Journal of Information Assurance and Security
ISSN 1554-1010 Volume 6 (2011) pp. 115-123
© MIR Labs, www.mirlabs.net/jias/index.html

An Integrated Approach to Security Risk Management for IT-Intensive Organizations

Jeffrey Mounzer[1], **Tansu Alpcan**[2], **and Nicholas Bambos**[1]

[1]Stanford University, Electrical Engineering Department
350 Serra Mall, Stanford, CA, 94305
jmounzer, bambos @ stanford.edu

[2]Technical University Berlin, Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7, D-10587, Berlin, Germany
alpcan @ sec.t-labs.tu-berlin.de

Abstract: **Security risk management is becoming increasingly important in a variety of areas related to information technology (IT), such as telecommunications, cloud computing, banking information systems, etc. In this paper, we develop a systematic quantitative framework for security risk management in IT-intensive organizations. This framework provides a unified viewpoint to consider a wide array of security risk factors which can disrupt business continuity. Our approach integrates the three phases of security risk management, namely risk modeling, assessment, and control/mitigation, through a formulation based on directed graphs, cascades of failures, and dynamic programming. We consider how security events can propagate through an organization and how resource allocation decisions can be made in order to optimally mitigate the amount of damage they cause. The applicability and effectiveness of our framework is demonstrated through a simple numerical study which shows significant cost reductions when compared to heuristic methods.**

Keywords: security risk management, risk modeling, risk assessment, risk mitigation, resource allocation

I. Introduction

Many modern-day large-scale corporations, universities, and government agencies are heavily dependent on information technology (IT). Although IT creates numerous opportunities for growth and improved efficiency, it also comes with a diverse set of security risks. These include malicious attacks on IT infrastructures (e.g., viruses, malware), unintentional or intentional damage caused by employees (e.g., misconfigurations of firewalls, disgruntled employees releasing sensitive data), and failures of critical processes, applications, or infrastructure components (e.g., computing servers, networks, data centers). Such events can lead to significant financial and productivity losses.

In recent years, there has been substantial growth in the field of IT security risk management, as organizations attempt to develop systematic approaches to reduce their risk exposure [1, 2]. Most of these approaches are qualitative and/or empirical in nature, and they often overlook the complicated interdependencies between different parts of an organization, even though these interdependencies can have a large impact on the organization's overall risk. Therefore, in this paper, we seek to develop a general *quantitative* framework for IT security risk management which accounts for these complex interdependencies. By using mathematical modeling and analysis techniques, this type of approach can provide decision support to risk managers, helping them to be more effective in deploying resources to determine and reduce their risk levels.

In general, risk management involves three phases: data collection, risk assessment, and risk mitigation. Consequently, our proposed quantitative framework integrates all three of these phases in order to provide a unified approach to the risk management process. First, the framework incorporates available risk data – which depends on the domain knowledge of security experts within an organization and which traditionally has been difficult to aggregate and analyze in a cohesive manner – into a single, flexible, graph-based risk model. This model informs the risk assessment phase of the framework, which is based on the methodology of cascades of failures. The risk assessment mechanism we present uses the data provided by the risk model to provide an estimate of the risk costs that will be incurred under the current security state of the organization. The risk modeling and assessment phases then allow for the introduction of mathematical optimization techniques, such as dynamic programming, to control and mitigate risk. Thus, the overall goal of our framework is to enable an organization to use the risk data it collects to analyze its overall risk state and then to deploy resources to reduce its risk exposure over time.

This work draws upon a growing foundation of literature either directly or indirectly related to the topic of security risk management. Several important qualitative and empirical approaches which we build upon are described in security risk management guides written by large information technology organizations or standards bodies [1, 2]. Some quantitative approaches which are related to but different from ours are the Risk-Rank algorithm [3] (based on diffusion processes over graphs), SecureRank [4] (an algorithm for prioritizing vulnerabilities to patch in computer networks), a queueing-theory-based model for risk dynamics presented in [5], and a stochastic modeling approach to security and dependability evaluation (based on Markov processes and game theory) in [6]. Another noteworthy risk analysis approach is Adversarial Risk Analysis, presented in [7], although this work employs a fundamentally different risk modeling and analysis technique. In [8], risk management is approached from a broad business-oriented perspective, and the argument is made that business processes need to be factored into the risk management process - a viewpoint that we share. Their work, however, does not consider interdependencies between business elements and is limited to simple cost-benefit calculations of security actions on individual business elements, as in the approaches of [1, 2].

There are also several related studies which impact only particular phases of our proposed framework. A useful discussion on the cost of providing security is given in [9]; the approach presented therein can be fed into our framework, particularly with respect to determining the costs of risk-mitigating actions. The risk assessment phase of our framework, which is based on the idea of cascades of failures, employs some mathematical techniques which are related to previous work on cascades of failures [10].

In this paper, we begin by describing a general graph-based risk model to capture an IT-intensive organization's risk data in Section 2. This is the model upon which our risk assessment and risk mitigation strategies will be deployed. Section 3 describes the risk assessment phase of our framework, while Section 4 explains the risk mitigation phase. A numerical example illustrating our approach is presented in Section 5. Finally, in Section 6, we offer some concluding thoughts and directions for future work.

II. Risk Model Development

In this section, we develop the underlying graph-based model which we shall use for both risk assessment and risk mitigation. This model allows the risk manager to view both security threats and the assets that they impact as nodes in the same graph, providing a logical, integrated structure for analysis and decision-making.

We begin by defining some key terms in security risk management, which will enable us to characterize the data used in the risk model. We define a *threat* as an action or event which would adversely affect the normal operation of an *asset*, i.e., a business unit, service, process, infrastructure element, etc. An *attack* is an attempt to carry out a threat, which can be successful or unsuccessful; we will also refer to attacks as *activations* of threats. Attacks can be carried out by malicious *agents*, such as hackers or robbers, or by non-malicious agents, such as inattentive employees or nature. A successful attack results in the *failure* or *compromise* of the target; we shall use these two terms interchangeably, although it should be recognized that some effects of attacks are more appropriately described by one or the other. Some examples of different types of threats, corresponding attacks, agents which can carry out these attacks, and failures/compromises caused by successful attacks are given in Table 1.

