Algorithmic Aspects of Risk Management

Ashish Gehani[1], Lee Zaniewski[2], and K. Subramani[2]

[1] SRI International
[2] West Virginia University

Abstract. Risk analysis has been used to manage the security of systems for several decades. However, its use has been limited to offline risk computation and manual response. In contrast, we use risk computation to drive changes in an operating system's security configuration. This allows risk management to occur in real time and reduces the window of exposure to attack. We posit that it is possible to protect a system by reducing its functionality temporarily when it is under siege. Our goal is to minimize the tension between security and usability by trading them dynamically. Instead of statically configuring a system, we aim to monitor the risk level, using it to drive the tradeoff between security and utility. The advantage of this approach is that it provides users with the maximum possible functionality for any predefined level of risk tolerance.

Risk management can be framed as an exercise in managing the constraints on edge and vertex weights of a tripartite graph, with the partitions corresponding to the threats, vulnerabilities, and assets in the system. If a threat requires a specific permission and affects a particular asset, an edge is added between the threat and the permission that mediates access to the vulnerable resource. Another edge is added between the permission and the asset. The presence of a path from a threat, through a permission check, to an asset contributes an element of risk. Risk can be reduced by denying access to a resource that contains a vulnerability or activating data protection measures. We analyze some of the problems that form the algorithmic underpinnings of optimal risk management.

1 Introduction

The frequency of attacks faced by the average host connected to the Internet remains elevated, making reliance on manual intervention for response decreasingly tenable. Operating system and application based mechanisms for automated response have increasing utility in this context. We analyze algorithmic aspects of a framework for systematic fine-grained response that is achieved by dynamically controlling the host's exposure to perceived threats and limiting the consequences of security breaches.

Maintaining the security of a host requires it to be continually monitored. When there is suspicion that an attack may be underway, it is prudent to effect a response. The first course of action would be to interrogate the runtime environment to obtain finer-grain data to cross-check the audit information that

G. Agha, O. Danvy, and J. Meseguer (Eds.): Talcott Festschrift, LNCS 7000, pp. 262–276, 2011.

raised the alarm. If the suspicion remains, the next step would be to reconfigure the system (potentially reducing functionality) to limit the exposure of portions that may be vulnerable to the attack in progress. Data that may be affected by the attack should be safeguarded. Measures should be taken to ensure the confidentiality, integrity, and availability of the data after a successful attack. Finally, an effort should be made to gather and preserve forensic information from the environment that may not be available later.

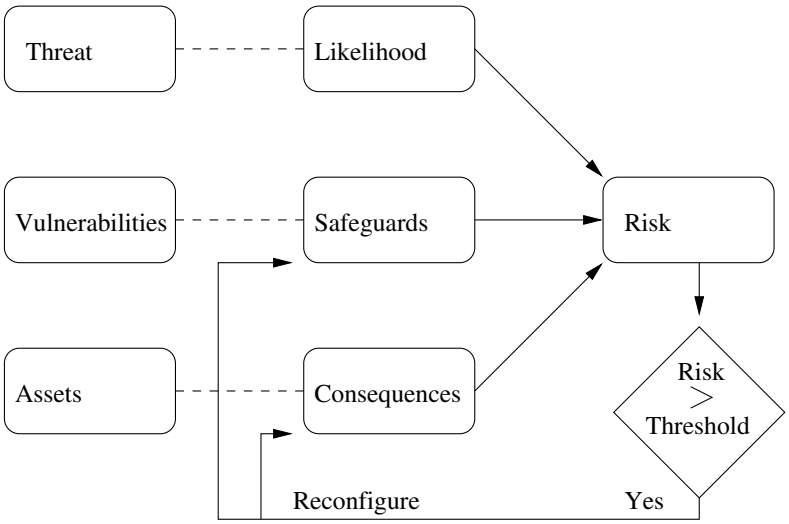

Fig. 1. Risk can be analyzed as a function of the threats, their likelihood, vulnerabilities, safeguards, assets, and consequences. Risk can be managed by using the safeguards to control the exposure of vulnerabilities and manipulating the assets to limit the consequences.

We equate protecting a system with minimizing the risk it faces. The risk is dependent on three factors. The first is the set of threats it faces and their likelihood of occurring. If there are no threats to the system, then it is not at risk. The second factor is the set of vulnerabilities that exist in the system, along with the probability of these being exposed. If there are no vulnerabilities, then even in the presence of a threat, no risk is posed to the system. The third factor is the consequence of an attack succeeding. If there is no consequence, then the system is not at risk.

Whereas threats are under the control of the attacker, vulnerabilities and consequences are within the control of, and can therefore be managed by, the defender. In contrast to previous approaches, we assume that a computation of risk will be used to drive changes in a system's security posture, as depicted in Figure 1. This allows risk management to occur in real time to reduce the window of exposure. We posit that it is possible to protect a system by reducing

its functionality. Our goal is to minimize the tension between security and usability by trading them dynamically. Instead of statically configuring a system, we aim to monitor the risk level, using it to drive the tradeoff between security and utility. The advantage of this approach is that it provides users with the maximum possible functionality for any predefined level of risk tolerance.

