks that are successful against one installation of vulnerable application are also likely to succeed on other installations of the same application. The overall correlation in failure of nodes can thus be determined by considering all the shared vulnerabilities present in them because of shared components in their software configuration. For example, a system with configuration 1, comprising of Linux OS, Firefox browser, Thunderbird e-mail client, and MySQL database, is unlikely to have a failure correlated with a system running configuration 2, comprising of Windows OS, IE browser, Outlook e-mail client, and SQL server.[3] On the other hand, if the systems have some common applications like a Flash plug-in, then attacks targeting a vulnerability in the Flash plug-in can affect both systems and hence increase the likelihood of correlated failure.

Failure Correlation

The correlation structure for node failure can be derived from a representation we introduce called *vulnerability matrix*. A vulnerability matrix is a mapping of software vulnerabilities to nodes (computers) on the network. To derive the vulnerability matrix, we first define the concept of *configuration*,

[3]Attacks that exploit protocol-level vulnerabilities can affect disparate software that share a common interface. We discuss this in detail later.



The organization's computer network with node having multiple software applications installed on them. Vulnerabilities in those applications, when exploited, cause correlated failure of nodes that share the same vulnerability (i.e., have a vulnerable link between them).

Figure 2. Representative Computer Network

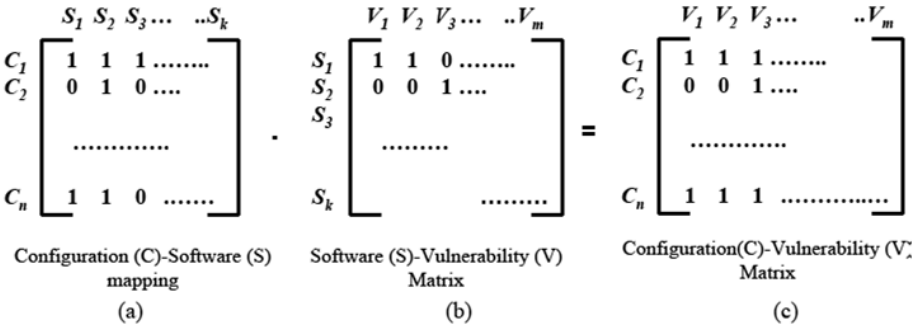
which is a set or stack of specific software to fulfill the functional requirements of a node, for example, Microsoft Windows, Microsoft Office, Outlook, and Internet Explorer (Figure 3a). Each node is assigned a configuration (Figure 4a). The software running on each node (or node–software mapping) is essentially the dot product of node–configuration mapping (Figure 4a) and configuration-software mapping (Figure 3a). The data required to populate the vulnerability matrix is derived from two sets of sources. The nodes or configurations can be obtained from the firms' configuration management database[4] (CMDB). The CMDB lists the software configurations or applications installed on each node and is an important resource maintained by most IT departments. The CMDB is represented in the matrix shown in Figure 3a. The list of vulnerabilities present in each software application (Figure 3b) is available from the National Vulnerability Database (see http://nvd.nist.gov/). Taking the dot product of these matrices (then dichotomizing values to binary) yields the configuration vulnerability matrix shown in Figure 3c. A row in this matrix represents the set of vulnerabilities present in a configuration and the columns represent the set of configurations that share a particular vulnerability. Such a configuration vulnerability matrix is useful when a firm is considering which configuration(s) to install and how to assign configurations to nodes. By taking the dot product of the node–configuration mapping and the configuration–vulnerability matrix, one obtains the vulnerability matrix (Figure 4c).[5] This vulnerability matrix changes as new software applications are installed or when new vulnerabilities are discovered in existing software and old vulnerabilities are patched or removed.

The vulnerability matrix is an important situational awareness representation informing the system administrators about which nodes on the network are vulnerable to an attack exploiting a specific vulnerability. We can further weight the vulnerability matrix by the severity level or impact of vulnerability (Park et al. 2007) to get better estimate of the shared risk among nodes. There are network management tools available from vendors such as IBM (Tivoli), HP (Openview), and Computer Associates (Unicenter), and open-source tools such as Nessus[6] that provide real-time information to the administrator about the state of hosts on their network in terms of software configuration and network connectivity. Vulnerability scanners in combination with these network management tools can readily populate the vulnerability matrix that we describe above. The recently released IF-MAP protocol which aims to provide a big picture of the network by "creating a structured way to store, correlate, and retrieve identity, access control, and security posture information about

[4]A configuration management database, as stipulated by the IT Infrastructure Library's best practice, is a compendium of IT assets of a firm. It contains information about all hardware and software components present on the network.

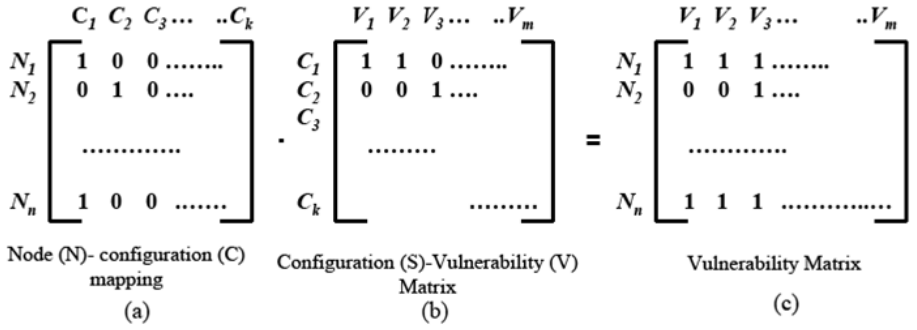
[5]Another way to obtain the vulnerability matrix is by taking the dot product of the node–software mapping and the software–vulnerability matrix.

[6]http://nessus.org.



A time varying mapping of vulnerabilities to configurations on a network; derived by multiplying configuration (C) – software (S) mapping with the software (S) – vulnerability (V) matrix.

Figure 3. Configuration Vulnerability Matrix



A time varying mapping of vulnerabilities to nodes on a network; derived by multiplying node (N) – configuration (C) mapping with the configuration (C) – vulnerability (V) matrix.

Figure 4. Vulnerability Matrix

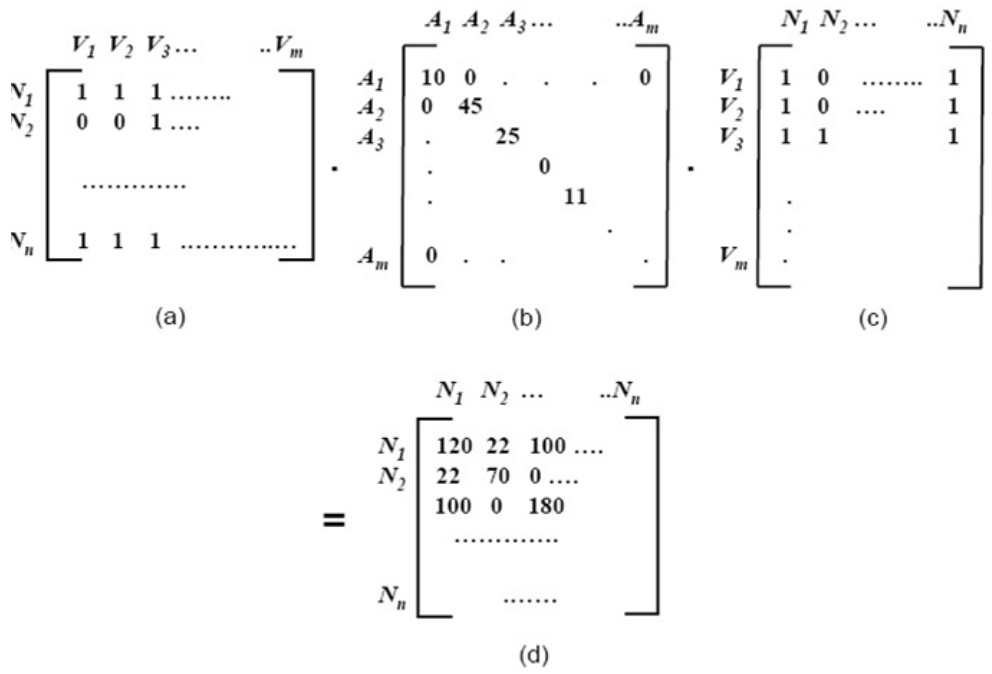
users and devices on a network"[7] would also be useful to create a dynamic vulnerability matrix. Tools implemented with both IF-MAP protocol and the vulnerability matrix will provide better information for an organization to understand and estimate the risk it faces and act accordingly.

Given the vulnerability matrix and intelligence reports about attack trends,[8] a system administrator can determine the probability of any two nodes failing together. For example, given the vulnerability matrix in Figure 4c, nodes N_1 and N_2 will not have simultaneous failure under an attack that exploits vulnerability V_1 because they do not share that vulnerability. However, they are likely to fail together when vulnerability V_3, which they do share, is exploited by an attack. Overall, the correlation in failure of nodes is higher if they share more vulnerabilities and if those vulnerabilities are attacked often. Post multiplying the transpose of vulnerability matrix with the product of vulnerability matrix and attack trend matrix (Figure 5b) determines the number of shared attacks between any pair of nodes (Figure 5d). Given the joint susceptibility of nodes to various types of attacks, we can compute the correlation of failure for a pair of nodes on the network as the correlation between two Bernoulli random variables:

$$\rho_{ij} = \frac{p_{ij} - p_i p_j}{\sqrt{p_i(1-p_i)p_j(1-p_j)}} \quad (7)$$

[7] http://www.infoblox.com/solutions/pdf/IFMAP-wp.pdf.

[8] CERT/CC (www.cert.org), CAIDA (www.caida.org), and SANS Institute (www.sans.org), along with many private security consultants, provide reports on attack trends to their clients and the public at large. Many large firms also use their internal IDS logs and honeynet logs to characterize attack trends.