With these definitions, the data inputs to our graph-based risk model come from the answers to the following five questions, which are reflective of the types of questions asked during the data collection phase of risk management [1, 2]:

1. What is the set of potential security threats?
2. What are the assets which can be compromised by attacks corresponding to these threats, and how much would it cost for each of them to fail?
3. What is the average or hypothetical rate at which each type of attack to the organization occurs?
4. For each identified threat, what are the estimated probabilities that its activation will lead directly to the failure/compromise of each of the assets it can affect, or that it will lead to another attack corresponding to a different threat?
5. For each identified asset, what is the probability that its failure/compromise will directly lead to the failure/compromise of another asset or to a new attack from a different type of threat?

The answers to this list of questions are readily mapped to a directed, weighted graph. Let us define a graph $\mathcal{G} = (\mathcal{N}, \mathcal{R})$, where $\mathcal{N} = \{\mathrm{n}_1, \mathrm{n}_2, \ldots, \mathrm{n}_x, \ldots \mathrm{n}_X\}$ is the set of nodes in the graph and $\mathcal{R} = \{\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_y, \ldots \mathrm{r}_Y\}$ is the set of directed, weighted edges between these nodes. We partition the set $\mathcal{N}$ into two subsets, $\mathcal{V} = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_i, \ldots, \mathrm{v}_I\}$ and $\mathcal{U} = \{\mathrm{u}_1, \mathrm{u}_2, \ldots \mathrm{u}_j, \ldots, \mathrm{u}_J\}$, which represent threats and assets, respectively. From the answer to question (2), each of the elements in $\mathcal{U}$ is assigned a *cost of compromise* $\mathrm{z}_j \in \mathbb{R}^+$, which represents the cost incurred when asset u_j fails or is compromised. From the answer to question (3), each of the elements in $\mathcal{V}$ is assigned a *rate of occurrence*, λ_i, which represents the average number of activations of threat v_i which occur in a time period t.

The set of edges $\mathcal{R}$ forms the operational core of both the assessment and mitigation phases of our risk management framework. Each of the edges in this set is called a *risk transfer probability* (RTP) and represents the probability that the destination node fails[1] as a *direct* consequence of the failure of the source node during a single cascade of failures. An RTP and "the probability that the destination node fails *given* that the source node fails" are *not* the same, since the latter should also incorporate finding all possible cascades of failures which can lead indirectly from the source to the destination. An RTP refers only to the *direct* causal relationship between the source and destination nodes. Accordingly, if a pair of elements of $\mathcal{N}$ lacks an RTP between them, then there is no direct causal relationship between their failures.

An important feature of our risk model is that an RTP can be placed between any two nodes; in particular, the graph is not required to be acyclic, as in Bayesian networks. This allows for more intuitive data collection and risk analysis. For example, one can expect that the failure of asset j can directly lead to the failure of asset j', but also that (at some

[1]Note that the term *fail* here also refers to threats "failing" - this just means that an attack is triggered. Since it will turn out that there is no difference between threats and assets in terms of the mechanics of a cascade of failures as described in Section III, we shall refer to both failures and attacks as simply failures.

Table 1: Examples of threats, assets, attacks, agents, and failures/compromises.

Threat	**Asset**	**Attack**	**Agent**	**Failure/Compromise**
Sensitive corporate data is stolen	Sensitive data	A hacker tries to gain access to a sensitive corporate server by guessing employee passwords	Hacker	Sensitive data is stolen and proprietary corporate secrets are revealed
Power outage brings down web servers	Web servers	Construction company accidentally cuts power lines to server facility and it takes hours to repair	Construction company	Significant loss in web page hits, advertising revenue, and reputation
Virus infects corporate networks and causes massive data loss	Corporate networks, data	Employee accidentally opens email with .exe attachment, allowing a virus to enter the network and possibly infect other computers	Employee, virus creator	If the virus spreads, significant productivity losses and effort spent in attempting to recover lost data

other time) the failure of asset j' can directly lead to the failure of asset j. By allowing for cyclic graphs, we enable the risk manager to easily incorporate such data into the model, even though it would not make sense for both of these events to occur during the *same* failure cascade; we discuss this issue in Section III. Indeed, the only restriction placed on $\mathcal{R}$ is that an RTP cannot have the same source and destination node, since this would have no meaning in our construction (it would represent the probability that a node fails as a direct result of its own failure).

Each RTP can take on values out of a discrete, ordered set of probabilities $\mathcal{D} = \{d_1, d_2, \ldots, d_w, \ldots, d_W\}$, $d_w \in [0,1]$, $d_1 < \ldots < d_W$. The values of the RTPs are provided by the answers to questions (4) and (5) above. A discrete set is used because in most approaches to risk management, the probabilities represented by the RTPs are estimated by humans, and therefore some level of quantization is typically used. For example, security levels are often rated in terms of High, Medium, and Low [1]. The assumption that all RTPs draw their values from the same set $\mathcal{D}$ is made simply for notational convenience, but it can easily be relaxed to allow RTPs to take values from different sets of probabilities.

We can now define the *security profile* (SP) of the organization during time period t as a vector

$$s(t) = \mathrm{s}_k = \left(d_{w_1}, d_{w_2}, \ldots, d_{w_y}, \ldots, d_{w_Y}\right)$$

where d_{w_y} is the value of the RTP r_y. The set of all possible security profiles is the set $\mathcal{S} = \{\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_k, \ldots, \mathrm{s}_K\}$.

In order to make the notion of RTPs more concrete and to highlight how they can be used to model a diverse set of real-world scenarios, we partition $\mathcal{R}$ into four subsets, which are treated in the same manner mathematically but which have substantially different interpretations. These subsets are described below, and their relationships to the partitions of $\mathcal{N}$ are summarized in Figure 1.

