Modeling Dependencies in Security Risk Management

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

Nick Bambos
Stanford University
CA, USA
Email: bambos@stanford.edu

***Abstract*—This paper develops a framework for analyzing security risk dependencies in organizations and ranking the risks. The framework captures how risk 'diffuses' via complex interactions and reaches an equilibrium by introducing a Risk-Rank algorithm. A conceptual structure of an organization – comprised of business units, security threats/vulnerabilities, and people – is leveraged for modeling risk dependencies and cascades. The Risk-Rank algorithm captures risk diffusion over time and ranks various risks based on a balancing of the immediate risk versus the future one emerging via cascading across system dependencies. Thus, the presented framework facilitates a systematic prioritization of risks in organizations.**

***Keywords*-Risk modeling, risk dependencies, risk diffusion.**

I. Introduction

Large scale organizations/businesses consist of relatively well-formed, yet interdependent units, typically linked via information technology (IT) infrastructures. Furthermore, they increasingly collaborate and interact with other organizations, due to intense global competition and complexity of modern products and services. Security risk assessment and mitigation in such large scale organizations requires analysis of complex networks of relationships and dependencies that are often only partially observable.

Security risk management poses significant challenges to modern organizations. They range from mitigating failures of infrastructure components, e.g. computing servers, networks or data centers, to managing people-induced damage. The latter can be, for example, due to negligence, such as mis-configuring a firewall or a backup system, or on-purpose, e.g. a disgruntled employee compromising a database, a hacker getting unauthorized access to a computer system, release of a computer virus/worm, or a denial of service attack. Such events can induce direct financial loss or reputation damage resulting in market share loss. For example, a moderate telecommunications service outage for a couple of hours could result in such significant losses.

How should infrastructure elements, business processes and people interactions be configured for higher security and risk mitigation? How should vulnerabilities be identified and incidents prevented? In order to be able to answer such questions in an organization with limited resources, a proper security risk management system has to be in place that prioritizes risks for mitigation actions.

Most of the early security risk management approaches, addressing the questions posed above, have been mainly empirical and qualitative in nature. While empirical approaches still dominate the field, there has been recently a growing interest in quantitative models and methods [1]. This paper develops a novel quantitative framework for security risk analysis in organizations. To that end, we leverage methods applied to document and image retrieval [2], [3], and the adjusted Page-Rank procedure used by the Google search engine [4]. A related but different investigation [5] has proposed Secure-Rank as a scheme for prioritizing vulnerabilities to patch in computer networks.

In this paper, we first develop (section II) a structural framework capturing risk dependencies within and across business units, security threats/vulnerabilities, and people using a graph-theoretic approach. We model (section III) the 'diffusion' of risk via transfer/cascade across such dependencies. The framework captures the dynamic evolution of risk due to complex interactions in contrast to static models focusing on isolated one-time risks. Subsequently, we introduce (section IV) the *Risk-Rank* family of algorithms. They can be used to rank and prioritize risks in an organization, taking into account both the current risks and their evolution over time due to dependencies enabling risk transfer/cascade. We provide (section V) a couple of high-level exemplary applications to operational scenarios. Finally, we conclude (section VI) the paper, discussing key extensions and future research directions.

II. Dependency Model

How can one jointly assess the importance of various *interdependent business units* (including systems, processes, services), the potential impact of various *vulnerabilities* or *threats*, and the risk implications of relationships between *people* in an organization? The term *business unit* of an organization here refers to infrastructure systems, software platforms (applications), business processes, as well as product lines and services. Vulnerabilities and threats can be very diverse ranging from generic malware to specifically targeted phishing attacks. The employees in an organization can have diverse roles, responsibilities, and relationships with respect to each other and the business processes. We view business units, threats/vulnerabilities, and people as the three main

factors involved in security risk management. The goal is to assess how their complex interdependencies contribute to security risk exposure of the organization.

Let us first define the sets of business units, threats and vulnerabilities, and people, as factors in security risk management.