Post multiplying the transpose of vulnerability matrix (c) with the product vulnerability matrix (a) and attack trend matrix (b) determines the number of shared attacks between any pair of nodes (d)

Figure 5. Shared Attacks Matrix

Where

p_i = probability/percentage of attacks node i receives
p_j = probability/percentage of attachs node j receives
p_{ij} = probability/percentage of attacks nodes i and j have in common

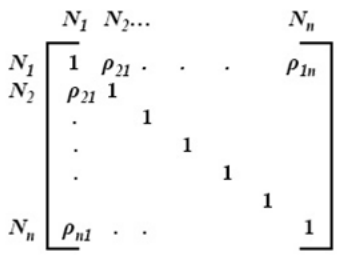
Given the expression for ρ_{ij}, if the network administrators have information regarding all software, vulnerabilities, and attacks on their network, then they can accurately calculate node correlation matrix (Figure 6). However, network administrators usually do not have complete information about susceptibility of nodes due to specific actions taken by users and/or attackers, which may lead to errors in their estimation of p_i, p_j, and p_{ij}. For example, nodes that share same configuration can be expected to have $p_i = p_j = p_{ij}$, which would imply correlation of 1; however, due to other contextual differences between nodes (e.g., user specified configuration settings), the correlation may be less than 1.

In the following subsection, we observe the effect correlation has on node failure distribution.

Failure Distribution

The number of nodes affected in an attack depends on the probability of an attack being successful (p) and the failure correlation (ρ). The node failure distribution reflects how vulnerable an organization is to a particular attack. Specifically, we define the correlation of failure between two nodes as the chance when the same vulnerability is exploited on these two nodes, or, equivalently, the correlation of failure measures the probability of failure of one node under a particular attack conditional on the other node being failed under the same attack. A perfect correlation of failure means that either no node fails or all nodes fail together. This definition of correlation of failure focuses on horizontal correlation among nodes. Note that failures may also be correlated vertically across vulnerabilities when multiple vulnerabilities are exploited on the same or different systems (e.g., a browser hole leads to a keylogger, installation which leads to password stealing, which leads to data theft).[9] This kind of correlated

[9]We thank an anonymous reviewer for pointing out this interesting example.



ρ_{ij} is the failure correlation between node *i* and node *j*

Figure 6. Failure Correlation Matrix

failure across vulnerabilities can be accommodated in our model in the following way, using the browser hole example: the browser hole would be the vulnerability we identify in the paper, while the keylogger installation, password stealing, and data theft would be the consequence of the vulnerability being exploited, and some efforts are then needed to repair the system, causing downtime to the system. In our research, we investigate the benefit of software diversification as an approach to reducing correlated failure among nodes because diversifying the software can successfully partition the vulnerability matrix and lower the failure correlation by reducing shared vulnerabilities.

We model the failure of a node as a Bernoulli trial. Since vulnerabilities are shared among nodes, the failure of one node is not independent of failure of other nodes on the network. Therefore, the total number of node failures can be considered as an outcome of a collection of correlated Bernoulli trials where each node failure is a coin-toss, the outcome of which depends on the outcome of other trials.

We model failure of nodes that share the same software using the beta–binomial distribution, which has been used in the computer science literature to model correlated failure of backup systems and failure across multiple versions of the same software (Bakkaloglu et al. 2002; Nicola and Goyal 1990). The beta–binomial distribution is computed by randomizing the parameter p (probability of failure) of the **binomial** distribution by the beta distribution. In general, the randomized binomial distribution is given by

$$b_{N(y)} = \int_0^1 c(N,y) * p^y (1-p)^{N-y} f_p(p) dp \qquad (8)$$

where

N = Total number of nodes[10]

$C(N, y)$ = Number of ways in which y out of N nodes can fail

The intensity function $f_p(p)$ gives the probability distribution that a fraction of all nodes fail. It is a unit impulse at average p when there is no correlation.[11] For the beta–binomial distribution, the intensity function follows the beta distribution:

$$f_p(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}; \quad 0 < p < 1, \quad \alpha,\beta > 0 \qquad (9)$$

where $B(\alpha, \beta)$ is the beta function with parameters α and β given by

$$\alpha = \pi\left(\frac{1}{\rho} - 1\right), \quad \beta = (1-\pi)\left(\frac{1}{\rho} - 1\right) \qquad (10)$$

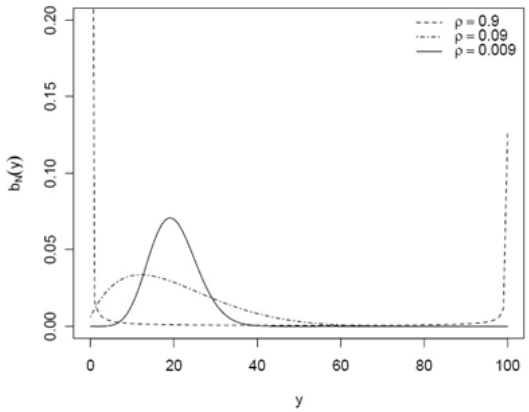
where

$$\pi = \frac{\alpha}{\alpha+\beta}, \quad \rho = \frac{1}{\alpha+\beta+1}$$

π is the unconditional probability of computer failure in an attack (i.e., the expected value of the random variable p), and $0 \le \rho \le 1$ is the correlation of failure across nodes. The limiting case $\rho = 0$ is when all computers fail independently (leading to binomial distribution for the number of failed com-

[10]Without loss of generality, in our examples for the rest of this paper, we assume the total number of nodes on the network to be 100.

[11]It then results in a binomial distribution, which is basically a collection of independent Bernoulli trials.



The beta–binomial distribution for computer failure on a 100 node network resembles binomial distribution for small values of ρ; however, as ρ increases, the probability mass shifts toward the ends, exhibiting highly correlated failure.

Figure 7. Beta–Binomial Distribution

puters),[12] whereas the limiting case $\rho = 1$ is when all computers fail together (perfectly correlated) as shown in Figure 7.[13]

Given this, the probability of y out of N computers failing in an attack is computed by substituting (9) in (8)

$$b_{N(y)} = \frac{B(\alpha + y, N + \beta - y)}{(N+1) * B(N - y + 1, y + 1) * B(\alpha, \beta)} \tag{11}$$

The mean and variance of beta–binomial distribution are given by

$$\mu = N\pi; \quad \sigma^2 = N\pi(1 - \pi)\rho\left(\frac{1}{\rho} - 1 + N\right) \tag{12}$$

If we have many diverse software configurations, then

$$E[Y] = \sum_{i=1}^{N} \pi_i; \quad E[Y^2] = \sum_{i=1}^{N} \sum_{j=1}^{N} Cov(i, j) \tag{13}$$

Where Y is the number of node failures in an attack incident. It is important to observe that a cautiously managed homogeneous network may exhibit low mean but still have high correlation in failure and therefore high variance, which can be detrimental to a firm seeking low variance in downtime loss. In the next section, we consider the tradeoff between investment in software diversification to limit correlation *vis-à-vis* investment in other security technologies. In the next subsection, we model the dynamics of attack rate.

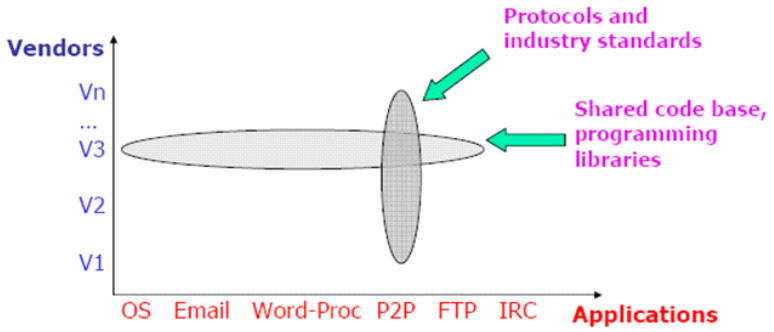
Number of Attacks (λ)

The average arrival rate of attacks faced by a software product may depend on many factors including type of software, industry where it is used, inherent security level of software, market share, sentiment against the software product, etc. There is some evidence suggesting that market share appears to be highly correlated to the number of attacks faced by a product (Symantec Corporation 2006). For instance, Windows, with over 90 percent market share in the desktop operating system market, receives considerably more attacks than Linux, Unix, and Solaris combined (Honeynet Project 2004). Consider a stylized model where a firm faces two different software configurations or choices that are functionally equivalent: software (configuration) 1 and software (configuration) 2.[14] We accommodate the heterogeneous attack rates across software by assuming that if software 1 on average faces λ attacks then software 2 would face $m \cdot \lambda$

[12] Since, in this setup nodes share the same software, there is no possibility of negative correlation.

[13] Without loss of generality, for the rest of this paper we assume ρ to have an intermediate value of 0.5 (i.e., some amount of correlation).

[14] For simplicity, we denote them as software 1 and software 2 in the paper.



Vulnerabilities may be shared by multiple software configurations due to shared code base or programming libraries from a vendor or due to flaws in protocols and industry standards.

Figure 8. Software Vulnerability

attacks where, m may depend on the relative market share and/or the relative security quality of the two software configurations. The basis for this assumption is that, from the attackers' point of view, there is both a direct as well as an indirect externality in attacking the software with large market share. A few skilled hackers develop tools to exploit vulnerabilities and a large number of script-kiddies (unskilled hackers) use those tools to attack as many computers as they can. Therefore, there is a force multiplier effect in the case of exploits that target software with a large installed base (Collins et al. 2006).

Most real-world software products are not completely independent of each other in terms of vulnerabilities (Figure 8). Many interfaces and functionalities are standardized, leading to similar or reused code. In many cases, code is also shared via the use of programming libraries. Due to this shared code base, it is inevitable that some vulnerabilities are common across different software. For example, Microsoft's Graphic Device Interface Plus (GDI+) library contained a vulnerability in the processing of JPEG images, which could be exploited in any application that uses the GDI+ library.[15] To capture the effect of attacks that target vulnerabilities present in multiple software configurations, we introduce a parameter c to account for the proportion of attacks that are common across software. Given m and c, the distribution of attacks is such that

$$Prob(attack = 1only) = \frac{1-c}{1+m-c} \tag{14}$$

$$Prob(attack = 2only) = \frac{m-c}{1=m-c} \tag{15}$$

$$Prob(attack = common) = \frac{c}{1+m-c} \tag{16}$$

If all attacks were successful in causing node failure, then the correlation in failure between two software configurations can be expressed using Equation (7) as

$$\rho_{12} = \frac{c(1+m-c)-m}{\sqrt{m(m-c)}}$$

This leads to the following observation:

> **Observation 1:** *As the number of shared vulnerabilities between two software configurations increases, their correlation of failure also increases.*[16]

In summary, the framework introduced in this section highlights several factors that have an impact on security loss: rate of attacks to a particular software λ, probability of node failure π, correlation of failure across nodes ρ, and service time per computer d. A firm may reduce its security loss by influencing one or more of these factors through employing different security measures. For example, a firm may reduce λ by choosing a more secure software, reduce π by installing a firewall or an intrusion detection system, and reduce d by enhancing the service capacity/efficiency. The proposed

[15]http://www.us-cert.gov/cas/techalerts/TA04-260A.html.