- $\mathcal{R}_{\mathcal{VU}} = \left\{\mathrm{r}_{\mathrm{v}_i\mathrm{u}_j},\ i = 1, \ldots, I,\ j = 1, \ldots, J\right\}$ contains all the edges in $\mathcal{R}$ which lead directly from a threat to an asset. Each element $\mathrm{r}_{\mathrm{v}_i\mathrm{u}_j} \in \mathcal{R}_{\mathcal{VU}}$ represents the probability that if threat i is activated, asset j fails as a *direct* result. For example, if v_i represents a hacker attempting to plant a virus and u_j is the computer which is the hacker's direct target, then $\mathrm{r}_{\mathrm{v}_i\mathrm{u}_j}$ is the probability that the hacker is successful in compromising that computer directly as a result of the attack.

- $\mathcal{R}_{\mathcal{UU}} = \left\{\mathrm{r}_{\mathrm{u}_j\mathrm{u}_{j'}},\ j = 1, \ldots, J,\ j' = 1, \ldots, J,\ j \neq j'\right\}$ is the set of all edges in $\mathcal{R}$ which connect one asset to another asset. Element $\mathrm{r}_{\mathrm{u}_j\mathrm{u}_{j'}} \in \mathcal{R}_{\mathcal{UU}}$ represents the probability that $\mathrm{u}_{j'}$ fails as an immediate consequence of u_j being compromised or failing. If u_j represents a data server and $\mathrm{u}_{j'}$ is a business process which depends on that server, then $\mathrm{r}_{\mathrm{u}_j\mathrm{u}_{j'}}$ is the probability that the failure of the server compromises the business process.

- $\mathcal{R}_{\mathcal{UV}} = \left\{\mathrm{r}_{\mathrm{u}_j\mathrm{v}_i},\ j = 1, \ldots, J,\ i = 1, \ldots, I\right\}$ contains those elements of $\mathcal{R}$ which go from an asset to a threat, and they represent the probabilities that the failures of assets directly cause the activation of additional threats. If u_j is a security camera system and v_i is the threat of property theft, then the failure of u_j leads directly to an activation of threat v_i with probability $\mathrm{r}_{\mathrm{u}_j\mathrm{v}_i}$.

- $\mathcal{R}_{\mathcal{VV}} = \left\{\mathrm{r}_{\mathrm{v}_i\mathrm{v}_{i'}},\ i = 1, \ldots, I,\ i' = 1, \ldots, I,\ i \neq i'\right\}$ contains the elements of $\mathcal{R}$ which connect threats to one another, and $\mathrm{r}_{\mathrm{v}_i\mathrm{v}_{i'}} \in \mathcal{R}_{\mathcal{VV}}$ represents the probability that threat i' is activated as an immediate consequence of the activation of threat i. If v_i represents a disgruntled employee and $\mathrm{v}_{i'}$ represents a theft of sensitive data, then the activation of v_i would immediately trigger the activation of $\mathrm{v}_{i'}$ with probability $\mathrm{r}_{\mathrm{v}_i\mathrm{v}_{i'}}$.

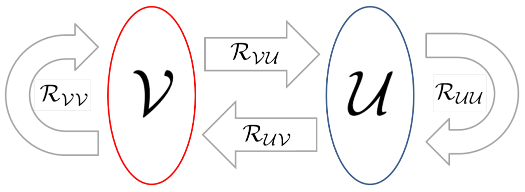
$\mathcal{R}_{\mathcal{VV}}$
$\mathcal{V}$
$\mathcal{R}_{\mathcal{VU}}$
$\mathcal{R}_{\mathcal{UV}}$
$\mathcal{U}$
$\mathcal{R}_{\mathcal{UU}}$

Figure. 1: Relationship between partitions of $\mathcal{R}$ and $\mathcal{N}$.

The description of these subsets of $\mathcal{R}$ highlights how this modeling approach can be applied to a diverse set of scenarios. Based on this general risk model, we can develop a large number of different graph structures in order to characterize real-world situations. For example, certain subsets of $\mathcal{R}$ may not be relevant, depending on the application. Indeed, the current state of the art approach to security risk management [1, 2] essentially only considers the subset $\mathcal{R}_{\mathcal{VU}}$; however, this overlooks the interdependencies between organizational elements which can lead to the types of cascades of failures described in Section III.

By defining the partitions of $\mathcal{N}$ and $\mathcal{R}$ shown in Figure 1, we have established a single mathematical framework for dealing with both threats and assets. Although the interpretations assigned to the members of $\mathcal{N}$ and $\mathcal{R}$ differ depending on which partition they belong to, they remain part of the same graph. This allows us to formulate a general control problem in Section IV for evaluating and comparing the effects of a large variety of types of actions that a risk manager might take.

III. Risk Assessment

Given the risk model above, we now develop the *risk assessment* phase of our framework. We seek to determine the consequences of the graph having a particular security profile over some time period of interest. To do so, we draw upon models for probabilistic cascades of failures, utilizing an approach related to that in [10].

We begin by making a simple intuitive argument based on the example risk model in Figure 2. Suppose that there are

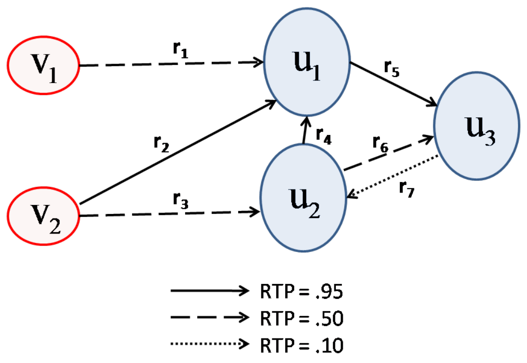

Figure. 2: Example deployment of risk model of Section II.

three security levels for each RTP and that the security profile is $\mathrm{s}_k = (d_2, d_3, d_2, d_3, d_3, d_2, d_1)$ during a 3-month time period t. Now suppose that an attack corresponding to threat v_2 occurs. In the next step of the cascade, asset u_1 will fail with probability d_3, and asset u_2 will fail with probability d_2. The failure of one or both of these assets can subsequently cause the failure of u_3, although here the analysis of the probability of this event becomes a little more complex, as we describe below. In general, as the cascade continues, more nodes will fail, and eventually the cascade will end after having caused the failure of some subset of the nodes in $\mathcal{N}$.