2 Risk Model

We now describe some aspects of our risk model, omitting several algorithmic issues covered in previously published work [9] [10] [11] [13], where we discussed mechanisms to efficiently recalculate the risk, subtle reasons for modeling risk tolerance the way we do, how to track the costs and benefits in real time, and how to adapt the model for risk relaxation to improve system performance without exceeding the threshold of risk tolerance.

2.1 Runtime Risk Factors

We model risk as the flow between the first and last partitions in a tripartite graph, depicted in Figure 2, where T is a partition of vertices t_i each represeting a unique threat, W is a partition of vertices w_j each representing a specific weakness in the system, and O is the partition of assets, with the vertices o_k each representing a data object.

Analyzing the risk that a system is faced with requires knowledge of a number of factors. Below we describe each of these factors along with its associated semantics. We define these in the context of the operating system paradigm since our goal is to manage the risk of a host.

Threats. A *threat* is an entity that can cause harm to an asset in the system. We define a threat to be a specific attack against any of the application or system software that is running on the host. It is characterized by an intrusion detection signature. The set of threats is denoted by $T = \{t_1, t_2, \ldots\}$, where $t_\alpha \in T$ is an intrusion detection signature. Since t_α is a host-based signature, it is composed of an *ordered set* of events $S(t_\alpha) = \{s_1, s_2, \ldots\}$. If this set occurs in the order recognized by the rules of the intrusion detector, it signifies the presence of an attack.

Likelihood. The *likelihood* of a threat is the hypothetical probability of its occurring. If a signature is partially matched, the extent of the match predicts the chance that it will lster be completely matched. A function μ is used to compute the likelihood of threat t_α. μ can be threat-specific and depends on the history of system events that are relevant to the intrusion signature. Thus, if $E = \{e_1, e_2, \ldots\}$ denotes the ordered set of all events that have occurred, then $\mathcal{T}(t_\alpha) = \mu(t_\alpha, E \overset{\prec}{\cap} S(t_\alpha))$ where $\overset{\prec}{\cap}$ yields the set of all events that occur *in the same order* in each input set.

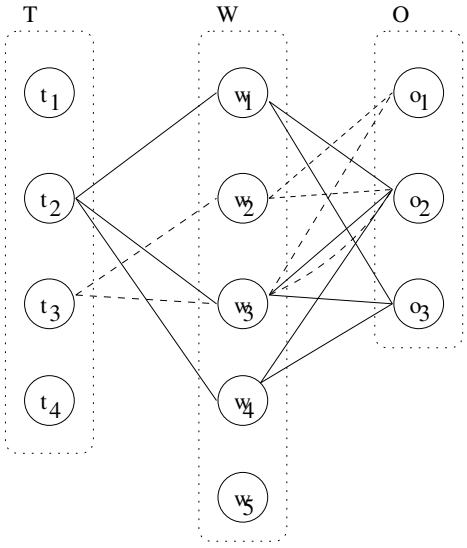

Fig. 2. Operating system risk can be modeled in terms of its constituent components. The threats, weaknesses (corresponding to specific vulnerabilities), and objects (that are the assets) form three disjoint sets. An edge between vertices represents a contribution to the system risk. The system's risk is the total flow between the first and third sets.

Assets. An *asset* is an item that has value. We define the assets as the data stored in the system. In particular, each file is considered a separate object $o_\beta \in O$, where $O = \{o_1, o_2, \ldots\}$ is the set of assets. A set of objects $A(t_\alpha) \subseteq O$ is associated with each threat t_α. Only objects $o_\beta \in A(t_\alpha)$ can be harmed if the attack that is characterized by t_α succeeds.

Consequences. A *consequence* is a type of harm that an asset may suffer. Three types of consequences can impact the data. These are the loss of confidentiality, integrity, and availability. If an object $o_\beta \in A(t_\alpha)$ is affected by the threat t_α, then the resulting costs due to the loss of confidentiality, integrity, and availability are denoted by $c(o_\beta)$, $i(o_\beta)$, and $a(o_\beta)$ respectively. Any of these values may be 0 if the attack cannot effect the relevant consequence. However, all three values associated with a single object cannot be 0, since in that case $o_\beta \in A(t_\alpha)$ would not hold. Thus, the consequence of a threat t_α is $\mathcal{C}(t_\alpha) = \sum_{o_\beta \in A(t_\alpha)} c(o_\beta) + i(o_\beta) + a(o_\beta)$.

By removing an asset from the system, the consequences it faces can be *curtailed* [13]. In the case of data availability, replication serves this purpose, while in the case of confidentiality and integrity, cryptographic operations can be used. For the purpose of estimating risk, a consequence *curtailment* effectively removes the asset from the analysis.

