Analytical Models for Risk-Based Intrusion Response*

Bugra Caskurlu †
LDCSEE
West Virginia University
Morgantown, WV
{caskurlu@gmail.com}

Ashish Gehani ‡
Computer Science Laboratory
SRI
Menlo Park, CA
{ashish.gehani@sri.com}

Cemal Cagatay Bilgin
Life Sciences Division
Lawrence Berkeley National Lab
Berkeley, CA
{ccbilgin@lbl.gov}

K. Subramani §
LDCSEE
West Virginia University
Morgantown, WV
{ksmani@csee.wvu.edu}

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 first show that algorithmic underpinnings of optimal risk management can be formulated as the Partial Vertex Cover (PVC) problem in bipartite graphs. We then experimentally compare several heuristics and a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm we designed for the problem.

*A preliminary version of this work have appeared in LNCS 7000 with the title "Algorithmic Aspects of Risk Management". This version is not only an extended form of the preliminary version, but also both the introduction of the design patterns and their experimental comparisons are completely new.

†This research has been supported in part by the National Science Foundation through Award CNS-0849735.

‡This research was supported in part by the Air Force Office of Scientific Research through Award FA9550-12-1-0199.

§This research was supported in part by the National Science Foundation through Awards CCF-0827397 and CNS-0849735, and Air Force Office of Scientific Research through Award FA9550-12-1-0199.

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 under way, 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 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.

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 [Geh03] [Geh04] [GeK04], 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.

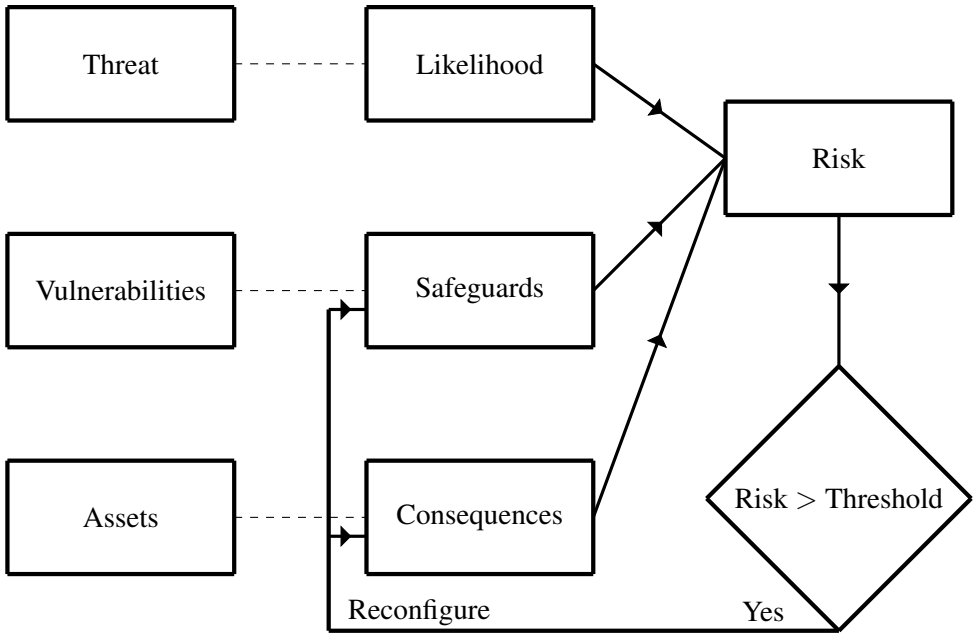


Figure 1: Risk can be analyzed as a function of the threats, their likelihood, vulnerabilities, safeguards, assets, and consequences. Risk can be managed by using the safeguards to control the exposure of vulnerabilities and manipulating the assets to limit the consequences.