[16]Please see Appendix A for all proofs.

diversification strategy aims to reduce ρ by reducing the shared vulnerability among nodes. The rate of attacks each software configuration receives (λ and $m\lambda$) and the number of common attacks received by different software configurations (c) are key factors that influence the effectiveness of the diversification strategy.

We should note that in addition to the commonality/diversity of software, network connectivity may also impact correlation of failure among nodes (Figure 6); however, the impact of network connectivity on node failure correlation is of second order. One important thing to note is that zero connectivity doesn't mean zero failure correlation. For nodes in disconnected networks but sharing the same vulnerability, their failure will not be independent of each other (that is, their failure correlation is nonzero) because they still face the same vulnerability and attacks that exploit the shared vulnerability and will have a positive correlation of failure. Network connectivity, however, will cause higher node failure correlation for nodes sharing the same vulnerabilities in the same network. On the other hand, network connectivity doesn't make a node more prone to failure when there is no shared vulnerability. From the discussion above, we know that commonality/diversity of software is the most important determinant of the node correlation matrix, while high network connectivity can boost the node failure correlation but zero connectivity doesn't eliminate the node failure correlation. In this paper, we assume that network connectivity is given (which is realistic since, in most cases, many node connections are required for functional reasons and cannot be changed) and, as a result, the decision of diversity is independent of network connectivity. In the following section, we explore how software diversification can be used as a means to reduce correlation and thus system downtime.

Software Diversification to Reduce Downtime Loss

In this section, we show how software diversity can be applied in practice to reduce system downtime. First, without loss of generality, we use a stylized two-node model to demonstrate that software diversity under certain conditions can stochastically dominate software homogeneity. Stochastic dominance is a comprehensive criteria to prove that a distribution of outcomes is superior to another (see Hadar and Russell 1969; Rothschild and Stiglitz 1970, 1971). It is often used in the financial markets to evaluate performance of one portfolio against another. In this paper, we use it to evaluate one software allocation "portfolio" versus another.

Stochastic Dominance: The Two-Node Case

Without loss of generality, we start with a simple two-node case to study the advantages of software diversification because closed-form solutions can be derived, allowing us to get a better understanding of the factors that impact the benefits of diversification. A lot of insights can also be derived in this simple case.

We first show the conditions when the node failure distribution under diversity second order stochastically dominates that under homogeneity. If the node failure distribution F second order stochastically dominates a distribution G, then for every concave utility function $u(x)$,

$$\int_0^n u(x)dF(x) \geq \int_0^n u(x)dG(x) \tag{17}$$

This implies that risk-averse firms would choose, from the set of possible designs for their networked information systems, a design for which loss distribution second order stochastically dominates the loss distributions associated with all other designs. Properties of risk aversion and second order stochastic dominance are discussed in detail by Hadar and Russell (1969) and Rothschild and Stiglitz (1970, 1971).

In a simple two node network, there are two possible software allocations: either both nodes share the same software (homogeneity) or they have different software (diversity). An attack on a two-node network can have three possible outcomes: zero failures, one failure, or two failures, as illustrated in Figure 9.

The failure distribution under homogeneity (software configuration 1 deployed on all nodes) can be written as

$$P_H(0F) = \left[\frac{1-c}{1+m-c} + \frac{c}{1+m-c}\right] \cdot b_2(0) + \frac{m-c}{1+m-c};$$

$b_N(y)$: beta–binomial distribution

$$P_H(1F) = \left[\frac{1-c}{1+m-c} + \frac{c}{1+m-c}\right] \cdot b_2(1)$$

$$P_H(2F) = \left[\frac{1-c}{1+m-c} + \frac{c}{1+m-c}\right] \cdot b_2(2)$$

The failure distribution under diversity can be written as

$$P_D(0F) = \left[\frac{1-c}{1+m-c} + \frac{m-c}{1+m-c}\right] \cdot (1-\pi) + \frac{c}{1+m-c} \times b_2(0)$$

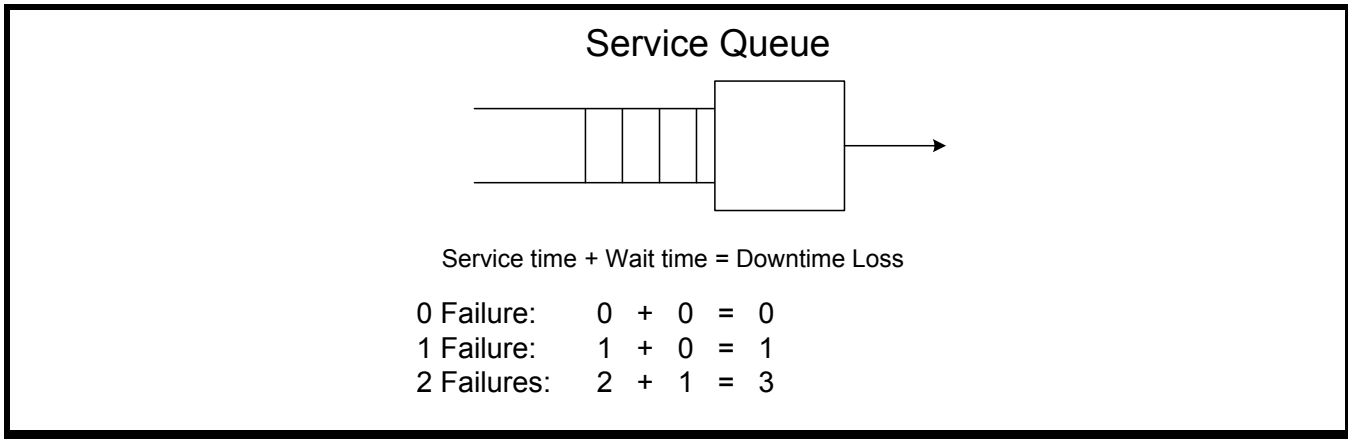


Figure 10. Loss as a Function of Node Failure

$$P_D(1F) = \left[\frac{1-c}{1+m-c} + \frac{m-c}{1+m-c}\right] \cdot \pi + \frac{c}{1+m-c} \times b_2(1)$$

$$P_D(2F) = 0 + \frac{c}{1+m-c} \cdot b_2(2)$$

Where $P_H(2F)$ indicates probability of two node failures under homogeneity, and π, as mentioned before, is the unconditional probability of node failure under an attack.

Figure 10 shows a simple downtime loss function that accounts for loss in availability due to service plus waiting time.

> **Proposition 1**. *When two software (configurations) have comparable attack rates such that*
>
> $$\frac{1+c(\pi+\rho-\pi\rho)}{1+\pi+\rho-\pi\rho} < m < c + (1-c)(1+\pi+\rho-\pi\rho),$$
>
> *software diversity second order stochastically dominates software homogeneity.*

Recall that m measures the relative attack rate of software 2 to that of software 1, which is normalized to 1. Proposition 1 shows the range of m when it is advantageous to diversify software (i.e., use both software) rather than using just the more secure software that receives fewer attack (min(1,m)). An interesting observation is that the upper bound in Proposition 1 is always greater than 1 unless $c = 1$ ($c = 1$ implies that all attacks software 1 receives are common attacks, i.e., these attacks also target software 2). This is interesting as it suggests that even if software 2 is less secure (i.e., when $m > 1$), the firm can still reduce security loss by employing both software configurations. This clearly demonstrates the benefit of diversification: *having both software configurations in place is more beneficial than employing only the more secure software configuration.*

Another interesting observation from Proposition 1 is that a broader range of m suggests that the optimal software strategy to reduce downtime loss is more likely to involve both software configurations. By studying how the range of m changes with c, π, and ρ, we can also get some insights on how the benefits of diversification change with c, π, and ρ.

> **Proposition 2.** *The range of m decreases in c and increases in π and ρ, suggesting that diversification becomes more attractive as the number of shared attacks (c) decreases, the probability of failure (π) increases, and the correlation of failure (ρ) increases.*

While it is intuitive to show that diversification is more advantageous when the different software configurations employed receive fewer common attacks, it is interesting to note from Proposition 2 that diversification is especially advantageous when other security measures are imperfect (i.e., when a firm faces higher π and/or ρ for whatever reasons). In addition, as noted before, certain software vulnerabilities may lead to higher failure rates, and high connectivity of nodes can also lead to high correlation of failure. Therefore, software diversification can also be advantageous in these cases.

In a two-node network, we have shown analytically when firms would prefer diversity. We next present a more general framework for software diversification.

Loss Reduction via Diversity

As noted before, a firm can reduce its security loss by manipulating one or more of the factors that link to security loss: rate of attacks λ, probability of node failure π, correlation of failure across nodes ρ, and service time d. In this section, we will show how diversity can reduce security loss by reducing ρ (the costs of diversification will be discussed in the next section).