Vulnerabilities. A *vulnerability* is a weakness in the system. It results from an error in the design, implementation, or configuration of either the operating system or application software. The set of vulnerabilities present in the system is denoted by $W = \{w_1, w_2, \ldots\}$. $W(t_\alpha) \subseteq W$ is the set of weaknesses exploited by the threat t_α to subvert the security policy.

Safeguards. A *safeguard* is a mechanism that controls the exposure of the system's assets. The reference monitor's set of permission checks $P = \{p_1, p_2, \ldots\}$ serve as safeguards in an operating system. Since the reference monitor mediates access to all objects, a vulnerability's exposure can be limited by denying the relevant permissions. The set $P(w_\gamma) \subseteq P$ contains all the permissions that are requested in the process of exploiting vulnerability w_γ.

The static configuration of a conventional reference monitor either grants or denies access to a permission p_λ. This *exposure* is denoted by $v(p_\lambda)$, with the value being either 0 or 1. An *active reference monitor* [11][12] allows each permission to be associated with an independent set of constraints that are verified at runtime before granting the permission. By limiting the circumstances under which the permission will be granted, the exposure of the resource being protected is reduced by a predetermined fraction.

The active reference monitor can therefore reduce the exposure of a statically granted permission to $v'(p_\lambda)$, a value in the range $[0, 1]$. This reflects the nuance that results from evaluating predicates as *auxiliary safeguards.* Thus, if all auxiliary safeguards are used, the total exposure to a threat t_α is $\mathcal{V}(t_\alpha) = \sum_{p_\lambda \in \hat{P}(t_\alpha)} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|}$ where $\hat{P}(t_\alpha) = \bigcup_{w_\gamma \in W(t_\alpha)} P(w_\gamma)$.

In practice, since the set of threats cannot be altered by the response apparatus, we can merge the first partition, which contains the threats, into the second by scaling each permission's weight (which represents its probability of being granted) with the sum of the threat likelihoods that have incident edges on the permission.

2.2 Risk Management

The risk to the host is the sum of the risks that result from each of the threats that it faces. The risk from a single threat is the product of the chance that the attack will occur, the exposure of the system to the attack, and the cost of the consequences of the attack succeeding [18]. Thus, the cumulative risk faced by the system is $\mathcal{R} = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}(t_\alpha) \times \mathcal{C}(t_\alpha)$.

If the risk posed to the system is to be managed, the current level must be continuously monitored. When the risk rises past the threshold that the host can tolerate, the system's security must be tightened. Similarly, when the risk decreases, the restrictions can be relaxed to improve performance and usability.

The system's risk can be reduced either by reducing the exposure of vulnerabilities or by limiting the consequences to the data in the event of a successful attack. The former is effected through the use of auxiliary safeguards before granting a permission. The latter is realized by cryptographically protecting and remotely replicating threatened files. Both approaches may also be used simultaneously.

The set of permissions P is kept partitioned into two disjoint sets, $\Psi(P)$ and $\Omega(P)$, that is, $\Psi(P) \cap \Omega(P) = \phi$ and $\Psi(P) \cup \Omega(P) = P$. The set $\Psi(P) \subseteq P$ contains the permissions for which auxiliary safeguards are currently active. The remaining permissions $\Omega(P) \subseteq P$ are handled conventionally by the reference monitor, using only static lookups rather than evaluating associated predicates before granting these permissions. Similarly, the set of files O is kept partitioned into two disjoint sets, $\Psi(O)$ and $\Omega(O)$, where $\Psi(O) \cap \Omega(O) = \phi$ and $\Psi(O) \cup \Omega(O) = O$. The set $\Psi(O) \subseteq O$ contains the files that are currently inaccessible and unmodifiable due to their cryptographic encapsulation. The remaining files $\Omega(O) \subseteq O$ are transparently accessible and modifiable.

At any given point, when safeguards $\Psi(P)$ and curtailments $\Psi(O)$ are in use, the current risk $\mathcal{R}'$ is calculated with $\mathcal{R}' = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}'(t_\alpha) \times \mathcal{C}'(t_\alpha)$ where

$$\mathcal{V}'(t_\alpha) = \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Omega(P)} \frac{v(p_\lambda)}{|\hat{P}(t_\alpha)|} + \sum_{p_\lambda \in \hat{P}(t_\alpha) \cap \Psi(P)} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|}$$

and

$$\mathcal{C}'(t_\alpha) = \sum_{o_\beta \in A(t_\alpha) \cap \Omega(O)} c(o_\beta) + i(o_\beta) + a(o_\beta).$$