2.1 Runtime Risk Factors

We model risk as the flow between the first and last partitions in a tripartite graph, depicted in Figure 2.1, where T is a partition of vertices t_i each representing 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 it occurring. If a signature is partially matched, the extent of the match predicts the chance that it will later be completely matched. A function μ is

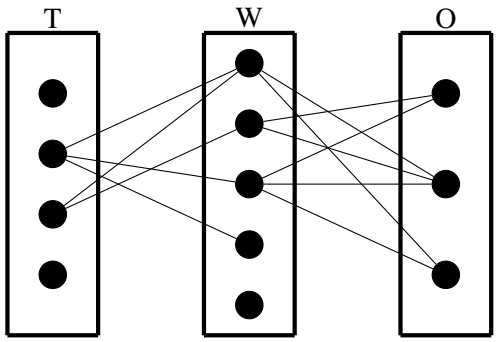


Figure 2: Operating system risk can be modeled in terms of its constituent components. The threats, weaknesses (corresponding to specific vulnerabilities), and objects (that are the assets) form three disjoint sets. An edge between vertices represents a contribution to the system risk. The system's risk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

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.

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* [GCK06]. 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 $P = \{p_1, p_2, \ldots\}$ safeguard 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* [GeK04][GK04] 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 [Nat96]. 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:

$$\begin{aligned} \text{minimize:} \quad & \zeta(\rho(\Omega(P)), \rho(\Omega(O))) \\ \text{subject to:} \quad & \mathcal{R}'' \leq \mathcal{R}_0 \end{aligned}$$

Risk management can be viewed as an exercise in picking vertices from the second and third partitions of Figure 2.1, 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 Statement of Problems

We formally define the graph-theoretic problems corresponding to our risk management model.

3.1 Integral Costs and Benefits

When performance-sensitive runtime risk management is viewed as a $0/1$ integer linear programming problem, 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 bipartite graph $G = \langle V, E, \vec{\mathbf{p}}, \vec{\mathbf{w}} \rangle$ with V denoting the set of vertices, E denoting the set of edges, $\vec{\mathbf{w}} : V \rightarrow Z$ denoting a weighting function from the vertices to the set of positive integers, and $\vec{\mathbf{p}} : E \rightarrow Z$ 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 bipartite graph $G = \langle V, E, \vec{\mathbf{p}}, \vec{\mathbf{w}} \rangle$ with V denoting the set of vertices, E denoting the set of edges, $\vec{\mathbf{w}} : V \rightarrow Z$ denoting a weighting function from the vertices to the set of positive integers, and $\vec{\mathbf{p}} : E \rightarrow Z$ 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 equivalent. 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 $\mathbf{P_2}$ by replacing T_2 with $\sum_{e \in E} p(e) - T_2$. Similarly, an instance of $\mathbf{P_2}$ can be represented as an instance of $\mathbf{P_1}$ by replacing T_2 with $\sum_{e \in E} p(e) - T_2$. An important point to note about $\mathbf{P_2}$ is that if a vertex in V does not have any incident edges, then it is automatically included in $V - V'$. In the rest of the text, we will refer to $\mathbf{P_2}$ as the WPVCB problem.

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. Each connected component of the corresponding graph is composed of two vertices and an edge in between. Optimal response selection in this scenario is algorithmically expensive as shown below:

Theorem 3.1 *WPVCB problem is* NP-hard *even if each connected component of G is composed of two vertices and an edge in between.*

Proof: We reduce the $0/1$ knapsack problem to the WPVCB problem. The knapsack problem is known to be NP-hard.

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.

Given the knapsack instance, we construct the following instance of the WPVCB problem. 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 w_i 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 WPVCB instance is.

Assume that the given knapsack instance is a "yes" instance – that is, there is a set of objects $O' \subseteq O$, such that $\sum_{y:y \in O'} w(y) \leq W$ and $\sum_{y:y \in O'} p(y) \geq T$. Pick only one the two vertices in the WPVCB instance that correspond to these objects. 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 WPVCB instance is a "yes" instance – that is, 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 T. Per the construction of the WPVCB instance, if vertex v_i is picked, then vertex v_{n+i} is not picked since these two vertices are incident on the same edge. Further, the contribution of this vertex to the vertex threshold is w_i and to the edge threshold is p_i. Consider the objects corresponding to the picked vertex pairs. Per the construction, their weights sum to at most W and their profits sum to at least T. ■□

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 S of vertices, subject to the constraint that the number of edges incident to S is above a predefined threshold T.*

In the rest of the text, we will refer to $\mathbf{P_3}$ as the PVCB problem.

