Dynamic Control and Mitigation of Interdependent IT Security Risks

Jeffrey Mounzer[1]
Electrical Engineering
Stanford University
Stanford, CA 94305
jmounzer@stanford.edu

Tansu Alpcan
Deutsche Telekom Laboratories
Berlin Technical University
10587 Berlin, Germany
tansu.alpcan@telekom.de

Nick Bambos
Electrical Engineering
Stanford University
Stanford, CA 94305
bambos@stanford.edu

***Abstract*—Security risk management for information technology-based organizations has become increasingly important in recent years. However, the risk assessment and mitigation strategies that these organizations employ have remained relatively ad hoc and qualitative. In this paper, we extend a quantitative framework for risk assessment called Risk-Rank [1] to include risk mitigation through Markov Decision Processes. By doing so, we provide an analysis-to-action quantitative approach to security risk management, enabling IT managers to perform more comprehensive evaluations of their risk exposures. We demonstrate the effectiveness of this approach through an example related to the patching of computers in a corporate network.**

I. Introduction

Security risk management is a challenging task for modern information technology (IT)-based organizations. As these organizations have grown in size and become highly interconnected through IT infrastructures and the Internet, the diversity of the security risks that they face has increased dramatically. These risks can range from failures of infrastructure components (e.g., computing servers, networks, and data centers) to employee-induced damage (e.g., accidentally misconfiguring a backup system, a disgruntled employee compromising a database) to malicious outside attacks (e.g., hackers disabling a corporate network or releasing malware). All of these can lead to significant financial losses. The IT security risk management problem is further compounded by the fact that there are many different levels and time scales at which risk must be dealt with; some of these are summarized in Table I. The diversity of decision makers who have to manage risk, the time scales over which different types of risks must be managed, and the variety of actions available to each decision-maker are among the factors that increase the complexity of this problem.

Accordingly, there is a growing need for the development of security risk management techniques to help IT organizations deal with their security risks appropriately. Most of the techniques currently used in practice, although systematic, are still largely empirical and/or qualitative in nature [2] and lack a well-developed quantitative structure, particularly for decision-making and optimization. One important way in which quantitative approaches to risk management can have a significant impact is by supplying *computer-assisted decision support* for risk managers, providing increased scalability, transparency, and clarity for the risk management process. Additionally, quantitative models make possible the automation or semi-automation of certain risk management tasks.

With these considerations in mind, in this paper we build upon a novel, broadly applicable quantitative risk assessment approach called Risk-Rank [1] to include risk mitigation techniques through Markov Decision Processes, thereby providing an analysis-to-action quantitative framework for risk management which considers the complex relationships between the various parts of an IT organization in order to assess and then reduce risk. We envision that the Risk-Rank algorithm for both risk assessment and mitigation can be utilized by risk managers to help them deploy their limited resources (such as investment dollars, personnel, and packet sampling mechanisms) more effectively in their efforts to reduce risk.

The amount of prior work in the field of quantitative risk management is still quite limited. The work in this paper extends the Risk-Rank algorithm first presented in [1]. The risk assessment approach used by Risk-Rank is based on methods used in document and image retrieval [3], [4], diffusion processes over graphs [4], [5], and the adjusted Page-Rank algorithm [6] used by the Google search engine. A related but different investigation [5] for risk assessment has proposed Secure-Rank as a scheme for prioritizing vulnerabilities to patch in computer networks, and a separate model for risk assessment based on queueing theory is presented in [7].

The organization of this paper is as follows. In Section II, we review the underlying system model from [1], which captures risk dependencies and risk diffusion across sets, and then describe the Risk-Rank algorithm for risk assessment. Section III explains how to use the Risk-Rank algorithm for risk mitigation and control. In Section IV, a detailed numerical example is explained, showing the complete Risk-Rank algorithm in action. Finally, Section V presents our conclusions and directions for future work.

II. Dependency Model and Risk Diffusion

In order to analyze the complex relationships which influence an organization's security risk exposure, we must first identify the space in which risk-affecting interactions occur.

[1]Research sponsored by Deutsche Telekom AG. J. Mounzer was with Deutsche Telekom Laboratories while this research was conducted.

TABLE I
EXAMPLES OF DECISION-MAKERS, TIME SCALES, AND AVAILABLE ACTIONS FOR IT SECURITY RISK MANAGEMENT.