2.3 Response Selection

The risk level *after* an event occurs is denoted by $\mathcal{R}_a$. If this increases past the threshold of risk tolerance $\mathcal{R}_0$, the goal of the response engine is to reduce the risk by $\delta_g \geq \mathcal{R}_a - \mathcal{R}_0$ to a level below the threshold. To do this, it must select a subset of permissions $\rho(\Omega(P)) \subseteq \Omega(P)$ and a subset of objects $\rho(\Omega(O)) \subseteq \Omega(O)$, such that adding safeguards and curtailments respectively to the two sets will reduce the risk to the desired level. The resulting risk level is reduced to $\mathcal{R}'' = \sum_{t_\alpha \in T} \mathcal{T}(t_\alpha) \times \mathcal{V}''(t_\alpha) \times \mathcal{C}''(t_\alpha)$ where the new vulnerability measure is

$$\mathcal{V}''(t_\alpha) = \sum_{p_\lambda \in (\hat{P}(t_\alpha) \cap \Omega(P) - \rho(\Omega(P)))} \frac{v(p_\lambda)}{|\hat{P}(t_\alpha)|} + \sum_{p_\lambda \in (\hat{P}(t_\alpha) \cap \Psi(P) \cup \rho(\Omega(P)))} \frac{v(p_\lambda) \times v'(p_\lambda)}{|\hat{P}(t_\alpha)|}$$

and the new consequence measure is

$$\mathcal{C}''(t_\alpha) = \sum_{o_\beta \in (A(t_\alpha) \cap \Omega(O) - \rho(\Omega(O)))} c(o_\beta) + i(o_\beta) + a(o_\beta).$$

2.4 Performance Sensitivity

The choice of safeguards and curtailments also impacts the performance of the system. Evaluating predicates before granting permissions introduces latency in system calls. Cryptographically protecting objects decreases usability. Hence, the choice of subsets $\rho(\Omega(P))$ and $\rho(\Omega(O))$ or subsets $\rho(\Psi(P))$ and $\rho(\Psi(O))$ is subject to the secondary goal of minimizing the overhead introduced.

The adverse impact of a safeguard or curtailment is proportional to the frequency with which it is used in the system's workload. Given a typical workload, we can count the frequency $f(p_\lambda)$ with which permission p_λ is requested in the workload. Similarly, we can count the frequency $f(o_\beta)$ with which file o_β is accessed in the workload. This can be done for all permissions and files. The cost of using subsets $\rho(\Omega(P))$ and $\rho(\Omega(O))$ for risk reduction can then be calculated with

$$\zeta(\rho(\Omega(P)), \rho(\Omega(O))) = \sum_{p_\lambda \in \rho(\Omega(P))} f(p_\lambda) + \sum_{o_\beta \in \rho(\Omega(O))} f(o_\beta).$$

2.5 Abstracting the Problem

The ideal choice of safeguards and curtailments minimizes the safeguards' and curtailments' impact on performance, while simultaneously ensuring that the risk remains below the threshold of tolerance. Thus, for risk reduction we wish to find:

minimize:	$\zeta(\rho(\Omega(P)), \rho(\Omega(O)))$
subject to:	$\mathcal{R}'' \leq \mathcal{R}_0$

Risk management can be viewed as an exercise in picking vertices from the second and third partitions of Figure 2, that need to be protected. Since the set of threats and their likelihoods cannot be altered by the response apparatus, we can merge the first partition, which contains the threats, into the second by scaling each vulnerability's weight with the sum of the threat likelihoods that have incident edges on it. We note that the semantics of risk management require that at each step, the risk must be reduced below the threshold of tolerance. This precludes optimization strategies such as minimizing a weighted sum of risk and runtime performance.

3 On the Hardness of Risk Management

We describe results that provide insights into the algorithmic hardness of the risk management problem.

3.1 Integral Costs and Benefits

When performance-sensitive runtime risk management is viewed as a 0/1 integer nonlinear programming problem with a linear objective function and a quadratic

constraint, it gives rise to a range of related graph problems. For example, consider the problem of selecting a set of responses such that the total cost of effecting them is below a threshold T_1, and simultaneously ensuring that the residual risk is below T_2 when the costs and benefits are integers. Since the costs correspond to the frequency with which a resource is accessed in the workload, the costs are positive integers. In scenarios where the risk associated with each edge is derived by counting the frequency with which the asset (associated with one vertex that the edge is incident upon) is accessed through the permission (associated with the other vertex that the edge is incident upon), the edge weights are also positive integers. This can be defined as the following problem $\mathbf{P}_1$:

Problem 1 ($\mathbf{P}_1$). Given a graph $G = \langle V, E, \boldsymbol{p}, \boldsymbol{w} \rangle$ with V denoting the set of vertices, E denoting the set of edges, $\boldsymbol{w} : V \rightarrow Z$ denoting a weighting function from the vertices to the set of positive integers, and $\boldsymbol{p} : E \rightarrow Z$ denoting a weighting function from the set of edges to the set of positive integers, a vertex threshold T_1 and an edge threshold T_2, is there a subset of vertices V' such that $\sum_{v \in V'} w(v) \leq T_1$ and $\sum_{e=(u,v);\ u,v \notin V'} p(e) \leq T_2$?