Consider a firm that decides to diversify its software in order to reduce the chances of simultaneous failure of multiple computers. It may do so by keeping x_1 proportion of its computers on software configuration 1 while switching to competing software configuration 2 for the remainder $(1 - x_1)$. Given a diverse software deployment, the distribution of the total number of computers affected in an attack can be calculated as per the previous section, which, in conjunction with the formulation also presented in that section, can be used to determine the expected loss faced by the firm:

$$DT(x) = d\left[\frac{1 + Bat(x) - \theta(x)}{2(1-\theta(x))}\right] \times \gamma(x) \tag{18}$$

Where

$$Bat(x) = \frac{E\left[Y^2 \middle/ X = x_1\right]}{E\left[Y \middle/ X = x_1\right]},$$

$$\theta(x) = \lambda \cdot d \cdot E\left[Y \middle/ X = x_1\right], \quad \gamma(x) = \lambda \cdot E\left[Y \middle/ X = x_1\right]$$

Notice that diversification increases the total number of attacks but reduces the exposure per attack as only a subset of nodes are vulnerable to any attack. Substituting expressions for *Bat(x)*, $\theta(x)$, and $\gamma(x)$ in (18) we get (please see Appendix A for derivation of *E[Y]* and *E[Y²]*)

$$\begin{aligned}
DT(x_1) = {} & d\lambda\left(\frac{(1-c)Nx_1\pi}{1+m-c} + \frac{(m-c)N(1-x_1)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right) \\
& \left[1 + \frac{(1-c)Nx_1\pi(1-\pi)\rho(\rho^{-1} - 1 + Nx_1}{1+m-c}\right] \\
& + \frac{(m-c)N(1-x_1)\pi(1-\pi)\rho\left(\rho^{-1} - 1 + N(1-x_1)\right)}{1+m-c} \\
& + \frac{cN\pi(1-\pi)\rho(\rho^{-1} - 1 + N)}{1+m-c} \\
& + (1-c)\left(Nx_1\pi - \frac{(1-c)Nx_1\pi}{1+m-c} - \frac{(m-c)N(1-x_1)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 \\
& (1+m-c)^{-1} \\
& + (m-c)\left(N(1-x_1)\pi - \frac{(1-c)Nx_1\pi}{1+m-c} - \frac{(m-c)N(1-x_1)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 \\
& (1+m-c)^{-1} \\
& + c\left((N\pi - \frac{(1-c)Nx_1\pi}{1+m-c} - \frac{(m-c)N(1-x_1)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 \\
& (1+m-c)^{-1} \\
& + \left(\frac{(1-c)Nx_1\pi}{1+m-c} + \frac{(m-c)N(1-x_1)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right)^2 \\
& \left(\frac{(1-c)Nx_1\pi}{1+m-c} + \frac{(m-c)N(1-x_1)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right)^{-1} \\
& - d\lambda\left(\frac{(1-c)Nx_1\pi}{1+m-c} + \frac{(m-c)N(1-x_1)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right) \\
& \left(2 - 2d\lambda\left(\frac{(1-c)Nx_1\pi}{1+m-c} + \frac{(m-c)N(1-x_1)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right)\right)^{-1}
\end{aligned} \tag{19}$$

The firms can reduce downtime loss by either investing in reducing probability of failure, π, given an attack, by deploying security technologies such as firewalls and anti-virus software; by introducing diversity, x_1, in their network to decrease the number of shared vulnerabilities and therefore correlation of failure on the network; or by reducing service time, d, to repair computers affected in an attack. The first two approaches alter the node failure distribution while the latter approach alters repair rate. The effectiveness of the diversification strategy to reduce node failure distribution depends on the exogenous factors of how likely each software configuration is to be under attack (λ and $m\lambda$) and how many attacks are shared among the software (c). Optimal choices of x, π, and d can be solved by minimizing *DT* with respect to x, π, and d subject to the relative security level of alternative software and the budget constraint on those:

$$\begin{aligned}
&Min\ Downtime = min_{z,x,d}\, DT(x, \pi, d, \rho, m, c, \lambda); \\
&k_1 x + k_2 \pi + k_3 d \le k_4
\end{aligned} \tag{20}$$

Where, k_1 to k_3 are the per unit direct costs of investment in diversity, security technology, and repair capacity, while k_4 is the total technology budget.

Equation (20) highlights the relevant factors that a firm has to consider when deciding its security strategy, including deciding the level of diversity. This framework allows us to compare the relative effectiveness of software diversity in reducing expected loss with that of security technologies and service capacity, which we will discuss next. We address the organizational costs of diversity in the section on maximizing net benefit.

Applications

As noted earlier, a firm may invest in security technologies, such as antivirus, firewall, and intrusion detection systems, which help in detecting and curtailing attacks. A firm may also invest in service capacity, which helps in its response to attacks by bringing failed nodes/systems back to work faster. Using the framework that we have developed, we compare the relative effectiveness of software diversity in reducing expected loss with that of security technologies and service capacity. First we consider how downtime loss varies with the extent of diversity under different service capacity levels, d (Figure 11).

Investment in Diversity Versus Investment in Response

As is evident from Figure 11, expected loss is reduced when firms deploy different software configurations that share minimal vulnerabilities. As shown earlier, such diversification strategy leads to lower correlation, and thus the extent of failure (i.e., the variance of computers failing per incident) are reduced. This is because under most attacks only a subset of computers are at risk (those running the software that is vulnerable to that attack). Consider a two-software configuration example as illustrated by Figure 11 where software configuration 2 is assumed to receive more attacks than software configuration 1 (m = 1.5). Even though software configuration 2 is less "secure" than software configuration 1, an optimal strategy to reduce security loss would involve deploying both software configurations in the network, and the optimal x_1 is greater than 0.5 (i.e., the more-secure software configuration 1 should be deployed on more computers). This suggests that deploying software configuration 2 together with software configuration 1 would result in lower overall security loss than running all nodes on the more-secure software configuration 1. This example echoes what we found in Proposition 1 and clearly demonstrates the benefits of diversity: even though the firm may face more attacks by also employing the less secure software configuration in its computing infrastructure (as opposed to just the most secure one), the consequence of each attack would be of smaller magnitude and more manageable. By reducing the variance of attack magnitudes, the firm greatly reduces the possibility of drastic loss.

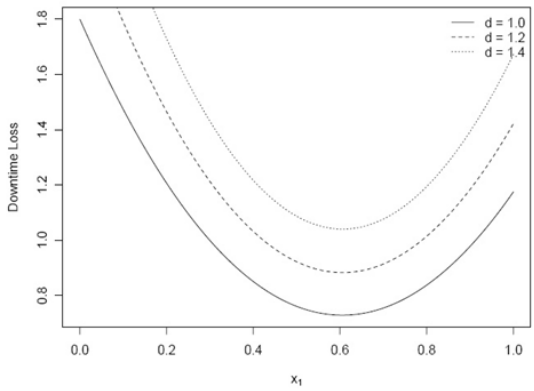
Figure 11 also shows that the magnitude of loss reduction from diversification alone decreases with increase in service capacity (i.e., decreasing d) as systems affected in either type of attack are serviced faster, suggesting that the diversification strategy becomes less attractive when a firm's service capacity is large or when repair time is short. However, it is important to note that surplus service manpower (i.e., lower d) comes with an organizational cost. From the budget allocation point of view, the optimal investment in service capacity is when the marginal benefit of a dollar spent on service capacity equals the marginal benefit from a dollar spent on software diversification.

Investment in Diversity Versus Investment in Detection

We next compare the relative effectiveness of diversification in reducing downtime loss *vis-à-vis* the effectiveness of basic security technologies (Figure 12). Once again, we see that expected loss is reduced under a diversification strategy. A firm can also reduce its expected loss by reducing the probability of computer failure (π) by investing in better protection technologies such as firewalls, antivirus software, etc. We can also observe that the magnitude of loss reduction from diversification decreases with decreases in π, suggesting that diversification becomes less attractive when other security measures are more effective. This again demonstrates what we found in Proposition 2. The optimal choice can be determined by comparing the marginal cost and marginal benefit of each approach. As noted earlier, the costs of diversity may include training and integration costs, while more strict security policies or security measures, such as firewalls or intrusion detection systems, are often associated with limited flexibility and thus a possible decrease in user efficiency. Given that different organizations have different levels of sophistication in technology use, the right amount of security investment would be organization and industry specific.

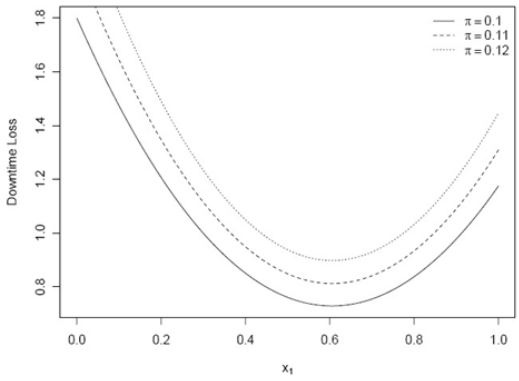
We would like to highlight an interesting observation from Figures 11 and 12: the optimal software allocation of the diversification strategy is insensitive to d and π; this is due to the fact that m and c (i.e., the relative security level of alternative software and the number of common attacks across software) and the relative costs of diversity to that of d and π do not change. While the level of d and π does influence the marginal benefits of the diversification strategy (as shown in Figures 11 and 12), they do not affect the optimal form of the diversification strategy (this is evident from the same shape of the curves in Figures 11 and 12, which show that the curves of downtime loss as a function of x are of the same shape under different d and π). The optimal allocation among alternative software under the diversification strategy depends only on m and c and the unit costs of x.

However, even though d and π do not change the shapes of the curves, they do influence the decision of whether the diver-



Expected downtime loss as a function of x1 for varying levels of service capacity (smaller d means higher capacity). m is kept constant at 1.5 (i.e., software 2 receives 50% more attacks than software 1); c = 0; λ = 0.01; π = 0.1; ρ = 0.5; N = 100.

Figure 11. Variations of Downtime Loss



Expected security loss as a function of x1 for varying levels of probability of failure π (π1 = π2). m is kept constant at 1.5 (i.e., software 2 receives 50% more attacks than software 1); c = 0; λ = 0.01; d = 1; ρ = 0.5; N = 100.

Figure 12. Expected Security Loss

sification strategy should be adopted since they influence the marginal benefits of diversification strategy. When both *d* and π are low enough, we can show that the optimal strategy involves no diversification; otherwise, when the diversification strategy is adopted, it always employs the same optimal allocation as long as *m* and *c* and the unit costs of *x* do not change.

So far, we have considered mainly the benefits of diversification and the optimal level of diversification when the objective function is to reduce expected downtime loss. However, in reality, firms may hesitate to mix different software applications due to the concerns of compatibility as well as the costs associated with managing a heterogeneous environment. While the trend in industry standardization and the development of the web services architecture has pushed toward compatibility, the concerns still remain. In general, no two diverse software applications are completely interchangeable in the sense that the functionalities provided by and the set of applications that work with a particular software application are not exactly the same. Furthermore, software choice depends on past experience and support, both of which have direct and indirect network externalities with the external world (i.e., if the rest of the world uses Windows, then it is easier to learn about and support Windows systems). In the next section, we consider an extended model where the

firm's concern is not only to minimize the expected security loss but also the potential costs associated with software diversification.