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 S with the lowest sum of vertex weights, subject to the constraint that the number of edges incident to S is above a predefined threshold T.*

In the rest of the text, we will refer to $\mathbf{P_4}$ as the VPVC problem.

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 smallest set S of vertices, subject to the constraint that the sum of the weights of the edges incident to S is above a predefined threshold T.*

In the rest of the text, we will refer to $\mathbf{P_5}$ as the EPVCB problem.

4 Related Work

The effort to manage the risk of information systems can be traced to the use of the Annual Loss Expectancy (ALE) metric [FIP74][FIP79] by large data processing centers. The use of the ALE paradigm by commercial tools [Nat91a], coupled with a focused research effort [Nat88][Nat89][Nat90][Nat91b], resulted in improvements in risk modeling. Although risk analysis has been used to manage the security of systems for several decades [FIP74], its use has been limited to offline risk computation and manual response. SooHoo [Soo02] proposed a general model using decision analysis to estimate computer security risk and automatically update input estimates. Bilar [Bil03] 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. In this paper, we modeled the risk management partial vertex cover problem in bipartite graphs allowing weights in both the vertices and the edges.

In the classical Weighted Vertex Cover (WVC) problem, we are given an undirected graph $G = \langle V, E, \vec{\mathbf{w}} \rangle$, where V is the vertex set with $|V| = n$, E is the edge set with $E = m$, and $\vec{\mathbf{w}}$ is the vector of positive weights on the vertices. We want to find a minimum weight subset $V' \subset V$ such that for every edge $e = (i, j) \in E$, either $i \in V'$ or $j \in V'$. The WVC problem is one of the classical *NP-complete* problems listed by Karp [Kar72]. There are several polynomial-time approximation algorithms for the WVC problem within a factor of 2, and

the best-known approximation algorithm for the WVC problem has an approximation factor of $2 - \theta\left(\frac{1}{\sqrt{\log n}}\right)$ [Kar09]. The WVC problem is known to be *APX-complete* [PY91]. Moreover, it cannot be approximated to within a factor of 1.3606 unless *P = NP* [DS05] and not within any constant factor smaller than 2, unless the *unique games conjecture* is false [KR08].

In the Weighted Partial Vertex Cover (WPVC) problem, we are given an undirected graph $G = \langle V, E, \vec{\mathbf{w}} \rangle$, where V is the vertex set with $|V| = n$, E is the edge set with $|E| = m$, and $\vec{\mathbf{w}}$ is the vector of positive weights on the vertices, and a positive number T. We want to find a minimum weight subset $V' \subset V$ such that V' *covers* at least T edges – that is, for at least T edges $(i, j) \in E$, either $i \in V'$ or $j \in V'$. It is trivial to observe that the WPVC problem is a generalization of the WVC problem, since the WPVC problem subsumes the WVC problem for $T = m$. Therefore, all the hardness results given above for the WVC problem directly apply to the WPVC problem.

The WVPC problem has been extensively studied for more than a decade (see [BYFMR07]) and the references therein). In particular, there is a $O(n \cdot \log n + m)$-time primal-dual 2-approximation algorithm[Mes09], a combinatorial 2-approximation algorithm [BYFMR07], and a slightly improved $2 - \theta\left(\frac{\ln \ln d}{\ln d}\right)$-approximation algorithm [HS02], where d is the maximum degree of a vertex in G.

Although the WVC problem and the WPVC problem have almost matching approximation ratios and inapproximability results, the WPVC problem is in some sense more difficult than the WVC problem. For the cardinality versions of the problems – that is, all vertex weights are 1, the WPVC problem was shown to be W[1]-complete, while the WVC is fixed parameter tractable [GNW05].

In this paper, we focus our attention to bipartite graphs and give a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm for the WPVC problem on bipartite graphs, which we refer to as the WPVCB problem.

5 Bounded-Error Approximation Algorithm

In this section, we present a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm for the WPVCB problem. Since the VPVCB, EPVCB, and PVCB problems are the special cases of the WPVCB problem, we have a $(1+\frac{\sqrt{2}}{2}+\epsilon)$-approximation algorithm for all four of the problems considered in this paper.