To formalize this argument, we impose the following set of rules on how cascades occur:

1. Each failure cascade occurs independently of all other cascades during time period t. Any single failure cascade begins and ends within the same time period.

2. The organization's SP does not change over the span of a single time period[2].

3. Every cascade has a threat as its root node. The average number of cascades in time period t which begin with threat v_i is λ_i.

4. During an individual cascade, each node can fail once at most, and no repairs occur.

5. When a cascade ends, some subset of the elements in $\mathcal{U}$ will have failed, and the associated cost of the cascade is the sum of the costs of compromise z_j corresponding to the failed assets.

6. After a cascade has ended, the system is repaired and all assets/threats are restored to their unfailed state. Therefore, the system is in its uncompromised state at the beginning of each new attack.

[2]Of course, the length of the time period can be scaled by the model user in order to make this assumption valid.

This set of rules is designed to reflect a large number of real-world scenarios. For example, if we take the perspective of a company's Chief Information Officer (CIO), then a reasonable conjecture is that the time periods of interest are fiscal quarters. Many of the risks that a CIO is concerned about, e.g., theft of data, power outages, denial of service attacks, virus attacks, etc., occur over a much shorter time span - hours, minutes, or even less. Once such an attack ends, the system is typically repaired quite quickly, but the attack will have inflicted some amount of financial damage. As an example, when a denial of service attack on a website ends, the website is quickly available again, but with a loss of user traffic during the attack. The assumption that the SP remains constant during a single time period (and accordingly, during an individual cascade) reflects the fact that the time it takes to enact changes to the SP is often much longer than the time span of the individual cascades of failures or malicious attacks which these changes seek to prevent.

Using the rules above, we now establish the probabilistic manner by which a cascade propagates through the organization. For this part of our discussion, it will be useful to introduce some new notation and definitions. First, we define *parent nodes* as follows: for any node $\mathrm{n}_x \in \mathcal{G}$, its parent nodes are the source nodes of all the edges in $\mathcal{R}$ which have n_x as their destination node. The set of parent nodes of n_x is

$$\Psi_x = \{\psi_{x,1}, \psi_{x,2}, \ldots, \psi_{x,p}, \ldots, \psi_{x,P}\}$$

The corresponding set of RTPs which have n_x as their destination node is then

$$\mathcal{R}_x^{\Psi} = \{\mathrm{r}_{x,1}^{\Psi}, \mathrm{r}_{x,2}^{\Psi}, \ldots, \mathrm{r}_{x,p}^{\Psi}, \ldots, \mathrm{r}_{x,P}^{\Psi}\}$$

Every cascade occurs as a series of stages indexed by the integers $\tau = 1, 2, 3, \ldots$, with stage 1 always being the activation of some threat v_i. We define $\Psi_x(\tau) \subseteq \Psi_x$, with members $\psi_{x,p,\tau}$, as the set of parent nodes of n_x which failed *during* stage $\tau - 1$; this is the set of *active parents* of n_x in stage τ. The corresponding set of RTPs in stage τ is $\mathcal{R}_x^{\Psi}(\tau) \subseteq \mathcal{R}_x^{\Psi}$, with members $\mathrm{r}_{x,p,\tau}^{\Psi}$. Finally, we define Ω_τ as the set of nodes which have *not yet failed* by the beginning of stage τ. We will assume that only active parents can cause new nodes to fail in each stage.

If node n_x belongs to Ω_τ and has only one active parent $\psi_{x,p,\tau}$ in stage τ, then the probability that it fails in that stage is simply given by the value of $\mathrm{r}_{x,p,\tau}^{\Psi}$. If n_x has multiple active parents in stage τ, then the probability that it fails during stage τ is calculated as if each active parent attempts to make it fail independently of the other active parents; they each take one turn at trying to make the node fail[3]. In other words, the probability that, during stage τ, an unfailed node with multiple active parents *survives* is given by the probability that it survives an independent attack from each of its active parents. Finally, if n_x does not have any active parents in stage τ, i.e., if $\Psi_x(\tau) = \{\emptyset\}$, then the node cannot fail in that stage. Observe that if $\Psi_x(\tau) = \{\emptyset\}$ for all $\mathrm{n}_x \in \Omega_\tau$, then the cascade terminated in stage $\tau - 1$, since no new nodes failed during that stage. The foregoing discussion can

[3]Other probabilistic structures which do not assume this independence are possible, but for simplicity we examine the independent case here.

be encapsulated in the following equation:

$$Pr(\mathrm{n}_x \in \Omega_\tau \text{ fails in stage } \tau) = \begin{cases} 1 - \prod_{p=1}^{P}(1 - \mathrm{r}^{\Psi}_{x,p,\tau}) & \Psi_x(\tau) \neq \{\emptyset\} \\ 0 & \Psi_x(\tau) = \{\emptyset\} \end{cases} \quad (1)$$

It is important to keep in mind that although cascades occur as a result of a sequence of events, we treat each cascade as occurring instantaneously with respect to the time period t - the time period t and the cascade stages τ represent different time scales. A helpful analogy is that each cascade occurs as an individual "lightning strike" over the network of nodes in the organization, with the average number of these lightning strikes per time period equal to $\sum_{i=1}^{I} \lambda_i$.