Software Diversification to Maximize Net Benefit to the Firm

We have, thus far, shown that diversification can be an effective way to reduce security loss. However, this benefit does not come without costs. Diversification is a costly process. First, the total cost of ownership is higher. There is the cost of supplementary software acquisition and possibly switching costs. It also involves training and maintaining diverse systems, resulting in potentially higher maintenance costs. Second, the benefit of full compatibility and interoperability may be hard to achieve with diverse software when two software technologies are not perfectly compatible. When compatibility and interoperability is a major concern, the sunk costs of past investments may prevent firms from fully executing the diversity strategy.

Altogether, there are essentially three types of costs firms have to balance in deciding the level of diversity: security costs due to repair and downtime, loss due to network effects from lack of homogenization, and maintenance costs for all computers within the firm, which may include routine check-up, upgrades, backups, installing updates, etc. Security costs were discussed earlier. In this section, we will discuss the costs due to possible loss of network effects and economies of scale in maintaining heterogeneous software. We will also explore the conditions under which diversity is preferred over homogeneity by simultaneously considering the security loss, network effects arising from compatibility, and the additional support costs associated with software diversification.

Network Effects from Compatibility

Consider a group of N nodes, which can directly exchange information or communicate with each other. Suppose there is value to each potential direct interaction, then the benefit of running the same software on all nodes is of the order N^2 since there are a total of $N * (N - 1)$ possible interactions (Shapiro and Varian 1999). To model the network effect benefit of software, we choose this standard N^2 benefit function. In the case of two software configurations that are not perfectly interoperable, we define a standardization index $b \in [0, 1]$, which captures the value of interaction between two nodes running diverse software relative to their running homogeneous software. A higher b implies that diverse software have a standardized interface and thus are interoperable with each other. With this setup, we can define the level of network effect benefits achievable from running two software configurations with standardization index b, as

$$NE(x) = C_{ne} * (N_1^2 + N_2^2 + 2bN_1N_2) \quad (21)$$

where

- N_1 = $N * x1$, number of nodes running software configuration/stack 1
- N_2 = $N * (1 - x1)$, number of nodes running software configuration/stack 2
- N = total number of nodes in network
- x_1 = the proportion of nodes running software configuration/stack 1
- C_{ne} = scaling constant to compare the magnitude of network effects with that of security loss

Notice that for $b \approx 0$, we have two disconnected networks, and for $b \approx 1$ we have one large network with seamless connectivity such that there is no loss of compatibility even when we are running two different software configurations. In practice, different classes of software satisfy varying degree of standardization. For instance, the web services architecture has been fairly standardized as a result of concerted industry initiative, for example, webmail ($b \approx 1$), whereas some document file formats are largely proprietary, for example, MS word ($b \approx 0$). In practice, we may measure b by the relative costs required to make two software applications communicate or exchange files seamlessly. In the case that two software applications are perfectly compatible, and no extra cost is needed to make them communicate, then b would be 1. The higher the costs, the lower the standardization index b. For example, if the cost required to make two software applications communicate or exchange files is $c1$ and the cost for making two incompatible software applications communicate is $c2$, then a simple way to measure b would be $(1 - c1 / c2)$. The point is that b should capture the value of software applications being able to communicate/interact directly with each other.

Given the strong industry trend toward standardization with the advent of HTTP, XML, SOAP, AJAX, etc., and the development of technological solutions that aim to overcome the limitations of interoperability (e.g VMware,[17] a popular virtual infrastructure software that emulates multiple operating systems on a single computer), we might observe decreasing barriers to diversification in the future.

[17] http://www.vmware.com.

Cost of Maintenance

As noted earlier, there are economies of scale in maintaining homogeneous systems, such as training, routine checkup, upgrades, backups, installing updates, etc. Therefore, firms may face additional support cost from maintaining a diversified software environment. This additional cost is due to the loss in economies of scale in acquiring and supporting diverse software. Without loss of generality, we model the economies of scale in maintaining software using a concave cost function. To keep our model tractable with minimum parameters, we consider the simple and widely used log cost function. The insights, however, are generalizable to other concave cost functions.

$$MC(x) = C_{mc} * \big(log(N_1 + 1) + log(N_2 + 1) - log(S+1)\big) \quad (22)^{18}$$

where

$MC(x)$ = cost of maintaining x level of diversity for $0 < x < 1$

C_{mc} = scaling constant to compare the magnitude of cost of maintenance with that of security loss

S = the common components shared by the two software

Optimal Diversification Strategy

As noted earlier, the potential costs of diversity may come from two sources: loss of compatibility and increases in maintenance costs. The optimal diversity in the network can be determined by maximizing the net benefit taking into account network effects, cost of maintenance and downtime loss (from the previous section on reducing downtime costs):

$$MeanNetBenefit(x) = NE(x) - MC(x) - DT(x) \quad (23)$$

We identify the conditions under which diversity is preferred over homogeneity by maximizing the net benefit with respect to the level of diversity x. Figure 13 illustrates the conditions when diversity is preferred over homogeneity, that is, $x^* \in (0,1)$. For each point on the graph we compute whether the net benefit is greater with diversification or without diversification; the shaded regions indicate the conditions where diversity would be preferred, while the white region indicates conditions suitable for homogeneity. The graph is a contour plot, and we summarize the findings below.

Observation 2:

(1) Diversification becomes more attractive as the standardization index b increases (i.e., as different software get standardized interfaces through XML, web services, etc.).

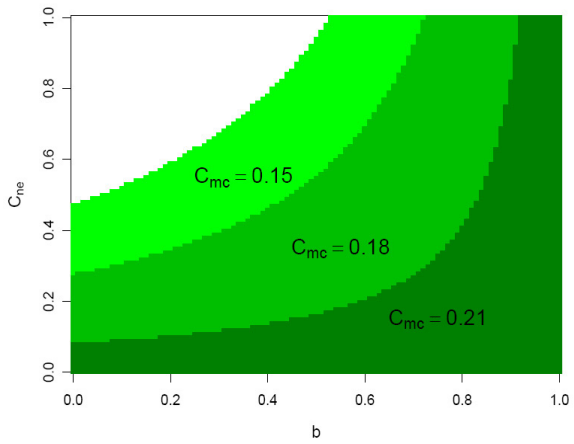
(2) Diversification becomes more attractive as the cost to ensure compatibility and interoperability (C_{ne}) decreases, or when the requirement for homogeneous deployment to gain the benefit of network effect diminishes.[19] C_{ne} is likely to be low for software applications where adapters or middleware are easily accessible, and for organizations that are not highly networked i.e., network effects are mild (low C_{ne}).

(3) Diversification is more attractive as the cost of supporting and maintaining diverse systems decreases (C_{mc}). This is evident in Figure 13 that as the cost of maintaining diverse systems decreases (indicated by decreasing C_{mc}), the region favorable to diversity expands (shown via lighter shades). Firms that are sophisticated in IT use or have good IT support infrastructure will have relatively low C_{mc}, and can therefore benefit more from diversity. We should also note that C_{mc} is application dependent too. Certain applications, such as ERP systems, may have high C_{mc} due to high acquisition and maintenance costs and, therefore, it may be more cost effective to invest in other security technologies and/or faster response.

Overall, we show that software diversification is a viable technology investment strategy under general conditions, which to date has been overlooked in the IS and economics network effects/standardization literature. In the following subsection, using real world data on vulnerabilities in common software applications such as Internet Explorer, Firefox web browser, and Outlook, we demonstrate the use of our framework in calculating the optimal software allocation for any network.

[18] We add +1 to the log function to make sure the log function does not become ill-defined when the element becomes zero. In addition, we expect the costs of managing two different software configurations to be lower when they share more components than when they share few components. S can be highly correlated to the amount of shared vulnerabilities between the two software configurations, although they are not necessarily the same because the security of each component may differ.

[19] As is evident in Figure 13, for low values of C_{ne}, diversification is a preferred strategy under a larger range of values for b and C_{mc} (i.e., more shaded region around low C_{ne}).



Dark areas indicate the region where the net benefit is higher with diversification; the color fades (diversification region expands) as costs of supporting diverse systems goes down; x-axis is standardization index b and y-axis is normalized scaling constant for network effects; N = 100; π = .1; ρ = .5; m = 1; c = 0; λ = .01; d = 1.

Figure 13. Conditions When Diversity Is Preferred

Application: Software Allocation

Our analyses have demonstrated the benefits of diversification to an organization. An interesting question that follows is how a firm can optimally allocate software to achieve the benefits of diversification. Specially, considering the configuration–vulnerability matrix presented in Figure 3c, should a firm consider deploying more than one software configuration? If yes, how should a firm allocate these different configurations to nodes in a network? For example, a question that has received a lot of attention recently is: Should firms switch their web browsers from Microsoft's Internet Explorer to Mozilla's Firefox? Firefox has gained tremendous publicity for being more secure than Internet Explorer. There are similar concerns regarding Microsoft's Outlook and the Windows operating system. We explore the options for software allocations for IT managers and how might they go about making an informed choice using a case example.

We define software allocation as the deployment of software configurations across nodes by assigning software configurations to nodes such that all nodes have the required set of applications such as e-mail, editor, and web browser. Specifically, if we consider the case of two software categories—web browser and e-mail client—the requirement is that each feasible allocation (i.e., software configuration) should consist of software from each of these categories.

Given software applications like Internet Explorer (IE), Firefox (FF), Outlook (OL), and Thunderbird (TB), a firm can choose from four possible configurations for its nodes viz. C1 (IE + OL), C2 (IE + TB), C3 (FF + OL), and C4 (FF + TB). Note that there are a total of $C(N + M - 1, M - 1)$ many ways of assigning M configurations to N nodes;[20] for $M = 4$ and $N = 10$, there are 286 possible allocations. However, considering positive network externalities alone and disregarding the risk posed by correlated failure, firms have usually chosen C1 for all of their N nodes. In our work, we challenge that traditional approach and consider all possible allocations to find the optimal one.