Decision-Maker	**Time Scale**	**Actions**	**Risks**
Chief Information Officer	Months	Company policies, security investments	Major information security breach
Department Manager	Days/Hours	Deployment of security systems and experts	Vulnerabilities, insider attacks
Intruder Protection System (IPS)	(milli)Seconds	Block ports, isolate networks, raise alarms	Malware (virus, worm)

Accordingly, we begin by defining the three key factors in security risk management:

- *Business units*, $\mathcal{N}_B = \{n_1^B, \ldots, n_{M_B}^B\}$, which can represent infrastructure elements (e.g. data servers), applications (e.g. office software), processes (e.g. customer relationship management, billing), and products/services (e.g. DSL, SMS).
- *Security threats/vulnerabilities*, $\mathcal{N}_S = \{n_1^S, \ldots, n_{M_S}^S\}$, which target and may adversely affect the business units $\mathcal{N}_B$.
- *People*, $\mathcal{N}_P = \{n_1^P, \ldots, n_{M_P}^P\}$, who are the managers and employees running the business units $\mathcal{N}_B$.

The positive integers M_B, M_S, and M_P are the cardinalities of the three sets, respectively.

Now, we can create a series of graphs which model intra-dependencies between members of each of these sets. We define $\mathcal{G}_B = (\mathcal{N}_B, \mathcal{E}_B)$ to be the graph of intra-dependencies amongst business units in $\mathcal{N}_B$, where each edge $\varepsilon_{ij}^B = (i,j) \in \mathcal{E}_B$ represents the dependency between business units i and j in $\mathcal{N}_B$ and is associated with the scalar weight w_{ij}^B denoting its propensity to transfer and cascade risk. The graphs $\mathcal{G}_S = (\mathcal{N}_S, \mathcal{E}_S)$ and $\mathcal{G}_P = (\mathcal{N}_P, \mathcal{E}_P)$, which model the intra-dependencies between security threats/vulnerabilities and between people, are defined similarly.

On the other hand, *inter*-dependencies (*across* the sets $\mathcal{N}_B$, $\mathcal{N}_S$, and $\mathcal{N}_P$) are modeled through *bipartite* graphs. As an example, consider the bipartite graph $G_{PB} = (\mathcal{N}_B, \mathcal{N}_P, \mathcal{E}_{PB})$ shown in Figure 1, which portrays the inter-dependencies (cross-relationships) between employees and business units.

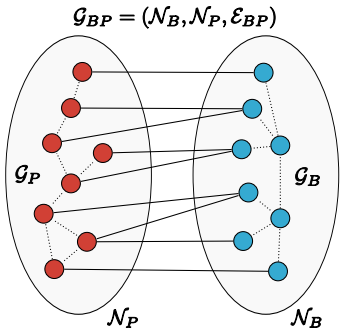

Fig. 1. Bipartite graphs can be used to capture interdependencies between sets, such as between people and business units.

Such a graph can capture important interrelationships, such as who is responsible for controlling a process, managing a product or service, or running an infrastructure system (such as a server). In this case, $\mathcal{E}_{PB}$ denotes the edges between the sets $\mathcal{N}_P$ and $\mathcal{N}_B$. Note that the edges of this graph are always between the members of the (disjoint) sets $\mathcal{N}_B$ and $\mathcal{N}_P$, but not between two members of the same set, because that case is already modeled by the graphs $\mathcal{G}_B$ and $\mathcal{G}_P$. The interrelationships between $\mathcal{N}_B$ and $\mathcal{N}_V$ and between $\mathcal{N}_P$ and $\mathcal{N}_V$ can be modeled through a similar bipartite graph approach. Throughout this paper, we shall focus on such bipartite graphs, in order to see how risk "ping-pongs" and diffuses across our three sets of business units, vulnerabilities, and people.

Next, we analyze how risk can spread throughout an organization, using the mathematical framework of diffusion processes over graphs. The goal is to quantify the risk diffusion process and to compute its equilibrium, so that we can obtain a more accurate idea of how a risk situation evolves over time.