Theorem 5.1 *There exists a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm for the WPVCB problem.*

The rest of the section is devoted to prove Theorem 5.1 by giving a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm for the WPVCB problem. The approximation algorithm is composed of three steps, which are presented in Section 5.1, 5.2, and 5.3.

5.1 Preprocessing Step

Recall that a WPVCB instance is composed of a bipartite graph $G = \langle V, E, \vec{\mathbf{p}}, \vec{\mathbf{w}} \rangle$, and a threshold T. In the optimization version of the WPVCB problem, the goal is to find a minimum weight subset S of vertices such that the sum of the weights of the edges incident to a vertex of S is at least a given threshold T. For a given WPVCB

instance, we first obtain a upper-bound B for the optimal solution to the WPVCB instance. There are various ways to obtain an upper-bound; for instance, taking the solution returned by the 2-approximation algorithm given in [Mes09]. Let $N \subset V$ be the set of vertices of G, whose weights are at least $\epsilon \cdot B$. Since B is an upper-bound for the optimal solution to the given WPVCB instance, an optimal solution to the given WPVCB instance will have at most $\frac{1}{\epsilon}$ of the elements of N for any $\epsilon > 0$.

In the preprocessing step, we enumerate all subsets of N, whose cardinalities are at most $\frac{1}{\epsilon}$. It is trivial to establish that there are at most $O(n^{\frac{1}{\epsilon}})$ such subsets. For each such subset P of N, we construct a different problem as explained below. We will solve all these $O(n^{\frac{1}{\epsilon}})$ problems and return the cheapest solution.

Problem for Subset P. For every vertex $i \in P$, we add vertex i to the solution and therefore, delete vertex i and all its incident edges from the graph G. We construct an Integer Programming (IP) formulation given below as System 1 for the WPVCB instance on the *remaining graph* $G' = \langle V', E', \vec{\mathbf{p}}, \vec{\mathbf{w}} \rangle$. Notice that $V' = V - P$, and $E' \subset E$ is the edges with both end-points in $V - P$. For all vertices of N that are not in P, we force the corresponding variable in the IP to be 0 so that our algorithm will never select them in the later stages by adding an extra constraint to the IP. This makes sure that the only heavy weight vertices (the elements of N) will be the elements of P.

$$
\begin{aligned}
\min \sum_{i \in V'} w(i) \cdot x_i & \\
f_{ij} \leq x_i + x_j \quad & \forall (i,j) \in E' \\
f_{ij} \leq 1 \quad & \forall (i,j) \in E' \\
x_i = 0 \quad & \forall i \in N - P \\
\sum_{(i,j)=e \in E'} f_{ij} \cdot p(e) \geq K & \\
x_i \in \{0,1\} \qquad & \forall i \in V'
\end{aligned} \tag{1}
$$

In the IP given as System 1, we have a variable x_i corresponding to each vertex $i \in V'$. We have a variable f_{ij} for every edge $(i,j) \in E'$ that keeps track of whether (i,j) is covered or not. The first two constraints and the fourth constraint ensure that $f_{i,j} = \max\{1, x_i + x_j\}$. Third constraint ensures that no element of $N - P$ will be selected in later stages of the algorithm. In the fourth constraint of the IP, we have an integer K which denotes the sum of the weights of the edges to be covered. Notice that K is not necessarily equal to T since the edges incident to a vertex in P are already erased. So, $K = T - \sum_{e=(u,v);\ u \in P \text{ or } v \in P} p(e)$.

We then solve our problem for all of the $O(n^{\frac{1}{\epsilon}})$ instances construction of which is explained in detail above. The rest of the algorithm is executed in all of these instances of the problem and the cheapest solution is reported at the end. Notice the the cost of a solution is *not* just the cost of the solution to the IP we will obtain since we need to take the weights of the vertices in P as well. Therefore, the cost of the solution for the subset P is the sum of the weights of the vertices of P plus the sum of the weights of the vertices in $V' - N$ that will be selected in the later stages of the algorithm.