The mean net benefit for an allocation a_k (k ranges from 1 to 286) can be written as

$$MeanNetBenefit(a_k) = NE(a_k) - MC(a_k) - DT(a_k) \quad (24)$$

The above formulation can be used to estimate the mean net benefit associated with each allocation, which can then be rank ordered to calculate the best allocation. However, as mentioned before, due to the correlated nature of node failure, it is not sufficient to calculate expected security loss just as a function of mean failure of nodes. Our calculation of security

[20]The implicit assumption here is that each node is symmetric. If the nodes are asymmetric, for example, due to the way they are connected, then there would be M^N possible allocations.

loss built on the $M^X/G/1$ queuing system takes into account not only the mean node failure but also the variance of nodes failure (which is captured by the endogenous correlation of failure in our model). An important goal for many organizations is to ensure business or operation continuity even in unfavorable circumstances, which would require developing bounds such that even in ε-worst case (ε-worst is defined as the event for which the probability of it happening is below ε), the proportion of node failures is no greater than a threshold (t). Consider this example: if important business data is replicated over two computers, then the firm can recover from the failure of one computer. However, if both nodes fail simultaneously, the loss would be catastrophic. Therefore, a more practical allocation problem is defined as follows:

$$\max_{\alpha_k} MeanNetBenefit(\alpha_k); \quad \alpha_k = (x_{1k},\ldots,x_{Mk}), \qquad \sum_{i=1}^{M} x_{ik} = 1 \tag{25}$$

subject to the constraint:

$$P(Y > t \cdot N) < \varepsilon$$

where x_1 through x_M are the proportion of nodes assigned configurations 1 through M, Y is the number of node failures in an attack incident, and N is the total number of nodes on the network.

Given the number of vulnerabilities in IE, FF, OL, and TB (Figure 14) and the rate of attacks against them, we calculate the joint susceptibility of attacks for C1 through C4 (Figure 15). In our analysis, we assume that owing to their popularity Microsoft software (IE and OL) receive m times as many attacks as FF and TB.[21] We then calculate the correlation of failure for all pairs of nodes for all possible allocation scenarios[22] and generate the node failure distributions. We find that the allocation that has the highest mean net benefit is ($x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 1$), however, this does not satisfy the above failure constraint for $t = 0.7$ and $\varepsilon = 0.05$. The allocation that has the highest mean net benefit subject to that constraint is ($x_1 = 0, x_2 = 0, x_3 = 0.2, x_4 = 0.8$). We compare allocations for $t = 0.7$, as it appears to be a reasonable reference point. As discussed next, other threshold values can be considered, depending on organizational goals. In Figure 16, we compare the failure distribution associated with these allocations with the failure distribution when all nodes are assigned the default configuration C1. Note that, the rightmost distribution is unambiguously unfavorable since it has a very high chance of simultaneous failures. Also note that between the leftmost and central distributions, the central one has a fatter right tail, and therefore is less ideal than the leftmost one.

We calculate the mean net benefit for all possible allocation scenarios such that an *efficient frontier* can be found for determining the maximum mean benefit that is possible under different failure constraints (Figure 17). The goal of IT managers should be to achieve an allocation such that the benefit lies on the efficient frontier. However, it is understandable that due to continuously varying vulnerability and attack conditions, the optimal allocation drifts away from the efficient frontier. We next discuss how organizations can cope with these changing vulnerabilities.

Monitoring Drift

Since software vulnerabilities change over time (old ones are patched while new ones are discovered), optimal allocation will not remain the same. For example, while Firefox has been considered more secure, recently it has been reported to be the most vulnerable browser due to a high number of vulnerabilities.[23] When new vulnerabilities are suddenly discovered in the Firefox browser, as an example, such that entry in cell(2,2) in Figure 14 changes from 17 to 50, then clearly the optimal allocation will change accordingly. We define this phenomenon as the drift which occurs due to change in external conditions such as the number of shared vulnerabilities and/or attack rates affecting them. Figure 17 shows how the location of previous-best allocation changes under the new set of conditions and a new efficient frontier. We believe that the concept of drift could be useful for IT managers in monitoring the state of their networks. In addition, based on our approach, they can develop policies to periodically review the drift so that software can be reassigned to nodes whenever the magnitude of the rectilinear drift from the efficient frontier is high *vis-à-vis* cost of reallocation. The rectilinear drift can be measured as either the horizontal distance or the vertical distance from the efficient frontier. Horizontal distance signifies compromise in failure

[21]Therefore, total number of attacks = 28m (attacks on IE) + 16 (attacks on FF, not included in IE) + 2m (attacks on OL) + 0 (attacks on TB, not included in FF).

[22]In addition to susceptibility to attacks, correlation also depends on individual actions taken at nodes like user specified software settings, additional precautions etc. To account for that we scale the correlation calculated by the formula for ρij by 0.5.

[23]http://news.cnet.com/8301-1009_3-10190206-83.html.

	IE	FF	OL	TB
Internet Explorer (IE)	28	1	0	0
Firefox (FF)	1	17	0	2
Outlook (OL)	0	0	2	0
Thunderbird (TB)	0	2	0	2

The vulnerabilities were retrieved from the National Vulnerability Database for the period of April–June 2007.

Figure 14. Number of Shared Vulnerabilities between a Set of Software Applications

	C1	C2	C3	C4
C1 (IE+OL)	$\frac{28*m+2*m}{28*m+16+2*m}$	$\frac{28*m}{28*m+16+2*m}$	$\frac{1*m+2*m}{28*m+16+2*m}$	$\frac{1*m}{28*m+16+2*m}$
C2 (IE+TB)	$\frac{28*m}{28*m+16+2*m}$	$\frac{28*m+2}{28*m+16+2*m}$	$\frac{1*m+2}{28*m+16+2*m}$	$\frac{1*m+2}{28*m+16+2*m}$
C3 (FF+OL)	$\frac{1*m+2*m}{28*m+16+2*m}$	$\frac{1*m+2}{28*m+16+2*m}$	$\frac{16+1*m+2*m}{28*m+16+2*m}$	$\frac{16+1*m}{28*m+16+2*m}$
C4 (FF+TB)	$\frac{1*m}{28*m+16+2*m}$	$\frac{1*m+2}{28*m+16+2*m}$	$\frac{16+1*m}{28*m+16+2*m}$	$\frac{16+1*m}{28*m+16+2*m}$

Joint susceptibility of attacks for configurations C1 through C4, which can be derived from the configuration–vulnerability matrix shown in Figure 3c.

Figure 15. Joint Susceptibility of Attacks

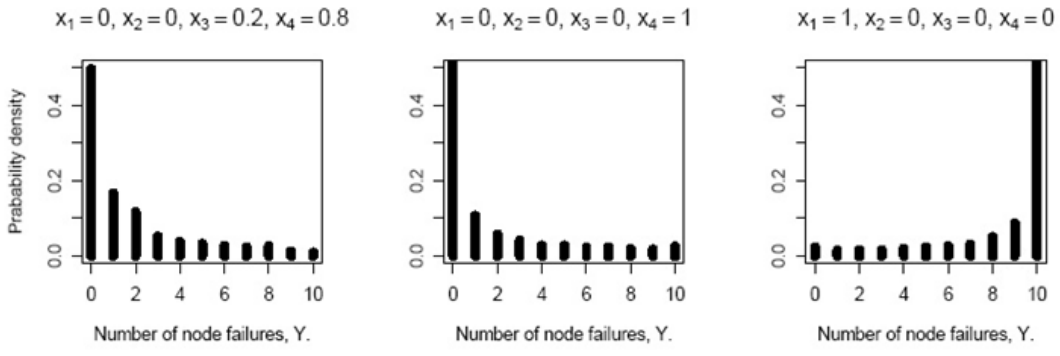
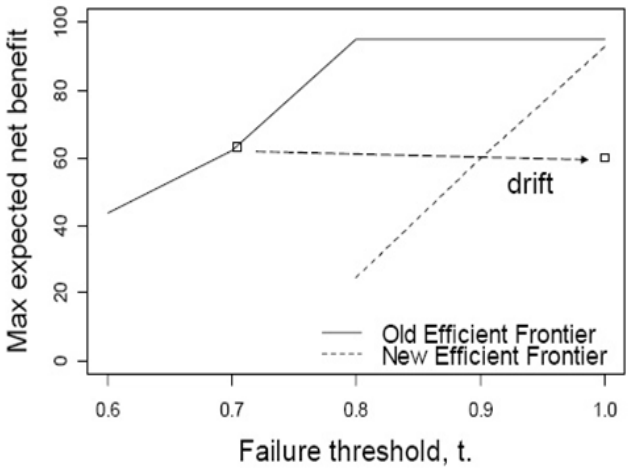


Figure 16 Failure Distributions Associated with Various Allocation Possibilities (m = 2; N = 10)



Highest achievable benefit under various failure constraints; m = 2; ε = 0.05; N = 10; C_{ne} = 1; C_{mc} = 1; b = 0; a = 0.01; d = 1.

Figure 17. Efficient Frontier

threshold whereas vertical distance signifies shortfall in maximum achievable benefit under that threshold. The appropriate weights can be assigned for them by the IT managers after taking into consideration organization-specific goals.

Discussion and Conclusion

Software vulnerabilities when exploited cause loss in confidentiality, integrity, and availability of information. Various encryption and encoding techniques have been proposed to protect confidentiality and integrity of information, while redundancy and fault tolerance have been the classical solutions to ensure availability. However, in a network environment, when a vulnerability common to all nodes is exploited, redundancy alone is not enough to ensure availability. In this paper, we argue that an alternative solution to maintain availability is to spread the risk (i.e., to diversify the network such that all nodes do not share the same vulnerabilities).

The objective of this paper is to quantify security loss due to unavailability and to examine whether a firm can benefit from maintaining a diversity of systems or software taking into account the benefits and costs of a diversification strategy. Unavailability may result from node/system failure, repair, and/or checkup. We introduce system downtime as a suitable metric to quantify security loss and develop a formulation based on queuing theory to quantify downtime loss due to unavailability of nodes. We present a comprehensive attack–failure–repair model to evaluate the impact of software diversity and develop a formulation for calculating failure distribution based on the correlation matrix. The novelty of this model is that we endogenize the failure distribution and the node correlation distribution, and show how the diversification strategy and other security measures may impact these two distributions, which in turn determine the expected security loss faced by the firm. In addition, our proposed model incorporates the three levels of security: prevention, detection, and response. As a result, it not only allows us to compare the benefits of diversification *vis-à-vis* investments in response or other security measures such as firewalls, antivirus software, intrusion detection systems, and security policies, but also enables us to study how diversification can be used with these security instruments to reduce security loss. While the reduction in downtime brought by diversity is a compelling consideration, we note that diversity has significant costs associated with it in the IS context due to the potential loss of network effects and the lack of economies of scale. Therefore, determining the optimal level of diversity and conditions when diversity is superior are non-trivial. We present a framework to guide optimal investment in software diversification taking into account these benefits and costs. We show that diversification is appealing under rather general conditions.