- *Business units* $\mathcal{N}_B = \{n_1^B, \ldots, n_{M_B}^B\}$, representing infrastructure elements (e.g. computing servers), applications (e.g. software), processes (e.g. billing, customer care), and products or services (e.g. DSL service, SMS service).
- *Security threats*/vulnerabilities $\mathcal{N}_S = \{n_1^S, \ldots, n_{M_S}^S\}$, targeting or adversely affecting the business units $\mathcal{N}_B$.
- *People* $\mathcal{N}_P = \{n_1^P, \ldots, n_{M_P}^P\}$, that is, 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 respective sets.

The intra-dependencies among members of each of the above sets are modeled using graphs, as follows.

- $\mathcal{G}_B = (\mathcal{N}_B, \mathcal{E}_B)$ is 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/interaction between business units i and j in $\mathcal{N}_B$ and is associated with the scalar weight w_{ij}^B denoting its 'intensity' (or propensity to transfer and cascade risk).
- $\mathcal{G}_S = (\mathcal{N}_S, \mathcal{E}_S)$ is the graph of intra-dependencies amongst the security threats/vulnerabilities in $\mathcal{N}_S$, where each edge $\varepsilon_{ij}^S = (i,j) \in \mathcal{E}_S$ represents the dependency between threats i and j in $\mathcal{N}_S$ and is associated with the scalar weight w_{ij}^S representing propensity to transfer and cascade risk (for example, the likelihood that exploiting one vulnerability i will lead to exploiting j subsequently).
- $\mathcal{G}_P = (\mathcal{N}_P, \mathcal{E}_P)$ is the graph of intra-dependencies amongst people in the organization, where each edge $\varepsilon_{ij}^P = (i,j) \in \mathcal{E}_P$ represents the relationship between employee i and j in $\mathcal{N}_P$ and is associated with scalar weight w_{ij}^P representing propensity to transfer and cascade risk (for example, compliance failure of employee i will lead to compliance failure of employee j).

Consequently, associated with the above graphs are the matrices $W^B = \{w_{ij}^B\}$, $W^S = \{w_{ij}^S\}$, and $W^P = \{w_{ij}^P\}$, which contain the respective scalar intra-dependency weights.

In order to model the *inter*-dependencies (*across* the sets $\mathcal{N}_B$, $\mathcal{N}_S$, and $\mathcal{N}_P$), we utilize *bipartite* graphs. Consider first the bipartite graph $\mathcal{G}_{PB} = (\mathcal{N}_B, \mathcal{N}_P, \mathcal{E}_{PB})$ shown in Figure 1 and representing the inter-dependencies (cross-relationships) between employees and business units. For example, it could reflect who is responsible for a process, or product/service, or for running an infrastructure system, such as a server. Here, $\mathcal{E}_{PB}$ denotes the edges between the sets $\mathcal{N}_P$ and $\mathcal{N}_B$. It is important to note that the edges of this graph are always between the members of the respective disjoint sets $\mathcal{N}_B$ and $\mathcal{N}_P$, but not within each set. The latter case is already modeled by the graphs $\mathcal{G}_B$ and $\mathcal{G}_P$. The cross-relationships across the other graphs can be modeled using similarly defined bipartite graphs. We actually focus on such bipartite graphs below to see how risk cascades, 'ping-pongs' and gradually spreads (diffuses) on them.

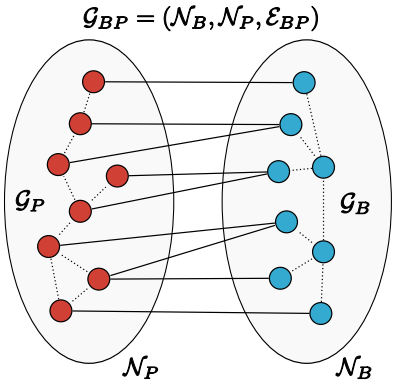

Figure 1. Regular and bipartite graphs can be used to represent the relationships between business units, people, and security threats.