Based on this cascade propagation model, we can determine the probabilities of all possible failure cascades which begin with an attack of a particular threat by creating an *event-based tree*. An *event* in stage τ is either the failure of some combination of nodes in Ω_τ or the end of a cascade. To build an event-based tree, for each event which could occur in stage τ, we determine all possible subsequent events which can occur in stage $\tau + 1$. The probability of each possible event in stage $\tau + 1$, given the event which preceded it in stage τ, is calculated under the assumption that each node in $\Omega_{\tau+1}$ fails or survives independently of all other members of $\Omega_{\tau+1}$. All events in stage $\tau + 1$ which can result from a particular event in stage τ are constructed to be mutually exclusive. For example, in Figure 2, since nodes u_1 and u_2 are the only nodes which can fail in stage 2 as a result of the activation of v_1 in stage 1, then the possible events in stage 2 would be

- Assets 1 and 2 fail.
- Only asset 1 fails.
- Only asset 2 fails.
- Neither asset 1 nor asset 2 fails, so the cascade ends.

Then, we determine all of the possible events which can occur in stage 3, given each of these events (any event corresponding to the end of a cascade cannot lead to any further events). The result of this process is a tree, as shown in Figure 3. Any path from the root node of the tree to a cascade-ending event (choosing one event per stage) represents a possible cascade which can occur if v_2 generates an attack.

Moving now to the general case, we describe a simple iterative process for enumerating the event-based tree associated with a particular threat v_i and for finding the probability of every possible associated cascade.

1. For each possible event which can occur in stage τ, list all of the events which can then occur in stage $\tau + 1$.
2. Find the conditional probability of each of these events given their predecessor in stage τ, using Equation 1 and the assumption of independence of individual failures during a single stage.
3. Multiply the conditional probability of each event given its predecessor by the probability that its predecessor occurs to obtain each event's unconditional probability, i.e., the probability that a cascade which starts with v_i results in this event during stage $\tau + 1$.
4. Continue until no more events can occur, i.e., until all events in stage τ are cascade-ending events.

Observe that the event "Cascade Ends" can occur after every other type of event, and that the conditional probability of this event given its predecessor event is the probability that every node survives in that stage. Also observe that this process for determining possible failure cascades accounts for $\mathcal{G}$ possibly being cyclic, as discussed in Section II.

Once the probabilities of all of the possible cascades associated with each threat have been determined, we can ascertain the cost of being in a certain SP for one time period, based on traditional calculations of risk cost (the probability of failure multiplied by the cost of failure), as

$$c_{RA}(\mathrm{s}_k) = \sum_{i=1}^{I} \lambda_i \left(\sum_{j=1}^{J} \mathrm{z}_j Pr(\mathrm{u}_j \, \mathit{fails} \mid \mathrm{v}_i \, \mathit{fails}, \ \mathit{SP \; is} \, \mathrm{s}_k) \right) \quad (2)$$

i.e., c_{RA} is a cost function which maps each security profile to a cost in $\mathbb{R}^+$.

We note that our framework supports other methods to determine the cost associated with a security profile, and that different approaches may be preferred for certain applications. For example, epidemiological models can be used to determine asset failure probabilities [11]. The reason for this flexibility is that the goal of the risk assessment phase is simply to associate a cost with each SP; as will be made evident in the next section, the exact mechanism employed to determine these costs does not affect the underlying risk mitigation methodology of our framework.

IV. Risk Mitigation and Control

A. Control Formulation

With a method established for assessing the risk cost associated with a security profile, we can proceed to use our framework for *risk mitigation*. The expected risk cost over time can be reduced by investing in security improvements (for example, by allocating resources to improve corporate firewalls or adding personnel to bolster IT security). In our framework, the impact of such investments is a change in the SP of the system; in particular, the values of individual RTPs are reduced by making relevant investments. For example, if v_i represents the threat of a disgruntled employee releasing sensitive corporate data and u_j represents that data, then the real-life action of reducing the employee's access privileges would correspond to a reduction in the value of $\mathrm{r}_{\mathrm{v}_i \mathrm{u}_j}$.

Using the particular risk assessment approach developed in Section III, as individual RTPs are reduced, the corresponding risk cost associated with the overall SP decreases. Accordingly, we can set up an optimization problem using dynamic programming, where the goal is to minimize the risk cost accumulated over time by taking appropriate actions. The key components of a dynamic programming formulation are a state space, a set of possible actions, a transition rule between states, and a cost function.

We define the state space as $\mathcal{S}$, which is the set of all possible SPs defined in Section II. It is important to note that while this state space arises naturally out of our framework, it is subject to Bellman's "Curse of Dimensionality,"

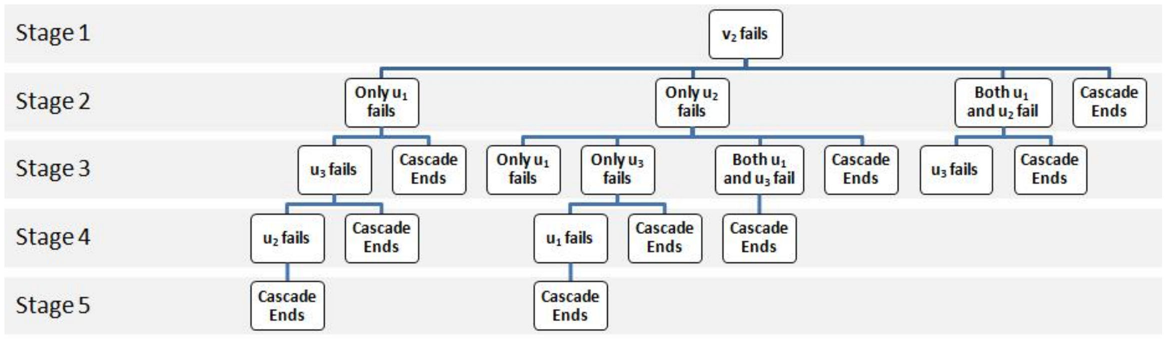

Figure. 3: Event-based tree for threat v_2 for the example in Figure 2.

i.e., as the number of edges in $\mathcal{R}$ grows, the number of possible states grows exponentially. There are several approaches for dealing with this. First, if the number of edges in the graph is relatively small (e.g., taking the perspective of a CIO looking at a high-level view of an organization, with perhaps tens of departments as assets and a small list of high-impact threats), then standard optimization techniques such as dynamic programming are likely possible. Furthermore, if the transition rule between states is taken to be deterministic, then a dynamic programming formulation maps to a shortest path problem, which can accommodate a large state space. If transitions are stochastic and the state space is very large, approximate dynamic programming or heuristics based on graph theory can be used [12]. In order to demonstrate the core ideas of our risk management approach, we consider only deterministic dynamic programming here and reserve discussion of other techniques for a separate investigation.