To demonstrate the basic concepts of risk diffusion, for now we will look at a generic bipartite graph $\mathcal{G}_{XY} = (\mathcal{N}_X, \mathcal{N}_Y, \mathcal{E}_{XY})$. We define

$$\mathbf{v}^X(t) := \left[v_1^X(t), ..., v_i^X(t), ..., v_{M_X}^X(t)\right] \in \mathcal{P}^{M_X}$$

to be the *relative risk probability vector* (RRPV) of the elements n_i^X of $\mathcal{N}_X$ in time slot $t \in \{0, 1, 2, ...\}$, where $\mathcal{P}^{M_X} \subset \mathbb{R}^{M_X}$ is defined as the probability simplex

$$\mathcal{P}^{M_X} := \{\mathbf{p} \in \mathbb{R}^{M_X} : p_i \in [0,1] \ \ \forall i \text{ and } \sum_{i=1}^{M_X} p_i = 1\}$$

Each element $v_i^X(t)$ of the RRPV can be interpreted as the proportion of the total amount of risk across all elements in $\mathcal{N}_X$ which lies on element n_i^X at time t. Similarly, we can define the RRPV of the set $\mathcal{N}_Y$ as $\mathbf{v}^Y(t) \in \mathcal{P}^{M_Y}$.

Now, let us define R_{ij} to be the *normalized* (proportional) risk transfer/cascade from node n_i^X of $\mathcal{N}_X$ at t to node n_j^Y of $\mathcal{N}_Y$ at $t+1$. Then, we have $v_j^Y(t+1) = \sum_{i=1}^{M_X} v_i^X(t) R_{ij}$, or in matrix notation

$$\mathbf{v}^Y(t+1) = \mathbf{v}^X(t)\ \mathbf{R}$$

where $\mathbf{R} = \{R_{ij}, i \in \{1, ...M_X\}, j \in \{1, ..., M_Y\}\}$ is the *normalized* $X \to Y$ risk cascade/transfer matrix with $\sum_{j=1}^{M_Y} R_{ij} = 1$ for each $i \in \{1, ..., M_X\}$. $\mathbf{R}$ provides the immediate (one step) risk cascade from X to Y within the bipartite graph $\mathcal{G}_{XY}$.

Similarly, define S_{ji} to be the *normalized* (proportional) risk cascade/transfer from node n_j^Y of $\mathcal{N}_Y$ at t to node n_i^X of $\mathcal{N}_X$ at $t+1$. Then, we can write $v_i^X(t+1) = \sum_{j=1}^{M_Y} v_j^X(t) S_{ji}$, or in matrix notation

$$\mathbf{v}^X(t+1) = \mathbf{v}^Y(t)\ \mathbf{S}$$

where $\mathbf{S} = \{S_{ji}, i \in \{1,...M_X\}, j \in \{1,...,M_Y\}\}$ is the *normalized* $Y \rightarrow X$ risk cascade/transfer matrix with $\sum_{i=1}^{M_X} S_{ji} = 1$ for each $j \in \{1,...,M_Y\}$. $\mathbf{S}$ provides the immediate (one-step) risk cascade from Y to X within the bipartite graph $\mathcal{G}_{XY}$.

Finally, using the $X \rightarrow Y$ risk transfer matrix $\mathbf{R}$, and the $Y \rightarrow X$ risk transfer matrix $\mathbf{S}$, the matrix

$$\mathbf{H} = \mathbf{SR}$$

captures the shortest (two-step) $Y \rightarrow Y$ risk diffusion via the $Y \rightarrow X \rightarrow Y$ 'ping-pong' risk cascade/transfer effect. Like $\mathbf{S}$ and $\mathbf{R}$, the matrix $\mathbf{H}$ is also *row-normalized* (stochastic), since

$$\sum_j H_{ij} = \sum_j \sum_k S_{ik} R_{kj} = \sum_k S_{ik} (\sum_j R_{kj}) = \sum_k S_{ik} = 1$$

for every $i \in \{1,...,M_Y\}$. An example calculation of $\mathbf{H}$ is given below in Section IV.

With this construction, the vectors $\mathbf{v}^X(t)$ and $\mathbf{v}^Y(t)$ can be viewed as probability vectors and the matrix $\mathbf{H}$ as a Markov transition matrix on $\mathbf{v}^Y(t)$. Assuming irreducibility of $\mathbf{H}$, and that the sets $\mathcal{N}_X$ and $\mathcal{N}_Y$ are finite, the process $\mathbf{v}^Y(t+1) = \mathbf{v}^Y(t)\ \mathbf{H}$ would eventually converge to a unique, normalized RRPV $\mathbf{v}^{Y*}$ satisfying $\mathbf{v}^{Y*} = \mathbf{v}^{Y*}\ \mathbf{H}$.