Alternatively, the risk management algorithm could attempt to select a set of responses that would impose a cost less than the threshold T_1 but subject to the constraint that the resulting risk reduction would exceed threshold T_2 (where any response primitive chosen would eliminate all risk contributions that depended on access to the targeted permission or asset). This can be formulated as the problem $\mathbf{P}_2$:

Problem 2 ($\mathbf{P}_2$). Given a graph $G = \langle V, E, \boldsymbol{p}, \boldsymbol{w} \rangle$ with V denoting the set of vertices, E denoting the set of edges, $\boldsymbol{w} : V \rightarrow Z$ denoting a weighting function from the vertices to the set of positive integers, and $\boldsymbol{p} : E \rightarrow Z$ denoting a weighting function from the set of edges to the set of positive integers, a vertex threshold T_1 and an edge threshold T_2, is there a subset of vertices V' such that $\sum_{v \in V'} w(v) \leq T_1$ and $\sum_{e=(u,v);\ u \in V' \text{ or } v \in V'} p(e) \geq T_2$?

The two problems $\mathbf{P}_1$ and $\mathbf{P}_2$ can be seen to be identical. Implementing a solution for one therefore immediately provides a mechanism to address the other. The equivalence can be seen since an instance of $\mathbf{P}_1$ can be represented as an instance of P_2 by replacing T_2 with $\sum_{e \in E} w(e) - T_2$ and vice versa. An important point to note about $\mathbf{P}_2$ is that if a vertex in $V - V'$ does not have any incident edges, then it is automatically included in V'.

3.2 Independent Vulnerabilities and Consequences

In our initial investigation, we found that even simplifications of the performance-sensitive runtime risk management problem are algorithmically hard to solve. For

example, consider the case where every attack relies on a single vulnerability and affects a single asset. The corresponding graph is a *matching*. Optimal response selection in this scenario is algorithmically expensive as shown below:

Theorem 1. $\mathbf{P}_2$ *is* NP-complete *even if* G *is a matching.*

Proof. We reduce the 0/1 knapsack problem to $\mathbf{P}_2$. The knapsack problem is known to be NP-complete [8].

An instance of the knapsack problem is characterized by n objects $O = \{o_1, o_2, \ldots, o_n\}$ with respective profits $\{p_1, p_2, \ldots, p_n\}$ and respective integer weights $\{w_1, w_2, \ldots, w_n\}$, a knapsack capacity W and a profit target T. The goal is to pack objects into the knapsack so as to obtain a profit of at least T, while ensuring that the sum of the weights of the objects is at most W. Without loss of generality, we can assume that the weights are even integers.

Given the knapsack instance, we construct the following instance of $\mathbf{P}_2$. Corresponding to object O_i, create two vertices v_i and v_{n+i} and an edge connecting them with weight p_i. The two vertices are given weight $\frac{w_i}{2}$ each. The vertex threshold is set at W and the edge threshold is set at T.

We claim that the knapsack instance is a "yes" instance if and only if the $\mathbf{P}_2$ instance is.

Assume that the given knapsack instance is a "yes" instance, i.e., there is a set of objects $O' \subseteq O$, such that $\sum_{y:y\in O'} w(y) \leq W$ and $\sum_{y:y\in O'} p(y) \geq K$. Pick the vertices in the $\mathbf{P}_2$ instance that correspond to these objects. As per the construction, the vertex threshold of these vertices is at most W and the edge threshold is at least T.

Now assume that the $\mathbf{P}_2$ instance is a "yes" instance, i.e., there is a collection of vertices whose combined weight is at most W and the sum of the weights of the edges connected to these vertices is at least K. As per the construction of the $\mathbf{P}_2$ instance, if vertex v_i is picked, then so is vertex v_{n+i}. Further, the contribution of these two vertices to the vertex threshold is w_{n+i} and to the edge threshold is p_i. Consider the objects corresponding to the picked vertex pairs. As per the construction, their weights sum to at most W and their profits sum to at least K. □

3.3 Qualitative Exposures and Consequences

Instead of considering the case when each vulnerability affects a different asset in the system, we extended the scope of the problem to consider the result when each vulnerability could affect multiple assets and each asset could be affected by multiple vulnerabilities. We restrict the problem to the case where only qualitative knowledge about the vulnerabilities and consequences in the system is available, with the result that a vertex exists for each vulnerability and asset in the system, but it is unweighted.

Since only their absence or presence is known, an unweighted edge between the permission guarding a vulnerability and the object affected by the consequence is inserted only when the vulnerability and consequence are both present. To ensure that the risk remains below a predefined threshold, vertices can be removed by deactivating the corresponding permissions or curtailing the relevant consequences. The result is that an edge incident on any of the removed vertices would itself be removed from the graph, reducing the risk. This is formulated as problem $\mathbf{P}_3$:

Problem 3 ($\mathbf{P}_3$). Given a bipartite graph with unweighted vertices and unweighted edges, find the smallest set of vertices, subject to the constraint that the number of edges remaining after the vertices are removed is below a predefined threshold.