The actions available to the risk manager are investments out of a budget of $b(t)$ dollars in each time period t. These investments can map to actions such as purchasing/implementing security software, deploying defense resources, hiring employees, assigning man-hours, etc. In general, $b(t)$ is assumed to evolve according to some underlying random process. However, we shall assume that the risk manager's budget for each stage is independent of his budget for previous stages, i.e., he cannot save money in one period to spend in another - this is reflective of the budgeting process in many organizations. Additionally, for simplicity, we shall assume that this quantity is constant and deterministic, although dynamic programming formulations are still possible even if it is random.

The effect of allocating a portion of this budget to an individual RTP r_y is to cause the value of r_y to change according to a (possibly time-dependent) *investment impact function* $F_y : \mathcal{D} \to \mathcal{D}$, so that

$$\mathrm{r}_y(t+1) = F_y(b_y(t), \mathrm{r}_y(t), t) \tag{3}$$

where $b_y(t)$ is the amount invested in reducing r_y during time period t and $\mathrm{r}_y(t)$ is the current value of r_y. The value of $b_y(t)$ is also assumed to be taken from a discrete set, which has minimum value 0 and maximum value $b(t)$. $F_y(b_y(t), \mathrm{r}_y(t), t)$ can be probabilistic, in which case stochastic dynamic programming methods would need to be employed. However, for the purposes of this paper, we shall assume that F is a time-invariant, deterministic function which is the same for all RTPs. Therefore, we shall simplify our notation for the investment impact function to $F(b_y(t), \mathrm{r}_y(t))$. At this point, an action space for our dynamic program can be derived by finding all possible combinations of investment choices which can be made in a single time period. We can list these as

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

With states and actions defined, we now describe how state transitions occur. In general, this model can accommodate any type of probabilistic Markovian state transition process; however, we limit ourselves to a simpler deterministic case here for illustrative purposes. We impose the following restrictions on the problem, which will determine how the state can change from time t to time $t+1$:

1. At the end of each time period, the risk manager can choose to invest $b(t)$ dollars into the system (i.e., the entire budget) or not to invest at all.

2. If the risk manager chooses not to invest in time period t, then the SP will be the same in the next time period.

3. If the risk manager chooses to invest in time period t, then the entire budget of $b(t)$ dollars must be invested into a single RTP[4]. The effect of this investment is to reduce the RTP from d_{w_y} in stage t to d_{w_y-1} in stage $t+1$, unless the RTP is already at d_1, in which case investing in that RTP has no effect. In terms of the investment impact function, we have

$$F(b_y(t), \mathrm{r}_y(t)) = \begin{cases} d_{w_y-1} & b_y(t) = b(t), \mathrm{r}_y(t) > d_1 \\ d_{w_y} & otherwise \end{cases} \tag{4}$$

The last dynamic programming element to define is how cost accumulates over time. Accordingly, we define a cumulative cost function

$$C(T) = c_{RA}(s(T)) + \sum_{t=1}^{T-1} \left[c_{RA}(s(t)) + c_a(a(t))\right] \tag{5}$$

[4]In general, not every RTP can be controlled; some RTPs are likely to be outside the sphere of influence of the risk manager. While in this paper we assume that every RTP is controllable, this assumption can be relaxed by changing the action space appropriately.

where $C(T)$ is the total risk cost incurred by the system up until time T, $s(t)$ represents the state during time period t, $a(t)$ represents the action taken during time period t, the cost function c_{RA} is taken from Section III to be the cost of a security profile based on the risk assessment phase of our framework, and the cost function c_a maps actions to their costs, i.e., to the corresponding investment amount. Based on the simplified state transitions described above, we have

$$c_a(a(t)) = \begin{cases} b(t) & a(t) = invest\,in\,any\,RTP \\ 0 & a(t) = do\,not\,invest \end{cases} \tag{6}$$

The objective of our optimization problem is to minimize the cost $C(T)$ for some T.

B. Solution Method

We investigate one typical optimization objective which a risk manager may want to achieve: the minimization of $C(T)$ over a finite time horizon T. This problem can be solved as a deterministic shortest path problem. We can set up the standard dynamic programming algorithm as

$$H_T(k) = c_{RA}(\mathrm{s}_k)$$

$$H_t(k) = \min_{k' \in \mathcal{S}} \left[c_a(\mathrm{a}_{k'}) + c_{RA}(\mathrm{s}_k) \right.$$
$$\left. + \, H_{t+1}(\mathrm{s}_{k'})\right],\; t = 1, \ldots, T-1 \tag{7}$$

where $H_t(k)$ is the optimal *cost-to-go* of being in state k at time t and $\mathrm{a}_{k'}$ is the particular action which takes the system from state k to state k'.

Since any finite-state deterministic dynamic programming problem is equivalent to a shortest path problem, we can choose any one of a number of efficient algorithms to solve for the optimal sequence of actions and the minimum cost, given a particular starting state. Depending on the assumptions used in setting up the problem, different techniques may be preferred. Under the specific state transition conditions that we describe above, we find it useful to employ an approach related to label-correcting methods [13]. Suppose that our initial security profile is $s(1)$, and that we have T stages over which to perform our cost minimization. Since the number of possible states may be quite large and not every state may be reachable, it would be inefficient to fully enumerate the state space, calculate the cost c_{RA} of every possible state, and then solve according to the general dynamic programming recursion, especially because calculating c_{RA} as described in Section III can itself be a time-intensive process. Indeed, under our conditions from above, any state in which any of the RTPs has a larger value then it does in $s(1)$ would be unreachable. Furthermore, T may not be large enough to reach the minimum cost SP, so there may also be some subset of $\mathcal{S}$ which is unreachable because there is not enough time to get there. These two facts justify the use of the following technique, based on *forward* dynamic programming.