Now, we can proceed to develop an algorithm to evaluate and characterize risk based on our discussion of risk diffusion processes. In particular, since risk can diffuse throughout an organization over time, it is important to strike a balance between the immediate/direct risk while anticipating and planning for the effects of risk diffusion. Accordingly, the Risk-Rank algorithm proposes that the RRPV $\mathbf{v}^Y(t)$ over the nodes in $\mathcal{N}_Y$ evolves according to

$$\mathbf{v}^Y(t+1) = \alpha \mathbf{v}^Y(t)\ \mathbf{H} + \beta \mathbf{v}^Y(0)$$

where $\mathbf{v}^Y(t) \in \mathbb{R}^{|N_Y|}$ is the risk state of the system at time t and belongs to the probability simplex, and where α and β are positive real numbers which sum to one. α and β are parameters chosen by the risk manager which indicate how much consideration should be given to the initial estimate of the RRPV versus its evolution over time. These parameters depend on the particular application and on the risk manager's domain knowledge/expertise.

This iterative process converges to

$$\mathbf{v}^Y_{RR} = \lim_{t \rightarrow \infty} \mathbf{v}^Y(t) = \beta \mathbf{v}^Y(0)[I - \alpha \mathbf{H}]^{-1} \qquad (1)$$

where $\mathbf{v}^Y_{RR}$ can then be used to rank the nodes with respect to their relative risk.

III. Control of Risk Dynamics

In the previous section, we presented the Risk-Rank algorithm for *risk assessment*; based on the data that managers collect, the Risk-Rank algorithm can help them prioritize their risks. However, this is only the first step in the risk management process. Once a risk assessment is complete, the next step is to take appropriate actions to perform *risk mitigation*. In this section, we show that the Risk-Rank approach can also be used to evaluate risk mitigation strategies. More specifically, we shall explore how to control the risk diffusion process over time in order to achieve a more favorable risk distribution across the elements of an organization. Throughout this section, we will assume that the decision maker (e.g., risk manager) is capable of modifying the evolution of the RRPV $\mathbf{v}^Y(t)$ through the allocation of security resources.

We first must establish what it means to achieve a more favorable risk distribution. Suppose that every element in $\mathcal{N}_Y$ has a value $z_y \in \mathbb{R}^+$, and that we denote these values with the vector $\mathbf{z}^Y := [z^Y_1, z^Y_2, \ldots, z^Y_{M_Y}]$. For example, if $\mathcal{N}_Y$ is a set of business units, then each unit would have an associated value which represents the cost incurred if that unit is compromised. Then, the *risk cost* incurred by the elements of $\mathcal{N}_Y$ during time period t can be quantified according to

$$c(t) = \mathbf{v}^Y(t) \cdot z^Y$$

where " $\cdot$ " denotes the dot product between two vectors.

This definition supports the intuition that the risk manager's expected cost increases as the higher-value nodes gain a larger proportion of the total risk. Therefore, the goal of the risk mitigation actions that a risk manager might implement should be to drive risk away from high-value nodes.

At this point, we must deal with a technical issue: due to the normalization inherent to the Risk-Rank approach, controlling the risk diffusion process over $\mathcal{N}_Y$ will result in a *redistribution* of risk over this set of nodes. However, nothing can be said about the *absolute* amount of risk. In order to allow the absolute amount of risk to decrease as a result of possible control actions, we shall find it convenient to define a *risk sink* as an artificial node with zero value, which we introduce into the set $\mathcal{N}_Y$. Any proportion of risk which accumulates on this node can be thought of as having left the network. Note that this requires appropriate modifications of $\mathbf{H}$, $\mathbf{v}^Y(t)$, and z^Y, but that once these are made, the mechanics of the risk diffusion process will remain intact. Also, we assume that if no control actions are taken, no risk will diffuse into the risk sink. We shall discuss the use of the risk sink further shortly, but in the meantime, we shall assume from now on that $\mathcal{N}_Y$ includes a risk sink and that the cardinality of $\mathcal{N}_Y$ is $M_Y + 1$.

Now we apply control actions in order to reduce the risk cost accumulated over time. To do so, we construct and solve a Markov Decision Process (MDP) [8]. An MDP is defined by four key elements: a state space, a set of possible actions, a set of transition probabilities between states given each action, and a cost or reward function. We next describe each of these elements in the context of security risk management.

In the Risk-Rank computation described in Equation 1, the RRPV $\mathbf{v}^Y(t) \in \mathcal{P}^{M_Y+1}$ (which now takes into account the risk sink) can be interpreted as the risk state of the system at time t. However, the relative risk probability simplex is not suitable as the state space for actual deployments of our proposed risk analysis techniques, as evaluation of risk in organizations is much more likely to be quantized to a finite number of levels. This is due to the issue of *observability*; as

the RRPV evolves, it is very difficult for an organization to track it with a high level of precision, and it is much more likely that only a rough estimate of the RRPV can be obtained. In addition, continuous state spaces are typically infeasible for evaluating MDPs numerically.