3.4 Known Workloads

The formulation of $\mathbf{P}_3$ did not account for the frequency with which each response primitive occurs in the workload. In practice, the frequency with which the safeguard or data protection primitive is invoked affects its impact on performance. Picking primitives with lower frequencies is therefore preferable. When a workload is known in advance, the problem can be formulated as $\mathbf{P}_4$:

Problem 4 ($\mathbf{P}_4$). Given a bipartite graph with weighted vertices and unweighted edges, find the set of vertices with the lowest sum of vertex weights, subject to the constraint that the number of edges remaining after the vertices are removed is below a predefined threshold.

3.5 Dynamic Application Workloads

We can generalize the risk model from the case where exposure and consequences are considered only qualitatively – that is, only their presence or absence is known, to the case where an estimate of their degree is known. If the degree is estimated with an integer, then the risk contributed by the presence of each exposure and consequence pair is also an integer (since it is the product of two integers). Therefore the edges in the bipartite graph constructed to represent the risk has integer weights.

In general, if the target application workload is known *a priori*, information gleaned from it can be used to optimize the choice of risk management responses. The approach comes with the caveat that predicting a target workload may be nontrivial. In particular, past workloads may not be available and even if they are, they may not be representative of future tasks. Additionally, if the target workload has high variance – that is, if it dynamically and significantly changes its characteristics, then the use of average frequencies for vertex weights

can result in distorted tradeoffs between cost and benefit estimates of selecting specific responses. In such a situation, we can factor out performance sensitivity by using unweighted vertices. The corresponding formulation is $\mathbf{P}_5$:

Problem 5 ($\mathbf{P}_5$). Given a bipartite graph with unweighted vertices and weighted edges, find the set of vertices with the lowest sum of vertex weights, subject to the constraint that the number of edges remaining after the vertices are removed is below a predefined threshold.

We show $\mathbf{P}_5$ is *NP-complete* by reducing the *vertex cover problem* to it. Recall that this determines whether it is possible to construct a *cover* of a specified size, where a cover is a subset of vertices with the property that every edge in the graph has at least one end incident upon one of the vertices in the cover.

Theorem 2. $\mathbf{P}_5$ *is* NP-complete.

Proof. Assume we wish to check whether a cover of size k exists for graph G. We construct a bipartite graph B with disjoint partitions π_1 and π_2 as follows.

For each vertex v_i in G, we add four vertices, $v_{i,1}$, $v_{i,2}$, $v_{i,3}$, and $v_{i,4}$ to B. $v_{i,1}$ and $v_{i,3}$ are inserted in π_1 while $v_{i,2}$ and $v_{i,4}$ are inserted in π_2. An edge between $v_{i,3}$ and $v_{i,4}$ is added to B and given weight 1. A second edge, between $v_{i,1}$ and $v_{i,4}$, and a third edge, between $v_{i,2}$ and $v_{i,3}$, are also added to B. The second and third edges are each given weight $k+1$.

For each edge (v_i, v_j) in G, we add two edges with weight $k+1$ to B. The first is between $v_{i,1}$ and $v_{j,2}$ and the second is between $v_{i,2}$ and $v_{j,1}$. This is illustrated in Figure 3.

The target number of vertices to be removed is set to $\rho = \frac{|\pi_1|+|\pi_2|}{2}$, that is, half the total number of vertices in B. We run our algorithm for $\mathbf{P}_5$ on B with threshold k. If we can remove ρ vertices subject to the constraint that the total weight of the remaining edges is below the threshold k, then there exists a vertex cover of size k for G.

The reason the reduction holds is as follows. Since the edges connecting $v_{i,1}$ to $v_{i,4}$ and $v_{i,2}$ to $v_{i,3}$ have weights that exceed the threshold, either $v_{i,1}$ or $v_{i,4}$ and either $v_{i,2}$ or $v_{i,3}$ must be removed for the total weight of the remaining edges to be below the threshold. Since half of the vertices can be removed from the graph, exactly one vertex is removed from each of the pairs. To remain below the threshold, it is necessary to remove all the edges in B that were added in correspondence to the edges in G. Specifically, if v_i is a vertex in the cover of G, then $v_{i,1}$ and $v_{i,2}$ must be removed from B for the threshold constraint to be maintained. Since only one of the vertices $v_{i,1}$ and $v_{i,4}$ and only one of $v_{i,2}$ and $v_{i,3}$ can be removed, if $v_{i,1}$ and $v_{i,2}$ are selected for removal, then $v_{i,3}$ and $v_{i,4}$ must remain in B along with the edge between the two. Conversely, any group

of four vertices ($v_{i,1}$, $v_{i,2}$, $v_{i,3}$, and $v_{i,4}$) in B that corresponds to a vertex v_i in G that is not in the cover can have either $v_{i,2}$ or $v_{i,3}$ removed without increasing the total weight of the edges. Thus, the only way half the vertices of B can be removed while the total weight of the edges remains below the threshold k is if the corresponding vertices in G form a cover of size k. □