1. Given the set $\mathcal{S}^t$ of states which are possible at time t, determine the set of states $\mathcal{S}^{t+1}$ that are reachable in time $t+1$. If any of these states have not been reached before, add them to the list of "discovered states," and determine their cost using c_{RA}.
2. For each of the reachable states $\mathrm{s}_{k'} \in \mathcal{S}^{t+1}$, determine the set of states in time period t which could have led to it. Call this set $\mathcal{S}^t_{k'}$.
3. For each possible state $s(t+1) = \mathrm{s}_{k'}$, determine the minimum *cost-to-arrive* at that state, $G_{t+1}(\mathrm{s}_{k'})$, based on the equation
$$G_{t+1}(\mathrm{s}_{k'}) = \min_{\mathrm{s}_k \in \mathcal{S}^t_{k'}} \{G_t(\mathrm{s}_k) + c_a(\mathrm{a}_{k'}) + c_{RA}(\mathrm{s}_k)\}$$
where again $\mathrm{a}_{k'}$ is the action which leads from s_k in time period t to $\mathrm{s}_{k'}$ in time period $t+1$.[5] To evaluate the cost $c_{RA}(\mathrm{s}_k)$, use the list of "discovered states." Store $G_{t+1}(\mathrm{s}_{k'})$ and the associated optimal action $\mathrm{a}_{k'}$ to arrive into this state.
4. Repeat until $G_T(\mathrm{s}_{k'})$ is found for every possible state $\mathrm{s}_{k'} \in \mathcal{S}^T$. Then, the overall optimal cost is
$$G^* = \min_{\mathrm{s}_{k'} \in \mathcal{S}^{t+1}} \{G_T(\mathrm{s}_{k'}) + c_{RA}(\mathrm{s}_{k'})\}$$
The optimal sequence of actions which leads to this optimal cost can be deduced by looking backwards starting at the final state $\mathrm{s}_{k'}$ associated with G^*.

V. Simple Numerical Example

We now provide a numerical example to show the entire framework in action. Suppose that the risk manager has worked with domain experts in his organization to identify the set of threats and their attack rates, the set of assets and their costs of compromise, and the set of RTPs along with their current values (and the set of values $\mathcal{D}$ that they can take). This data is incorporated into Figure 4. The risk manager can proceed to perform risk assessment on this model using the method described in Section III. The cost of the current SP turns out to be about 5.6 million dollars/quarter.

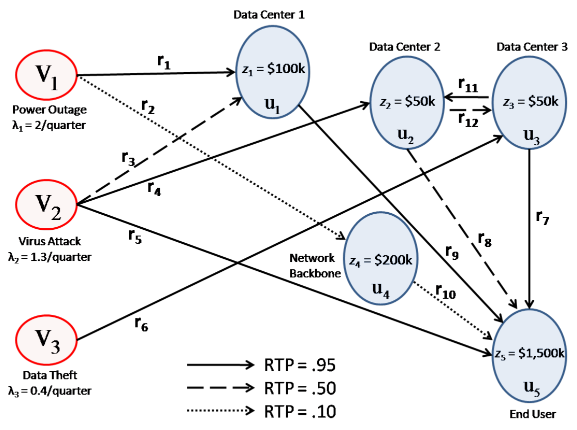

Figure. 4: Risk model description for numerical example.

Now, suppose that the risk manager also has a budget of $b(t) = \$1\text{k}$ which he can invest to reduce risk each quarter, and that he is interested in a time span of 6 quarters; therefore, he can take 6 actions to reduce his risk in the manner

[5]Note that, due to our state transition rule, we do not need to minimize over the action $\mathrm{a}_{k'}$, since there is only one action $\mathrm{a}_{k'}$ which leads from s_k in time period t to $\mathrm{s}_{k'}$ in time period $t+1$.

described in Section IV. We have chosen a small budget for the example simply for illustrative purposes; this will encourage the risk manager to take actions, since now the risk cost incurred by not acting will most likely outweigh the action cost $b(t)$. By using the solution technique outlined in Section IV-B, he obtains the optimal set of actions and security profiles for each of his time periods of interest. A comparison of the optimal solution to an approach in which actions are chosen at random is shown in Figure 5. It is clear that the optimal solution can provide very significant cost savings over the average cost incurred by random action sequences - in this particular example, the optimal solution achieves a 27.8% improvement (approximately nine million dollar cost reduction) over the average cost of randomly chosen actions over the same time span. Another interesting result is that,

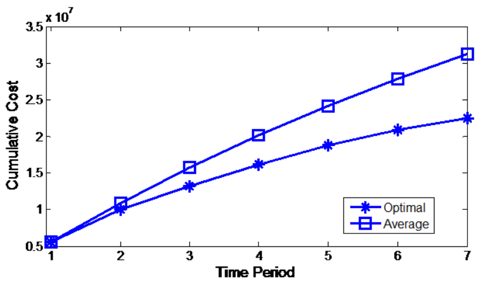

Figure. 5: Performance comparison showing that the optimal action strategy achieves a 27.8% improvement over the average cost of random actions.

for this example, the optimal solution achieves a 16.7% improvement (approximately 4.5 million dollar cost reduction) over the average cost which would result if we allowed the risk manager to randomly choose actions which only affected RTPs in $\mathcal{R}_{\mathcal{VU}}$ (i.e., mapping the problem to the current state of the art approach mentioned in Section II). Therefore, ignoring networked connections between assets can result in significantly higher risk costs incurred over time.

VI. Conclusion

In this paper, we have presented an integrated quantitative framework for security risk management. Based on the general risk model of Section II, we developed a risk assessment technique based on cascades of failures, and we provided a formulation for risk mitigation through mathematical optimization. We then discussed a stylized numerical example in which our framework significantly reduced risk costs.