III. Cross-Set Risk Diffusion and Equilibrium

In order to investigate how business units, security vulnerabilities/threats, and people in an organization affect and relate to each other, we adopt an approach based on 'diffusion processes' over a graph [3], [5]. Diffusion processes, as means for computing similarities among the vertices of graphs and ranking them, have been investigated by several studies [6]–[8].

We aim to leverage this methodology for security risk assessment in organizations. Our objective is to explore how risk cascades (direct risk transfer + secondary + tertiary + etc.) and 'diffuses' across and between the business units $\mathcal{N}_B$, people $\mathcal{N}_P$, and vulnerability/threat $\mathcal{N}_S$ factors in an organization. In fact, we want to quantify the risk diffusion process and compute its equilibrium, so that we can assess our current and future risk exposure and identify the highest risk elements that contribute the most.

For simplicity and demonstration purposes, we focus in the limited scope of this paper on the risk diffusion across a generic bipartite graph $\mathcal{G}_{XY} = (\mathcal{N}_X, \mathcal{N}_Y, \mathcal{E}_{XY})$, as shown in Figure 2. On such bipartite graphs the 'ping-pong' dynamics of risk diffusion can be clearly seen. The case for general graphs are discussed at the end of the paper along with future extensions.

Referring to the generic graph $\mathcal{G}_{XY} = (\mathcal{N}_X, \mathcal{N}_Y, \mathcal{E}_{XY})$ of Figure 2 (bipartite), define

$$\mathbf{v}^X(t) = \left(v_1^X(t), ..., v_i^X(t), ..., v_{M_X}^X(t)\right)$$

be the *normalized* risk vector of the elements n_i^X of $\mathcal{N}_X$ at time slot $t \in \{0, 1, 2, ...\}$, where $v_i^X(t) \in [0, 1]$ for every

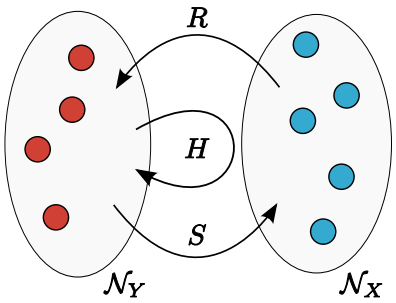

Figure 2. Risk diffusion on a bipartite graph and the 'ping-pong' effect.

$i \in \{1, ..., M_X\}$ and $\sum_{i=1}^{M_X} v_i^X(t) = 1$ for all t. Similarly, define

$$\mathbf{v}^Y(t) = \left(v_1^Y(t), ..., v_j^Y(t), ..., v_{M_X}^Y(t)\right)$$

be the *normalized* risk vector of the elements n_j^Y of $\mathcal{N}_Y$ at time slot $t \in \{0, 1, 2, ...\}$, where $v_j^Y(t) \in [0, 1]$ for every $j \in \{1, ..., M_Y\}$ and $\sum_{j=1}^{M_Y} v_j^Y(t) = 1$ for all t.

Define now 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 $t+1$. Consequently, we can write $v_j^Y(t+1) = \sum_{i=1}^{M_X} v_i^X(t) R_{ij}$ 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 $i \in \{1, ..., M_X\}$. It provides the immediate (one step) risk cascade $X \to Y$ within the bipartite graph $\mathcal{G}_{XY}$. Note that here we ignore (for simplicity) any intra-node risk transfer between $\mathcal{N}_Y$-nodes themselves. We briefly discuss this case at the end.

Similarly, define S_{ji} to be the *normalized* (proportional) risk transfer or cascade of risk from node n_j^Y of $\mathcal{N}_Y$ at t to node n_i^X of $\mathcal{N}_X$ at $t+1$. Consequently, 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* $Y \to X$ risk cascade/transfer matrix with $\sum_{i=1}^{M_X} S_{ji} = 1$ for each $j \in \{1, ..., M_Y\}$. It provides the immediate (one-step) risk cascade and distribution $Y \to X$ within the bipartite graph $\mathcal{G}_{XY}$. Note that here we also ignore (for simplicity) any intra-node risk transfer between $\mathcal{N}_X$-nodes themselves. Again, we briefly discuss the general case at the end.