For these reasons, we define an alternative, finite state space

$$\mathcal{S} = \{\mathrm{s}_1, \mathrm{s}_2, ..., \mathrm{s}_k, \ldots, \mathrm{s}_K\}$$

after quantizing the relative risk probability simplex $\mathcal{P}$ into K *risk regions* (such as those in [7]). Figure 2 shows a visualization of a sample partitioning of the probability simplex $\mathcal{P}^3$ into risk regions. The definitions of risk regions can be very application-specific, due to the diversity of risk metrics employed in industry today. One example partitioning is discussed below in Section IV. It is also useful to define here the average risk vector of state (region) k, $\bar{\mathbf{v}}_k^Y$, which is taken to be the arithmetic mean of risk vectors which belong to partition k. One way that this average vector can be determined is through Monte Carlo simulation, i.e., by generating random vectors according to a uniform distribution over the probability simplex.

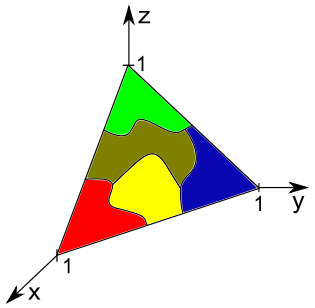

Fig. 2. An example partitioning of the probability simplex $\mathcal{P}^3$ into risk regions.

Once the state space is determined, an action space must be defined. We assume that in each time step, the risk manager can take one action a_l out of a set of available actions

$$\mathcal{A} = \{\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_l, \ldots \mathrm{a}_L\}$$

The result of any of these actions is a modification of the weights of the edges in the bipartite graph model of Section II. In the context of the Risk-Rank algorithm, these actions correspond to modified versions of the matrix $\mathbf{H}$, which we denote as the set

$$\mathcal{H} = \{\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_l, \ldots, \mathbf{H}_L\}$$

The probability of transitioning from the current state $s(t) = \mathrm{s}_k$ to $\mathrm{s}_{k'}$, given that action $a(t) = \mathrm{a}_l$ is taken, is

$$Pr\left(s(t+1) = \mathrm{s}_{k'} | s(t) = \mathrm{s}_k, a(t) = \mathrm{a}_l\right)$$

For each action a_l, we proceed to construct a $K \times K$ transition probability matrix $\mathbf{P}(\mathrm{a}_l)$, where the entries are

$$\mathbf{P}(\mathrm{a}_l)_{ij} = Pr\left(s(t+1) = j | s(t) = i, a(t) = \mathrm{a}_l\right)$$

The transition probabilities which make up $\mathbf{P}(\mathrm{a}_l)$ can again be estimated using Monte Carlo simulation. It is important to distinguish between $\mathbf{H}$, which describes the evolution of relative risk probabilities, and $\mathbf{P}(\mathrm{a}_l)$, which denotes the transition probabilities from one state (risk region) to another.

With the transition probabilities determined, we can describe how the states evolve over time. Given a vector $\mathbf{x}^Y(t) \in \mathbb{R}^K$ which represents the probabilities of being in each state at time t, then after taking action a_l, we have

$$\mathbf{x}^Y(t+1) = \mathbf{x}^Y(t)\,\mathbf{P}(\mathrm{a}_l)$$

The last step in setting up the MDP is to describe an additive cost function which sums the risk cost accumulated by the system over the time stages $1, ..., T$. At time t, the current state contributes a cost $c_s(s(t))$, while the action taken contributes a cost $c_a(a(t))$. While in general we consider c_a to be an application-specific function that captures the cost of performing some action (for example, hardening a node against attack), we provide a more analytical structure for c_s based on our definition of risk cost above. In particular, we define

$$c_s(s_i) = \bar{\mathbf{v}}_i^Y \cdot \mathbf{z}^Y,$$

where $\bar{\mathbf{v}}_i^Y$, is the representative (mean) probability vector of partition (region) i. Therefore, the total cost $C(T)$ accumulated up to time T is

$$C\left(T\right) = \gamma^T c_s(s(T)) + \sum_{t=0}^{T-1} \gamma^t \left[c_s(s(t)) + c_a(a(t))\right], \quad (2)$$

where $\gamma \in [0, 1]$ is a discount factor which captures the typical scenario that risk reductions closer to the present are of more value than those in the future. The goal of the risk manager is then to minimize $C(T)$, where T is the time span of interest.