Once we have constructed the integer programs for all $O(n^{\frac{1}{\epsilon}})$ problems, we continue with the second step of our algorithm. The second phase of the algorithm is composed of usual iterative rounding steps as explained in Section 5.2.

5.2 Iterative Rounding Step

This subsection is devoted to the second phase of the algorithm. This phase of the algorithm is composed of a series of iterative rounding steps. We first solve the linear programming relaxation (LP) of the IP given above. If there exists a variable x_i such that $x_i \geq 2 - \sqrt{2}$, we round this variable to 1. Rounding a variable to 1 means adding the corresponding vertex into the partial cover (which initially contains only the vertices of P) and deleting that vertex as well as all its incident edges from the graph. At this point, one step of the iterative rounding algorithm is fulfilled and we obtained a smaller graph. We will then repeat the same procedure on this smaller graph. Notice that after each rounding step the value of K is to be reduced as much as the sum of the weights of the erased edges in that step.

If K becomes 0 or less, then the set of vertices added to the cover so far by the algorithm constitutes a feasible partial cover and the algorithm terminates for the subset P. Since the algorithm only rounds the variables that has a value of $2 - \sqrt{2}$ or more, the solution returned will be within a factor of $\frac{1}{2-\sqrt{2}} = 1 + \frac{\sqrt{2}}{2}$ of the optimum integral solution to the initial IP as desired.

Since the iterative rounding algorithm adds one vertex to the partial cover at each step as long as we did not obtain a feasible solution and the optimal solution to the LP relaxation has a variable x_i such that $x_i \geq 2 - \sqrt{2}$, we do not obtain a $(1+\frac{\sqrt{2}}{2})$-approximate solution during the iterative rounding step if and only if all the variables of the optimal solution of the LP relaxation are less that $(2 - \sqrt{2})$ and $K > 0$. Notice that the LP relaxation may have more than one optimal solution and even though the optimal solution we found does not have a variable that has a value of $(2 - \sqrt{2})$ or more, there may be another optimal solution, where one of the variables has a value of $(2 - \sqrt{2})$ or more. However, it is trivial to answer the following query in polynomial-time: "For a given variable x_i, is there a fractional optimal solution to the given LP relaxation, where x_i is $(2 - \sqrt{2})$ or more?" If the answer to this query is "Yes", then we will continue the iterative rounding algorithm by rounding that variable to 1. If the answer to this query is "No", then we are done with the iterative rounding step.

If we did not obtain an integral solution at the end of the iterative rounding step, we obtain a graph G'' such that the sum of the weights of the edges of G'' is more than $\sum_{e \in E} p(e) - T$, and no variable can have a value of $(2 - \sqrt{2})$ or more in any optimal solution to the LP relaxation of System 1 for G''. In that case, we execute the third phase of our algorithm, which is given in Section 5.3 to obtain an integral solution to the problem we generated for the subset P.

5.3 Main Step of the Algorithm

In this subsection, we describe the third phase of our algorithm in detail. Observation 5.1 gives us the structure of the fractional optimal solution to System 1 on G''.

Observation 5.1 *Assume that we have not obtained a $(1+\frac{\sqrt{2}}{2})$-approximate solution at the end of the iterative rounding step, and end up with a subgraph G'' such that the sum of the weights of the edges of G'' is more than $\sum_{e\in E} p(e) - T$. There exists an optimal fractional solution to System 1 on G'' that has the following properties:*

- *The value of each variable x_i is strictly less than $(2-\sqrt{2})$;*
- *There exists at most one variable x_i that has a nonzero value which is $(\sqrt{2}-1)$ or less.*

Proof: The first property listed in the Observation is trivial since it is the stopping condition for the iterative rounding step, i.e., if there is a variable with a value of $(2-\sqrt{2})$ or more, then the iterative rounding algorithm would not terminate and continue by rounding that variable up.

For the purpose of contradiction, assume that in all of the optimal fractional solutions of the LP relaxation of System 1 on G'', there are 2 or more variables that have a nonzero value of $(\sqrt{2}-1)$ or less. Let S^* be an arbitrary optimal solution to the LP relaxation and let M be the set of such variables.