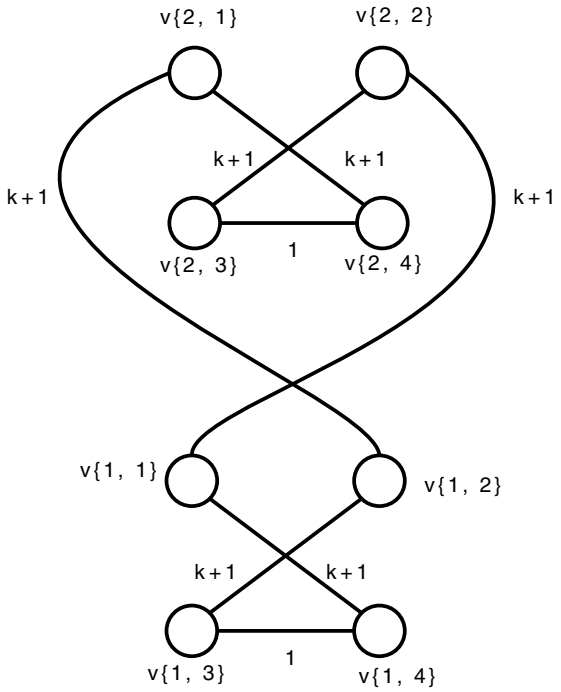

Fig. 3. Given a graph with two vertices, v_1 and v_2, and an edge between them, this is the corresponding bipartite graph

4 Open Questions

Although problems $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are the minimally and maximally constrained versions of $\mathbf{P}_4$ and $\mathbf{P}_5$, the complexity of $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ remains to be analyzed. In addition, efficient solutions (and approximation algorithms, as needed) must be designed for these problems. Further, variants of the problems where edges (representing risk) can have real values instead of integers when an active monitor (modeled in Section 2) is used to dynamically limit the exposure of the system, or the consequences are estimated with fractional values.

Cascading Dependencies

Our experience developing a prototype risk manager for the operating system paradigm uncovered the problem of *cascading dependencies*. If the semantics

of risk reduction through edge removal requires both vertices that the edge is incident upon to also be removed, then the effect may be a cascade of edge removal. The total risk reduction that results by selecting a set of responses is not just the risk reduction corresponding to the sum of the edge weights of the induced subgraph. Instead, it also includes the risk from the sum of the edges that have a single end incident on any vertex in the set of selected responses. This significantly complicates the problem, since it may potentially introduce a cascading set of dependencies, all of which must be examined to determine the optimal choice of edges. Figure 4 illustrates the issue.

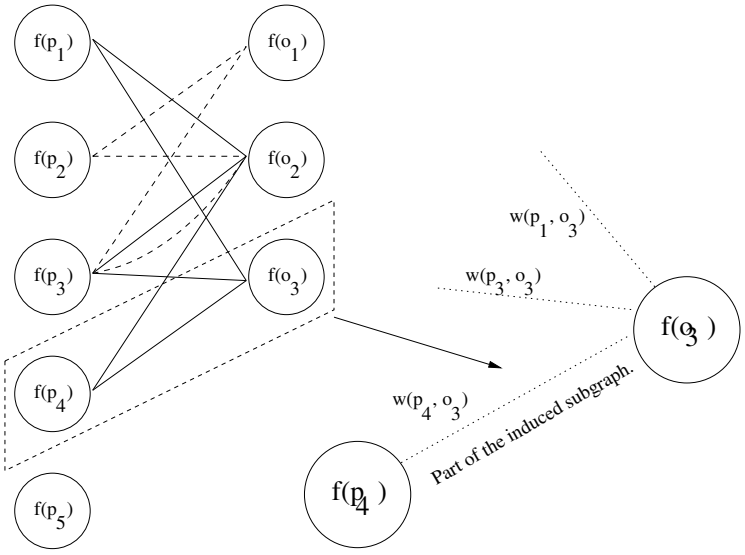

Fig. 4. When edge (p_4, o_3) is removed, if the semantics require that both vertices that it is incident upon must be inactivated, then the dependency cascade would leave only p_5 operational

5 Related Work

The effort to manage the risk of information systems can be traced to the use of the Annual Loss Expectancy (ALE) metric [6][7] by large data processing centers. The use of the ALE paradigm by commercial tools [19], coupled with a focused research effort [2][3][4][5], resulted in improvements in risk modeling. Although risk analysis has been used to manage the security of systems for several decades [6], its use has been limited to offline risk computation and manual response. SooHoo [20] proposed a general model using decision analysis to estimate computer security risk and automatically update input estimates. Bilar [1] used reliability modeling to analyze the risk of a distributed system. Risk is calculated as a function of the probability of faults being present in the system's

constituent components. Risk management is framed as an integer linear programming problem, aiming to find an alternate system configuration, subject to constraints such as acceptable risk level and maximum cost for reconfiguration.