Given the definition of the MDP above, standard methods such as value or policy iteration [8] can be used to solve for the optimal risk cost-minimizing strategy. The control approach we have described in this section is particularly applicable to scenarios in which a fixed resource must be redeployed in each time period in reaction to a dynamic, rapidly evolving risk situation. A salaried IT employee or a company's Internal Audit division are two examples of such fixed resources.

IV. Numerical Example and Analysis

The dynamic control and mitigation of risk is illustrated here with a numerical example, where the focus is on the patching of computers. We consider an IT manager, who is responsible for five corporate subnets that consist of multiple computers and servers. On each subnet, there are a number of high-priority vulnerabilities which need patching. Each of these vulnerabilities belongs to one of five common *vulnerability classes*, listed in Table II.

A set of costs is associated with the set of subnets, signifying the productivity losses incurred if the subnet is brought down by the exploitation of a vulnerability. Since patching is a time-intensive process [5], it is often the case that an

TABLE II
COMMON VULNERABILITY CLASSES

Class 1	Windows Vulnerabilities
Class 2	Microsoft SQL Server Vulnerabilities
Class 3	Internet Information Services (IIS) Vulnerabilities
Class 4	Microsoft Office Vulnerabilities
Class 5	Phishing-Related Vulnerabilities

IT manager can only work on one subnet at a time (e.g., one subnet per week). Hence, it is important to determine the proper order of patching; a poor ordering can have a significant adverse financial impact.

A simplistic approach would be to start with the subnet which contains the highest number of vulnerabilities, patch all of them, and then proceed to the next subnet. However, this strategy is far from optimal, for several reasons. First of all, since new vulnerabilities are announced quite frequently [5], it is unlikely that the IT manager will ever be able to completely finish patching any single subnet. More significantly, since the subnets all belong to the same corporate network, when a vulnerability on one of the subnets is exploited in an attack, all of the other subnets are at a much higher risk. Accordingly, the risk on each subnet affects the other subnets; clearly, the "ping-pong" effect mentioned in Section II must be considered. These considerations suggest that the IT manager should employ a more sophisticated strategy which takes into account this complex, dynamic environment. Therefore, we explore how the IT manager could use the Risk-Rank algorithm to develop a patching strategy.

We begin by defining $\mathcal{N}_S$ to be the set of vulnerability classes, indexed by $i = 1, \ldots, 5$, and $\mathcal{N}_B$ to be the set of subnets, indexed by $j = 1, \ldots, 5$. The productivity losses incurred by the failures of subnets make up the vector $\mathbf{z}^Y$. Time is discrete and indexed by $t = 0, 1, \ldots, T$, where t has units of weeks. The number of vulnerabilities on each subnet at time $t = 0$ is q_j, and the number of class i vulnerabilities on each subnet at time $t = 0$ is q_j^i.

Given the number of vulnerabilities in each class and consequently for each subnet, the IT manager can determine the relative risk posed to each subnet by each vulnerability class using the simple equation $\theta_{ji} := q_j^i / q_j$, so that θ_{ji} denotes the proportion of the vulnerabilities on subnet j which belonging to class i. These values can be interpreted as the relative probabilities that each subnet will serve as an entry point into the overall corporate network for a vulnerability of the associated class.

At the same time, the IT manager also knows how many class i vulnerabilities exist across the entire network, as well as the distribution of these vulnerabilities over the various subnets. Suppose that there are b_i total class i vulnerabilities in the system. Then, the proportion of these vulnerabilities which lies on each subnet is $\phi_{ij} := q_j^i / b_i$, which can be thought of as the relative probability that if a class i vulnerability is exploited on the network, it would occur on subnet j.

Mapping this information into the notation used in Section II, let $\mathbf{S}$ be the matrix which represents cross-set risk diffusion from $\mathcal{N}_B$ to $\mathcal{N}_S$, and let $\mathbf{R}$ be the matrix which represents cross-set risk diffusion back from $\mathcal{N}_S$ to $\mathcal{N}_B$. Then, the entries of $\mathbf{S}$ and $\mathbf{R}$ can be defined as $S_{ji} = \theta_{ji}$ and $R_{ij} = \phi_{ij}$ $\forall i, j$. The diffusion matrix $\mathbf{H}$ from $\mathcal{N}_B$ to $\mathcal{N}_B$ is consequently $\mathbf{H} = \mathbf{SR}$. For this example, $\mathbf{H}$ shall take the following value (prior to the inclusion of the risk sink):

$$\mathbf{H} = \begin{bmatrix} .61 & .29 & .02 & .04 & .04 \\ .09 & .78 & .05 & .04 & .04 \\ .03 & .30 & .53 & .05 & .09 \\ .05 & .15 & .04 & .71 & .05 \\ .05 & .21 & .08 & .06 & .60 \end{bmatrix}$$