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

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

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

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

for every $i \in \{1, ..., M_Y\}$, since $\mathbf{R}$ and $\mathbf{S}$ are row-normalized (stochastic) matrices, so $\sum_j R_{kj} = 1$ for all k and $\sum_k S_{ik} = 1$ for all i. In fact, 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 N_X and N_Y are finite), the iteration $\mathbf{v}^Y(t+1) = \mathbf{v}^Y(t)\ \mathbf{H}$ would eventually converge to a unique normalized *risk equilibrium* vector $\mathbf{v}^{Y*}$ satisfying $\mathbf{v}^{Y*} = \mathbf{v}^{Y*}\ \mathbf{H}$.

IV. Risk-Rank (RR) Algorithms

Given an initial (normalized) risk vector $\mathbf{v}^Y(0)$ on $\mathcal{N}_Y$, one could myopically (as it turns out) rank the nodes in $\mathcal{N}_Y$ based on their immediate/direct risk, and declare the one with with maximum immediate risk to be the most 'dangerous' - that is, the $j_o(0) = \arg\max_j \{v_j^Y(0)\}$ one. This ranking strategy, however, would grossly fail to recognize that there is cascade/transfer of risk from $\mathcal{N}_Y$ to $\mathcal{N}_X$ and back, which skews the initial risk vector, via secondary, tertiary, etc. influences, induced by the risk 'ping-pong' effect. To take the latter effect into account, one might go to the extreme of calculating the eventual (stationary) risk vector $\mathbf{v}^{Y*} = \lim_{t\to\infty} \mathbf{v}^Y(t)$ obtained via the iteration $\mathbf{v}^Y(t+1) = \mathbf{v}^Y(t)\ \mathbf{H}$ (or as the solution of $\mathbf{v}^{Y*} = \mathbf{v}^{Y*}\ \mathbf{H}$). Then the nodes could be ranked based on the eventual risk values in $\mathbf{v}^{Y*}$ where the most 'dangerous' node would be the $j_o^* = \arg\max_j \{v_j^{Y*}\}$ one. In fact, the eventually most 'dangerous' node j_o^* could be very different than the immediately most 'dangerous' one $j_o(0)$. Indeed, the eventual risk vector $\mathbf{v}^{Y*}$ does not retain any memory of the initial one $\mathbf{v}^Y(0)$; the memory fades away as the 'ping-pong' effect of risk transfer proceeds.

It is useful to balance out the immediate/direct risk – i.e. present 'danger' – while anticipating and planning for the later induced one. To enable this balance and manage the relevant tradeoff, we introduce the family of **Risk-Rank (RR)** algorithms, which compute the risk vector $\mathbf{v}^Y(t)$ according to the iteration

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

where positive real parameters α and β reflect the weights of current vs. future risk and add up to 1 ($\alpha + \beta = 1$, so that the iteration yields again a normalized risk vector). It can be shown that the Risk-Rank iteration (1) converges to

$$\mathbf{v}_{RR}^Y = \lim_{t\to\infty} \mathbf{v}^Y(t) = \beta \mathbf{v}^Y(0)[I - \alpha \mathbf{H}]^{-1}. \qquad (2)$$

This is can heuristically be seen by direct substitution into (1), but appropriate technical assumptions are needed on

invertibility of $[I-\alpha\mathbf{H}]$, etc. Note that $\mathbf{v}^Y_{RR}$ retains memory of the initial risk vector $\mathbf{v}^Y(0)$, and the initial vs. future risk weights α and β. The Risk-Rank vector $\mathbf{v}^Y_{RR}$ can then be used to rank the nodes with respect to risk, selecting as most 'dangerous' the node $j_o^{RR} = \arg\max_j\{v^Y_{RR,j}\}$, balancing between current and future risk.