Problems $\mathbf{P}_1$ through $\mathbf{P}_5$ are being proposed for the first time, although variants have been studied and described earlier. The 0/1 1-dimensional knapsack problem is called a *weakly NP-complete* problem, since it admits algorithms whose running times are polynomial if the problem parameters are represented in unary. Such algorithms are called pseudo-polynomial algorithms. Ibarra and Kim presented a pseudo-polynomial time algorithm [14] for the knapsack problem that has been used as the basis for the design of an efficient fully polynomial approximation scheme for the knapsack problem. Efficient approaches to unidimensional and multidimensional variants of the knapsack have also been developed [17][16]. This paper shows that problems $\mathbf{P}_1$ through $\mathbf{P}_5$ are NP-complete.

In the prototype implemented [9], a heuristic was used to guarantee that the risk is maintained below a threshold. Although approximation algorithms exist [15], they were not employed since the choices had to be made in real time. The heuristic used is based on the greedy algorithm for the *0-1 Knapsack Problem* that yields a solution that is always within a factor of 2 of the optimal choice [8]. When the risk needs to be reduced, the heuristic uses the greedy strategy of picking the response primitive with the highest benefit-to-cost ratio repeatedly until the constraint is satisfied. By maintaining the choices in a *heap* data structure keyed on the benefit-to-cost ratio, each primitive in the response set can be chosen in $O(1)$ time. This is significant, since implementing a single response primitive is often sufficient to disrupt an attack in progress. A separate *heap* is used to maintain the active safeguards keyed by the cost-to-benefit ratio instead. When the risk needs to be relaxed, the active safeguards with the highest cost-to-benefit ratios can be selected, since these yield the best improvement to system performance. A future avenue of research is empirical comparison of the approximation and NP-complete algorithms to the heuristic.

6 Conclusion

We note that the semantics of risk management require that the risk be reduced below the threshold of tolerance each time it is found to exceed it. Risk can be reduced by denying access to a resource that contains a vulnerability or by activating data protection measures. This is modeled as the removal of edges representing risk in the aforementioned graph. Depending on whether the risk estimates are integers or reals, whether the vulnerabilities and consequences are independent or conditional, whether the application workload is known in advance, whether the workload is stable or changing rapidly, and depending on the semantics of response selection, there are different underlying graph problems. We analyzed some of the problems that form the algorithmic underpinnings of optimal risk management.

References

1. Bilar, D.: Quantitative Risk Analysis of Computer Networks, Ph.D. Thesis, Dartmouth College (2003)
2. 1st Computer Security Risk Management Model Builders Workshop, Martin Marietta, Denver, Colorado, National Bureau of Standards (May 1988)
3. 2nd Computer Security Risk Management Model Builders Workshop, AIT Corporation, Ottawa, Canada, National Institute of Standards and Technology (June 1989)
4. 3rd International Computer Security Risk Management Model Builders Workshop, Los Alamos National Laboratory, Santa Fe, New Mexico, National Institute of Standards and Technology (August 1990)
5. 4th International Computer Security Risk Management Model Builders Workshop, University of Maryland, College Park, Maryland, National Institute of Standards and Technology (August 1991)
6. Guidelines for Automatic Data Processing Physical Security and Risk Management, National Bureau of Standards (1974)
7. Guidelines for Automatic Data Processing Risk Analysis, National Bureau of Standards (1979)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
9. Gehani, A.: Support for Automated Passive Host-based Intrusion Response, PhD thesis, Duke University (2003)
10. Gehani, A.: Performance-sensitive Real-time Risk Management is NP-Hard. In: Workshop on Foundations of Computer Security affiliated with the 19th IEEE Symposium on Logic in Computer Science (2004)
11. Gehani, A., Kedem, G.: RheoStat: Real-time Risk Management. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 296–314. Springer, Heidelberg (2004)
12. Gehani, A., Kedem, G.: Real-time Access Control Reconfiguration. In: International Infrastructure Survivability Workshop affiliated with the 25th IEEE International Real-Time Systems Symposium (2004)
13. Gehani, A., Chandra, S., Kedem, G.: Augmenting Storage with an Intrusion Response Primitive to Ensure the Security of Critical Data. In: 1st ACM Symposium on Information, Computer and Communications Security (2006)
14. Ibarra, O., Kim, C.: Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. Journal of the ACM 22(4) (1975)
15. Kellerer, H., Pferschy, U.: A new fully polynomial approximation scheme for the knapsack problem. In: Jansen, K., Rolim, J.D.P. (eds.) APPROX 1998. LNCS, vol. 1444, pp. 123–134. Springer, Heidelberg (1998)
16. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
17. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley and Sons, New York (1990)
18. Guidelines for Automatic Data Processing Physical Security and Risk Management, National Institute of Standards and Technology (1996)
19. Description of Automated Risk Management Packages that NIST/NCSC Risk Management Research Laboratory Has Examined, National Institute of Standards and Technology (1991)
20. Hoo, K.S.: Guidelines for Automatic Data Processing Physical Security and Risk Management, Ph.D. Thesis, Stanford University (2002)