Suppose that the IT manager has observed in the past that the relative frequency of vulnerability exploitations across the subnets is $(.2, .16, .26, .18, .2)$, and that this is used as an estimate of the initial risk vector $\mathbf{v}^Y(0)$. Choosing $\alpha = 0.95$ and $\beta = 0.05$ in the Risk-Rank algorithm (1), which suggests a highly dynamic risk evolution, we obtain a very different perspective due to the cascade of risks, as shown in Figure 3.

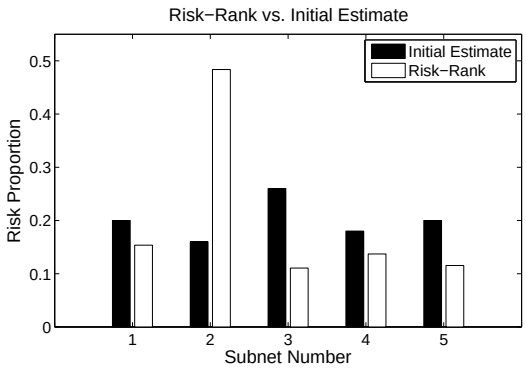

Fig. 3. A comparison between the initial risk estimate and the Risk-Rank one highlights the effects of complex interdependencies. The result indicates that in fact, Subnet #2 is the one at the most risk in the long run, instead of our initial assumption that Subnet #3 had the most risk.

The risk assessment obtained through the Risk-Rank algorithm can be used as a basis for developing an optimal patching strategy for risk mitigation. First, we add a risk sink to the set $\mathcal{N}_B$, which is called node n_6^B. This results in a change to the matrix $\mathbf{H}$; the risk sink is incorporated in a way such that no risk can diffuse into the risk sink, but a small amount of risk can diffuse out of it. This can be thought of as a small amount of additional risk entering the network over time, perhaps due to the announcement of new vulnerabilities.

Next, the resulting probability simplex $\mathcal{P}^6$ is partitioned into risk regions by quantizing each entry $\mathbf{v}_j^Y(t)$ of $\mathbf{v}^Y(t)$ into five levels through a function $\mathcal{L} : \mathcal{P}^6 \longmapsto Q = \{1, 2, 3, 4, 5\}$. These five levels can be interpreted as risk levels such as {very low, low, medium, high, very high}. With five risk levels, there are 210 feasible risk regions for this problem.

The set of actions available to the IT manager constitutes the final component of our risk mitigation framework. These actions could be, for example, assigning employees to patch

subnets, purchasing software upgrades, etc. We assume that in each time period t, the IT manager takes these actions on one particular subnet or does nothing. The patching actions are labeled as $\mathrm{a}_1, \ldots, \mathrm{a}_5$ for the respective subnets, and the action corresponding to doing nothing is denoted a_6.

When an action is taken, the effect of the action manifests itself as a modification of the matrix $\mathbf{H}$. Furthermore, each action is associated with a cost reflecting the effort required to take that action; in this example, we shall assume that this cost is small compared to the productivity losses incurred if a subnet is compromised. Also, we make the following assumptions: if no action is taken (corresponding to a_6), the matrix remains the same, i.e., $\mathbf{H}_6 = \mathbf{H}$. However, if an action is taken to patch the subnet n_j^B, then the corresponding row of $\mathbf{H}$ is modified such that

$$\mathbf{H}_l(j, j') = \begin{cases} (1-\delta)\,\mathbf{H}(j, j') & \text{, if } j' = 1, \ldots, 5 \\ \delta & \text{, if } j' = 6 \end{cases}$$

where $\delta \in [0, 1]$ represents the proportion of risk that is taken out of the network by taking patching action on a subnet. This change to $\mathbf{H}$ represents the fact that patching a subnet will reduce the risk it transfers to the other subnets.

The example described is an illustration of the MDP problem in Section III, and it is solved numerically for the given matrix $\mathbf{H}$ with $\mathbf{z}^Y = (70k, 800k, 10k, 15k, 20k)$, $\delta = .1$, $\gamma = 1$, $\alpha = .95$, $\beta = 0.05$ and a time horizon of $T = 30$. The MDP transition probability matrices $\mathbf{P}(\mathrm{a}_l)$ are found through Monte Carlo simulation as described in the previous section, using $2 * 10^6$ samples per action. The results obtained are shown in Figures 4 and 5. Clearly, in comparison to choosing actions at random, the Risk-Rank method can provide considerable risk cost savings.