There are many possible directions for future work; we highlight only a couple here. Several interesting extensions to the current model can be made to explore issues related to *observability*. These issues are faced by any risk management technique; it is often the case that there are unobserved dependencies, undiscovered security threats, insufficient or erroneous data to characterize the risk state, or inaccurate determinations of the effects of control actions. Therefore, assessing the robustness of our framework to these types of problems is an important direction for future study.

Another area for future work is the application of different techniques for the risk mitigation phase of our framework. Stochastic dynamic programming techniques can provide insight into stochastic security profile transitions. Additionally, we are investigating approximate dynamic programming techniques to handle extremely large state spaces.

A third area for further work which shows much promise is the application of game theoretic analysis techniques to our framework. It is often the case that attackers are intelligent and malicious as opposed to random, which means that their actions can be more accurately modeled using game theory. Accordingly, introducing game theory into the model can potentially have intriguing effects.

Acknowledgments

The authors thank Deutsche Telekom AG for their sponsorship of this work.

References

[1] K. Dillard, J. Pfost, and S. Ryan, “Microsoft MSSC and SCOE: The Security Risk Management Guide,” The Microsoft Corporation, 2006. [Online]. Available: http://technet.microsoft.com/en-us/library/cc163143.aspx.

[2] G. Stoneburner, A. Goguen, and A. Feringa, “Risk Management Guide for Information Technology Systems,” National Institute of Standards and Technology, 2002. [Online]. Available: http://csrc.nist.gov.

[3] T. Alpcan and N. Bambos, “Modeling dependencies in security risk management,” 4th IEEE International Conference on Risks and Security of Internet and Systems (CRiSIS), 2009.

[4] R. A. Miura-Ko and N. Bambos, “SecureRank: A risk-based vulnerability management scheme for computing infrastructures,” IEEE International Conference on Communications, 2007, pp. 1455-1460.

[5] R. A. Miura-Ko and N. Bambos, “Dynamic Risk Mitigation in Computing Infrastructures,” IEEE International Symposium on Information Assurance and Security, 2007.

[6] K. Sallhammar, B. Helvik, and S. Knapskog, “On Stochastic Modeling for Integrated Security and Dependability Evaluation,” The Journal of Networks, Vol. 1, No. 5, September/October 2006.

[7] D. Insua, J. Rios, and D. Banks, “Adversarial Risk Analysis,” Journal of the American Statistical Association. Vol. 104, No. 486, June 2009, pp. 841-854.

[8] A. Bialas and K. Lisek, “Integrated, Business-Oriented, Two-Stage Risk Analysis,” Vol. 2, No. 3, 2007.

[9] W. de Bruijn, M. Spruit, and M. van den Heuvel, “Identifying the Cost of Security,” Journal of Information Assurance and Security, Vol. 5, No. 1, 2010.

[10] S. Iyer, M. Nakayama, and A. Gerbessiotis, “A Markovian Dependability Model with Cascading Failures,” IEEE Transactions on Computers, Vol. 58, 2009, pp. 1238-1249.

[11] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, “Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint,” Proc. IEEE SRDS, 2003.

[12] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific, 1996.

[13] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, Massachusetts: Athena Scientific, 2007.

Author Biographies

Jeffrey Mounzer is currently a Ph.D. candidate in the Electrical Engineering Department at Stanford University. He received a B.S. degree in electrical engineering and a B.A. degree in economics from the University of California, San Diego in 2008, as well as an M.S. degree in electrical engineering from Stanford University in 2010. He is a recipient of the Cisco Systems Stanford Graduate Fellowship. His research interests include mathematical modeling, optimization, and control of networked systems, particularly with applications to wireless networking, smart grids, and IT security risk management.

Tansu Alpcan received the B.S. degree in electrical engineering from Bogazici University, Istanbul, Turkey in 1998. He received the M.S. and Ph.D. degrees in electrical and computer engineering from University of Illinois at Urbana-Champaign (UIUC) in 2001 and 2006, respectively. His research involves applications of distributed decision making, game theory, and control to various security and resource allocation problems in complex and networked systems. He has received Fulbright scholarship in 1999 and best student paper award in IEEE Conf. on Control Applications in 2003. Tansu Alpcan has received Robert T. Chien Research Award from the UIUC Department of Electrical and Computer Engineering and Ross J. Martin Research Award from the UIUC College of Engineering in 2006. He was an associate editor for IEEE Conference on Control Applications (CCA) in 2005 and has been TPC member of several conferences including IEEE Infocom 2007-2009. He was the co-chair of the workshop on Game Theory in Communication Networks (GameComm) 2008 and publicity chair of GameNets 2009. He is steering board member and general chair of Conference on Decision and Game Theory for Security (GameSec) 2010. He has received the best paper award in 2010 IEEE Intl. Conf. on Communications (ICC), Comm. and Inf. Sys. Security Symposium. Tansu Alpcan has been a member of IEEE since 1998. He is the (co-)author of more than 90 journal and conference articles as well as the book ”Network Security: A Decision and Game Theoretic Appraoch” published by Cambridge University Press. He has worked as a senior research scientist in Deutsche Telekom Laboratories, Berlin, Germany, between 2006 and 2009. Tansu Alpcan is currently assistant professor (juniorprofessur) in Technical University Berlin while continuing his affiliation with Deutsche Telekom Laboratories.

Nick Bambos received his Ph.D. in electrical engineering and computer science from U.C. Berkeley in 1989, after graduating in Electrical Engineering from the National Technical University of Athens, Greece in 1984. He served on the Electrical Engineering faculty of UCLA from 1990 to 1995 and joined Stanford University in 1996, where he is now a professor in the Electrical Engineering department and the Management Science and Engineering department. His current research interests are in performance engineering of communication networks and computing systems, including queuing and scheduling issues in wireless and wireline networks, as well as ergodic random processes, queuing theory and adaptive control of stochastic processing networks.