If the risk manager, additionally has an estimate of the initial (normalized) risk vector $\mathbf{v}^X(0)$ on the nodes of $\mathcal{N}_X$, then this can be incorporated into the Risk-Rank algorithm, as follows:

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

with positive weights α, β^Y, β^X with $\alpha + \beta^Y + \beta^Y = 1$. This iteration now converges to

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

as can heuristically be seen by substitution (given appropriate technical assumptions). Then, $\mathbf{v}^Y_{RR}$ can be used to rank the risk of various nodes in N_Y, retaining the influence of the initial risk vectors $\mathbf{v}^Y(0)$ and $\mathbf{v}^X(0)$. The Risk-Rank algorithm is reminiscent of the Google Page-Rank algorithm [4], which inspires the name.

V. Application Examples

Let us now provide a couple of application examples of the Risk-Rank algorithm in operational situations. Consider $\mathcal{N}_Y$ to be the set of business units $\mathcal{N}_B$, and $\mathcal{N}_X$ the set of people $\mathcal{N}_P$ running the business units in the organization. Failure of an employee to perform a server maintenance function, for example, can result in compromising of a business application, process, and ultimately service. In turn, that business unit compromise could enable potential failures of other employees to perform further maintenance functions, which may further compromise other business units, and so on. This vicious cycle can evolve to several levels of depth, cascading business unit risk via several 'ping-pong' iterations.

Alternatively, consider N_Y to be the set of of business units $\mathcal{N}_B$, and $\mathcal{N}_X$ the set of possible vulnerabilities or attacks $\mathcal{N}_S$. Exploitation of a vulnerability can compromise a business server, application, process, and ultimately service. In turn, that server compromise may enable (or increase the risk of) further vulnerabilities being exploited, which may further compromise other business unit servers, and so on. Again, this vicious cycle can cascade risk via several 'ping-pong' iterations. How could one decide which is the most significant element of the structure in terms of risk, that is, the most risky for initiating and propagating failures and compromises?

The Risk-Rank framework takes a step towards answering such questions. In order to apply Risk-Rank successfully to real world scenarios, it is necessary to have an appropriate model of the organizational structure and assessment of the initial (normalized) risk vectors $\mathbf{v}^Y$, $\mathbf{v}^X$ and the risk transfer matrices $\mathbf{R}$, $\mathbf{M}$. This can only be achieved through proper data collection mechanisms and involvement of experts with domain knowledge, who can assess these parameters accurately.

VI. The General Case and Conclusions

To best demonstrate the risk transfer and 'ping-pong' effect over time, we have used the generic bipartite graph of Figure 2, suppressing the intra-dependencies and risk transfers internally between nodes in $\mathcal{N}_X$ and internally between nodes in $\mathcal{N}_Y$. In general, one needs to consider such internal risk transfers. Indeed, one has to consider the whole graph of 1) intra-dependencies internal to the sets $\mathcal{N}_B$, N_S and N_P (and other ones that may be appropriate), as well as 2) all the inter-dependencies across elements/nodes of such sets. The Risk-Rank framework and methodology carries over to such general formulations.

Ongoing research includes 1) operational case studies, illustrating the Risk-Rank approach, 2) investigation of risk mitigation and control strategies that affect the risk vector equilibrium via manipulation of dependencies, and 3) security game formulations, based on the above graph models.

Acknowledgments

The authors thank C. Bauckhage for helpful discussions.

References

[1] 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

[2] 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.

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

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

[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] S. Agarwal, "Ranking on graph data," in *Proc. Int. Conf. on Machine Learning, ICML*, 2006, pp. 25–32.

[7] H. Kashima, K. Tsuda, and A. Inokuchi, "Kernels for graphs," in *Kernel Methods in Computational Biology*, B. Schölkopf, K. Tsuda, and J.-P. Vert, Eds. MIT Press, 2004, pp. 155–170.

[8] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf, "Ranking on Data Manifolds," in *Proc. NIPS*, 2004, pp. 169–176.