Diversity has often been ignored in the IS and economics network effects literature due to over-emphasis on network effects. Our analyses contribute to this literature by showing

how diversity can benefit firms and developing a framework and several tools to guide the technology investment decisions of firms. This research also contributes to the risk analysis literature in addition to the information security literature by showing that diversity is advantageous not only to a risk-averse firm but also to a risk-neutral firm that cares about minimizing mean downtime. A risk-averse firm that also cares about the variance of downtime would attach even greater weight to downtime loss when determining the optimal diversity. Our finding that diversification benefits firms has an important policy implication as well. In particular, it suggests that the government should provide an environment where multiple software/system configurations can coexist. While we do not study socially optimal diversification in our paper directly, our finding does suggest that society as a whole will benefit from a diversification strategy by spreading the risks and reducing drastic risks and loss.

In addition to contributing to the information security and risk management literature, this paper also makes several practical contributions. First of all, we propose managing security loss through managing shared vulnerability among systems. We propose that we can better preserve the availability of system functionality through diversity, consistent with the idea of "functionality defense by heterogeneity" by Sharman et al. (2004). We introduce the concepts of the configuration–vulnerability matrix, the (node) vulnerability matrix, and the node failure correlation matrix, which can be used as risk awareness tools that can help organizations better manage their security risks (Straub et al. 2008). Furthermore, our proposed framework is holistic and practice oriented and incorporates other security measures and practical concerns facing a firm (such as costs to maintain a diverse environment and a minimum availability guarantee), and can help a firm develop its software acquisition/allocation strategy. Our analysis also suggests that diversity can make other security measures more effective.

Our formulation is not particularly dependent on the functional form of the attack process or the service routine. Our assumption of Poisson arrival for attacks is a nonrestrictive one, because if the attacks were not independent, then it would lead to further buildup in the service queue causing even higher downtime, which would further strengthen our argument for diversity as a means to avoid queue buildup. The use of $M^X/G/1$ formulation, though, has a minor limitation: it assumes that whenever an attack incident occurs, the entire network is exposed to it; however, because some nodes may be in repair queue at the time of attack, they may not be susceptible to those attacks. Practically, nodes will be in a functional state most of the time; therefore, the above concern does not pose a risk in most cases. Our framework is general and applies to any scenario where loss is convex in the number of failed nodes. Our analysis indicates that as technology markets trend toward more standardized environments, the barriers to diversity become less of a concern, making diversity more acceptable. Similarly, diversity is preferred when the cost of supporting diverse systems declines.

However, this paper suffers from some limitations. First of all, even though we believe that the assumptions of single server and sequential repair are not material to the insights that are gained from our model, future research may relax these assumptions to see how sensitive the benefits of diversification are to number of servers and simultaneous repair (although individual attention to each affected computer in order to account for differences in user settings, permissions, data backups, and custom applications is often needed even when computers can be repaired simultaneously). We also did not consider attacker behavior and did not formally account for the dampening effect of legacy systems on new software adoption, or the costs related to redesign the service facility to accommodate diverse software environments.

Despite these limitations, we believe that our framework has contributed to understanding the main tradeoffs of a software diversification strategy, and it can be extended to overcome the aforementioned limitations. In addition, the cost of supporting diverse systems can be highly dependent on the overall service capability of the IT department. Some software configurations may be easy to support while others are more costly to support and may require considerable fixed costs (e.g., hiring a specialist). Under these conditions, the redesign or reorganization of the service capacity becomes a direct function of diversity. Therefore, we believe that the complete cost–benefit analysis is a multivariate constrained optimization problem which, in addition to all the factors discussed in this paper, should also take into account the network redesign issues and service facility reorganization. An interesting future research project would be to develop an IT acquisition and deployment decision support system supported by the formulation of a combinatorial optimization model taking into account the factors and tradeoffs considered in this paper as well as other practical constraints faced by the firm to help guide its system deployment strategy (e.g., how many different software configurations to deploy and how to deploy them). Furthermore, there are many strategic factors that would also be relevant to the design of the decision support system. For example, the vulnerability disclosure policy of the software or third party vendor and the time it takes for patches to be available may be relevant to a firm's choice of vendors/software (Arora et al. 2010). The attack trend of vulnerabilities may differ across firms depending on their patching policies and the channel through which they

acquire the vulnerability information (Kannan and Telang 2005; Li and Rao 2007). Moreover, as markets evolve, the market share of competing software would change, which in turn may affect attack distribution. For example, would Linux have a higher market share in the future and would this change the nature of the attack distribution on Linux? Would Windows or MS-Office be made available in multiple versions with few or no overlapping vulnerabilities in order to preserve the benefits of interoperability while also reducing attack exposure? These are interesting future research questions to explore. Finally, our model deals with the effects of attacks on availability, an important aspect of security cost that is tied closely to a firm's financial loss. However, diversification may also have an impact on confidentiality and integrity of data. When the same data is replicated in two systems that share few common vulnerabilities, it is more difficult for an attacker to attack both systems and, therefore, it becomes easier to detect whether data has been manipulated or not by comparing data residing in both systems. On the other hand, it may become more difficult and costly to protect data from being stolen in a diverse environment because the firm has to protect multiple systems with different vulnerabilities. This seems to suggest that diversification has a negative impact on confidentiality while having a positive impact on integrity of data. However, further research is needed to explore the real implications of diversification on confidentiality and integrity of data.

Acknowledgments

This work was supported in part by grant no. 0433540 from the National Science Foundation.

References

Arora, A., Krishnan, R., Telang, R., and Yang, Y. 2010. "An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure," *Information Systems Research*. (21:1), pp. 115-132.

Avizienis, A. 1995. "The Methodology of n-Version Programming," Chapter 2 in *Software Fault Tolerance*, M. R. Lyu (ed.), New York: Wiley.

Bain, C., Faatz, D., Fayad, A., and Williams, D. 2001. "Diversity as a Defense Strategy in Information Systems," Technical Report, The MITRE Corporation, Bedford, MA.

Bakkaloglu, M., Wylie, J., Wang, C., and Ganger, G. 2002. "On Correlated Failures in Survivable Storage Systems, 2002," Technical Report CMU-CS-02-129, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

Barrantes, E. G., Ackley, D. H., Forrest, S., Palmer, T. S., Stefanovic, D., and Zovi, D. D. 2003. "Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, October 27-30, Washington, DC, pp. 281-289.

Brynjolfsson, E., and Kemerer, C. 1996. "Network Externalities in Microcomputer Software: An Econometric Analysis of the Spreadsheet Market," *Management Science* (42:12), pp. 1627-2647.

Collins, M., Gates, C., and Kataria, G. 2006. "A Model for Opportunistic Network Exploits: The Case of P2P Worms," in *Proceedings of the Fifth Workshop on the Economics of Information Security*, June 26-28, University of Cambridge, UK (available online, http://weis2006.econinfosec.org/docs/16.pdf).

Deswarte, Y., Kanoun, K., and Laprie, J. C. 1998. "Diversity Against Accidental and Deliberate Faults," in *Proceedings of Computer Security, Dependability and Assurance: From Needs to Solutions*, Los Alamitos, CA: IEEE Computer Society, pp. 171-181.

Eckberg, A. 1985. "Approximation for Bursty (and Smoothed) Arrival Queuing Delays Based on Generalized Peakedness," in *Proceedings of the 11th International Teletraffic Congress,* Kyoto.

Economides, N. 2001. "The Microsoft Antitrust Case," *Journal of Industry, Competition and Trade: From Theory to Policy* (1:1), pp. 71-79.

Farrell, J., and Klemperer, P. 2001. "Coordination and Lock-In: Competition with Switching Costs and Network Effects," Chapter 31 in *Handbook of Industrial Organization* (Volume 3), M. Armstrong and R. Porter (eds.), Amsterdam: North Holland.

Geer, D., Bace, R., Gutmann, P., Metzger, P, Pfleeger, C. P., Quarterman, J. S., and Schneier, B. 2003. "Cyber*In*security: The Cost of Monopoly" (available online, http://www.ccianet.org/CCIA/files/ccLibraryFiles/Filename/000000000061/cyberinsecurity.pdf).

Hadar, J., and Russell, W. 1969. "Rules for Ordering Uncertain Prospects," *American Economic Review* (59), pp. 25-34.

Honeynet Project. 2004. "Know Your Enemy: Trends" (available online, http://old.honeynet.org/papers/enemy/).

Howard, J. D. 1997. *An Analysis of Security Incidents on the Internet: 1989–1995*, unpublished Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Kannan, K., and Telang, R. "Market for Software Vulnerabilities? Think Again," *Management Science* (51:5), pp. 726-740.

Katz, M. L., and Shapiro, C. 1985. "Network Externalities, Competition, and Compatibility," *American Economic Review* (75:3), pp. 424-440.

Katz, M. L., and Shapiro, C. 1986. "Technology Adoption in the Presence of Network Externalities," *Journal of Political Economy* (94:4), pp. 822-841.

Katz, M. L., and Shapiro, C. 1994. "Systems Competition and Network Effects," *Journal of Economic Perspectives* (8:2), pp. 93-115.

Kc, G. S., Keromytis, A. D., and Prevelakis, V. 2003. "Countering Code-Injection Attacks with Instruction-Set Randomization," in *Proceedings of the 10th ACM Conference on Computer and*

Communications Security, October 27-30, Washington, DC, pp. 272-280.

Kleinrock, L. 1975. *Queuing Theory* (Volume 1), New York: John Wiley and Sons.

Kunreuther, H., and Heal, G. 2003. "Interdependent Security," *Journal of Risk and Uncertainty* (26:2/3), pp. 231-249.

Li, P., and Rao, H. R. 2007. "An Examination of Private Intermediaries' Roles in Software Vulnerabilities Disclosure," *Information Systems Frontiers* (9:5), pp. 531-539.