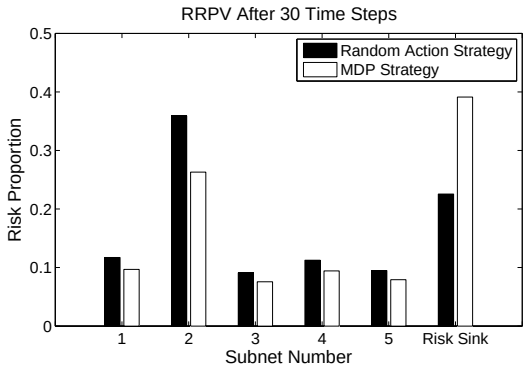

Fig. 4. The Risk-Rank control strategy allows one to optimize the evolution of risk probabilities over time, driving risk away from high-value nodes.

V. Conclusion

The high complexity of modern IT-based organizations means that evaluating and reducing risk is extremely challenging. Systematic, quantitative methods which take into account the interdependence between the different parts of an organization are becoming more and more vital to enable risk managers to make informed decisions. This paper builds upon

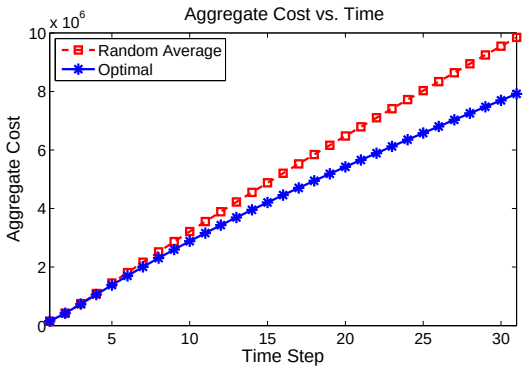

Fig. 5. The aggregate cost of the optimal MDP-based risk mitigation strategy is compared to the average cost of random strategies. A 20 percent reduction in cost is observed over a time period of 30 steps.

the Risk-Rank algorithm [1] and develops a risk control and mitigation framework. Utilizing bipartite graphs and diffusion to model interdepencies among risk factors, the risk state of an organization is quantified. Employing Markov Decision Processes within this model allows risk managers to influence the risk diffusion process, so that they can minimize their risk costs over time. Hence, a quantitative means of risk mitigation is provided. An example for patching networked systems is used to numerically illustrate the dynamic risk control process.

Quantitative methods for dynamic risk control and mitigation are in early stages of research, and there are a number of future directions. One of the particularly interesting aspects of risk management is the issue of observability. Risk can be very difficult to assess, even for those with domain knowledge. Therefore, exploring the impact of inaccurate state estimation and partially observable Markov processes are important research directions. Another topic to investigate is how the dynamics of the risk diffusion process change with the presence of intelligent attackers. Security games can be potentially useful in such an analysis, and they constitute a particularly promising direction for further research.

References

[1] T. Alpcan and N. Bambos, "Modeling dependencies in security risk management," in *CRiSIS*. IEEE, 2009.

[2] K. Dillard, J. Pfost, and S. Ryan, "Microsoft mssc and scoe: The security risk management guide," Online, 2006. [Online]. Available: http://www.microsoft.com/mof

[3] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 355–369, 2007.

[4] C. Bauckhage, *Pattern Recognition*, ser. LNCS. Berlin / Heidelberg: Springer, 2008, vol. 5096/2008, ch. Image Tagging Using PageRank over Bipartite Graphs.

[5] R. A. Miura-Ko and N. Bambos, "Securerank: A risk-based vulnerability management scheme for computing infrastructures," in *ICC*. IEEE, 2007, pp. 1455–1460.

[6] A. Langville and C. Meyer, "A Survey of Eigenvector Methods for Web Information Retrieval," *SIAM Review*, vol. 47, no. 1, pp. 135–161, 2005.

[7] R. A. Miura-Ko and N. Bambos, "Dynamic risk mitigation in computing infrastructures," in *Third International Symposium on Information Assurance and Security*. IEEE, 2007, pp. 325–328.

[8] S. M. Ross, *Applied Probability with Optimization Applications*. San Francisco: Holden-Day, Inc., 1970.