Nicola, V. F., and Goyal, A. 1990. "Modeling of Correlated Failures and Community Error Recovery in Multiversion Software," *IEEE Transactions on Software Engineering* (16:3), pp. 350-359.

Ogut, H., Menon, N., and Ragunathan, S. 2005. "Cyber Insurance and IT Security Investment: Impact of Independent Risk," in *Proceedings of the Workshop on the Economics of Information Security (WEIS)*, Harvard University, Cambridge, MA (available online, http://infosecon.net/workshop/pdf/56.pdf)

Park, I., Sharman, R., Rao, H. R., and Upadhyaya, S. 2007. "Short Term and Total Life Impact Analysis of Email Worms in Computer Systems," *Decision Support Systems* (43), pp. 827-841.

Patterson, D. 2002. "A Simple Way to Estimate the Cost of Downtime," unpublished paper, University of California, Berkeley (available online, http://roc.cs.berkeley.edu/projects/downtime).

Rothschild, M., and Stiglitz, J. 1970. "Increasing Risk: I. A Definition," *Journal of Economic Theory* (2), pp. 225-243.

Rothschild, M., and Stiglitz, J. 1971. "Increasing Risk: II. Its Economics Consequences," *Journal of Economic Theory* (3), pp. 66-84.

Shapiro, C., and Varian, H. R. 1999. *Information Rules: A Strategic Guide to the Network Economy*, Boston: Harvard Business School Press.

Sharman, R., Rao, H. R., Upadhyaya, S., Khot, P., Manocha, S., and Ganguly, S. 2004. "Functionality Defense by Heterogeneity: A New Paradigm for Securing Systems," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Los Alamitos, CA: IEEE Computer Society.

Sohraby, K. 1989. "Delay Analysis of a Single Server Queue with Poisson Cluster Arrival Process Arising in ATM Networks," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM), November 27-30, pp. 611-616.

Straub, D., Goodman, S., and Baskerville, R. 2008. "Framing of Information Security Policies and Practices," in *Information Security Policies, Processes, and Practices*, D. Straub, S. Goodman and R. Baskerville (eds.), Armonk, NY: M. E. Sharpe, pp. 5-12.

Symantec Corporation. 2006. "Symantec Internet Security Threat Report, Volume IX," The Symantec Corporation, 20330 Stevens Creek Road, Cupertino CA, 2006.

Varian, H. 1992. *Microeconomic Analysis* (3rd ed.), New York: W. W. Norton & Company.

White, G. B., Dietrich, G., and Goles, T. "Cyber Security Exercises: Testing an Organization's Ability to Prevent, Detect, and Respond to Cyber Security Events," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Los Alamitos, CA: IEEE Computer Society.

About the Authors

Pei-yu Chen is an associate professor of Management Information Systems in the Fox School of Business at Temple University. Prior to this position, she was an assistant professor of Information Systems at the Tepper School of Business, Carnegie Mellon University. She received her Ph.D. from the Wharton School, University of Pennsylvania, in 2002. Her work has been published in *Information Systems Research*, *Management Science*, *MIS Quarterly*, and *Operations Research*. She currently serves on the editorial boards of *Management Science* and *Production and Operations Management*.

Gaurav Kataria is an associate with Booz & Co., an international management consulting firm. He holds a Ph.D. in Management Science from Carnegie Mellon University with a focus on risk management, and a bachelors degree in Electrical Engineering from the Indian Institute of Technology (IIT). His research interest is in the area of economics of information security. Dr. Kataria has published many articles in the areas of insurance strategy, consumer choice modeling, and software asset management. At Booz & Co., he has worked on projects involving post-merger integration, marketing channel effectiveness, and Internet marketing for Fortune 500 clients.

Ramayya Krishnan is Dean of Carnegie Mellon's Heinz College, and the W. W. Cooper and Ruth F. Cooper Professor of Information Systems at Carnegie Mellon University. He is also the faculty chair of the university's Master's of Information Systems Management program (www.mism.cmu.edu).

MIS Quarterly

RESEARCH ARTICLE

CORRELATED FAILURES, DIVERSIFICATION, AND INFORMATION SECURITY RISK MANAGEMENT

Pei-yu Chen
Department of Management Information Systems, Fox School of Business and Management, Temple University,
1801 N. Broad Street, Philadelphia, PA 19122 U.S.A. {pychen@temple.edu}

Gaurav Kataria
Booz & Co., 127 Public Squire, Suite 5300, Cleveland, OH 44114 U.S.A. {gkataria@google.com}

Ramayya Krishnan
School of Information Systems and Management, The Heinz College, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213 U.S.A. {rk2x@cmu.edu}

Appendix A

Proofs

Proof of Observation 1

$$\frac{\partial \rho_{12}}{\partial c} = \frac{1 + m - 2c}{\sqrt{m(m-c)}} + 1/2\,\frac{((c(1\_m - c) = m)m}{\left(m(m-c)\right)^{3/2}}$$

$$= 1/2\,\frac{2m + 1 - 3c}{\sqrt{-m(-m+c)}}$$

$$> 0 \because m > c \;\&\; c < 1$$

Since $\frac{\partial \rho}{\partial c} > 0$, increase in shared vulnerabilities increases correlation of failure. QED.

Proof of Proposition 1

Loss distribution under diversity second order stochastically dominates homogeneity if the cumulative area under its cumulative distribution function (CDF) is lower than under homogeneity (see Figure A1), that is,

$$2 \times P_H(2F) > 2 \times P_D(2F), \textit{ and}$$

$$2 \times P_H(2F) + 1 \times \left[P_H(2F) + P_H(1F)\right] > 2 \times P_D(2F) + 1 \times \left[P_D(2F) + P_D(1F)\right]$$

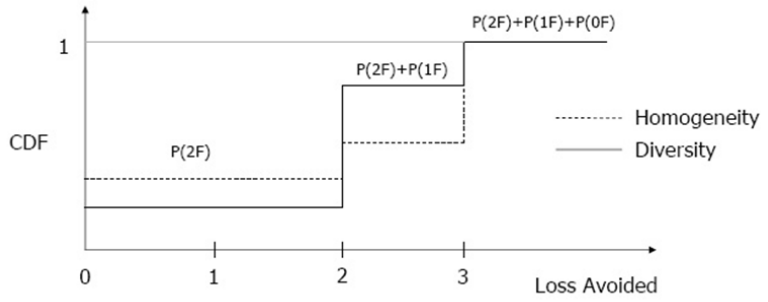


Figure A1. Availability Loss: Cumulative Distribution of Function (CDP)

Which implies that

$$\frac{1+c(\pi+\rho-\pi\rho)}{1+\pi+\rho-\pi\rho} < m < c+(1-c)(1+\pi+\rho-\pi\rho)$$

This takes into account both the software homogeneity scenarios (i.e., all nodes have software configuration 1 or all nodes have software configuration 2). ρ, as before, is the failure correlation for software (configuration) 1 and software (configuration) 2. The aforementioned condition states that when the two software (configurations) have comparable attack rates, software diversity second order stochastically dominates software homogeneity. QED.

Proof of Proposition 2

We can get the results by differentiating the lower bound and upper bound of Proposition 1 to c, π, and ρ.

Derivation of E[Y] and E[Y^2]

$E[Y] = E_{attack}E[Y/attack]$

An attack can be one of three types:

1. Specific to vulnerability in software 1.
2. Specific to vulnerability in software 2.
3. Exploiting a vulnerability common to both software 1 and 2.

Therefore, the expected number of failures under diverse deployment is given by

E[Y] = Prob(attack = 1) * E[Y/attack = 1] +Prob(attack = 2) * E[Y/attack = 2] +Prob(attack = common) * E[Y/attack = common]

Now, a is the rate of attacks on software 1, and $m \cdot$ a is the rate of attacks on software 2, where m is related to relative market shares. $c \cdot a$ is the rate of attacks which are common to both software configurations. Then,

$$Prob(attack = 1 only) = \frac{1-c}{1+m-c}$$

$$Prob(attack = 2 only) = \frac{m-c}{1+m-c}$$

$$Prob(attack = common) = \frac{c}{1+m-c}$$

Therefore,

$$E[Y] = \frac{1-c}{1+m-c} * E[Y_1] + \frac{m-c}{1\_m-c} * E[Y_2] + \frac{c}{1+m-c} * E[Y]$$
$$= \frac{1-c}{1+m-c} * Nx_1\pi + \frac{m-c}{1+m-c} * N(1-x_1)\pi + \frac{c}{1+m-c} * N\pi$$

We know that $E[Y^2] = V[Y] + E[Y]^2$, and $V[Y] = E_{attack}[V[Y/attack]] + V_{attack}[E[Y/attack]]$, using the two we get $E[Y^2] = E_{attack}[V[Y/attack]] + V_{attack}[E[Y/attack]] + E[Y]^2$. Where variance $V[Y]$ for a beta–binomial distribution is given by $V[Y] = N\pi(1-\pi)\rho(1/\rho - 1 + N)$. Therefore, $E[Y^2]$ can be expanded as

$$\begin{aligned}
E[Y^2] = {} & \frac{(1-c)Nx\pi(1-\pi)\rho(\rho^{-1}-1+Nx)}{m-c} \\
& + \frac{(m-c)N(1-x)\pi(1-\pi)\rho\left(\rho^{-1}-1+N(1-x)\right)}{1+m-c} \\
& + \frac{cN\pi(1-\pi)\rho(\rho^{-1}-1+N)}{1+m-c} \\
& + (1-c)\left(Nx\pi - \frac{(1-c)Nx\pi}{1+m-c} - \frac{(m-c)N(1-x)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 (1+m-c)^{-1} \\
& + (m-c)\left(N(1-x)\pi - \frac{(1-c)Nx\pi}{1+m-c} - \frac{(m-c)N(1-x)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 (1+m-c)^{-1} \\
& + c\left(N\pi - \frac{(1-c)Nx\pi}{1+m-c} - \frac{(m-c)N(1-x)\pi}{1+m-c} - \frac{cN\pi}{1+m-c}\right)^2 (1+m-c)^{-1} \\
& + \left(\frac{(1-c)Nx\pi}{1+m-c} + \frac{(m-c)N(1-x)\pi}{1+m-c} + \frac{cN\pi}{1+m-c}\right)^2
\end{aligned}$$