Lemma 5.1 *Let i and j be two distinct elements of M. Then, $\frac{|\delta(i)|}{w(i)} = \frac{|\delta(j)|}{w(j)}$, where $|\delta(i)|$ and $|\delta(j)|$ denote the sum of the weights of the incident edges of the vertices i and j respectively.*

Proof: Without loss of generality, assume $\frac{|\delta(i)|}{w(i)} > \frac{|\delta(j)|}{w(j)}$. Let $\epsilon = \min\{\left(\sqrt{2}-1-x_i\right), x_j\frac{|\delta(j)|}{|\delta(i)|}\}$. Let S be the following solution to LP:

$$S(x_k) = \begin{cases} S^*(x_i)+\epsilon, & \text{if } k=i \\ S^*(x_j)-\epsilon\frac{|\delta(i)|}{|\delta(j)|}, & \text{if } k=j \\ S^*(x_k), & \text{otherwise} \end{cases}$$

S is a feasible solution to the LP relaxation. Since $\sum_{(i,j)=e\in E''} f_{ij}$ is the same both in S^* and S. However, the objective function value of solution S is strictly less than the objective function value of the solution S^*. This is a contradiction since S^* is an optimal solution to the LP relaxation. □

In the proof of Lemma 5.1, we obtained a solution cheaper than S^* because we assumed $\frac{|\delta(i)|}{w(i)} > \frac{|\delta(j)|}{w(j)}$. Notice that if $\frac{|\delta(i)|}{w(i)} = \frac{|\delta(j)|}{w(j)}$ (which is true by Lemma 5.1), the solution S constructed would have the same cost as S^*. Therefore, starting with S^*, we can obtain a different optimal solution S to the LP relaxation by applying the operation given in the proof of Lemma 5.1. Notice that either $S(x_i) = \sqrt{2}-1$, or $S(x_j) = 0$. So, the operation reduces $|M|$ by 1 at each step. We will repeat this operation until $|M| = 1$ which satisfies the properties given in the Observation 5.1. □

Recall that there exists a solution to the IP, where each variable f_{ij} is either 0 or 1. This is not true for the LP relaxations. Therefore, we introduce the following definitions.

Definition 5.1 *An edge (i,j) is called a tight edge if $x_i + x_j \geq 1$. We call (i,j) a fractionally covered edge if $x_i + x_j < 1$.*

Notice that since there is no variable that is $2-\sqrt{2}$ or more, for any edge $(i,j) \in E''$, if (i,j) is a tight edge in the fractional solution, we have both x_i and x_j are more than $(\sqrt{2}-1)$. Therefore, we have a bunch of nodes that has a value in the interval $(\sqrt{2}-1, 2-\sqrt{2})$. All other variables are between 0 and $\sqrt{2}-1$. By the arguments

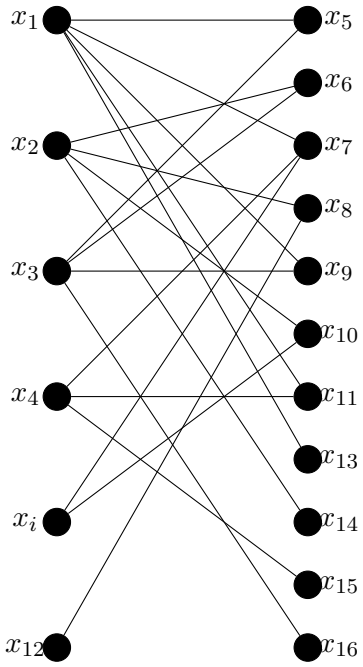


Figure 3: The bipartite graph shows our graph G''. The variables x_1, x_2, x_3 and x_4 form the set A. The variables $x_5, x_6, x_7, x_8, x_9, x_{10}$ and x_{11} form the set B. The variables x_i, x_{12}, x_{13} and x_{14} form the set C. Notice that an edge can be fully covered if it has one endpoint in A and one endpoint in C.

given above as Observation 5.1, we can ensure that at most one of the variables whose value is between 0 and $\sqrt{2}-1$ is nonzero. Let this variable be x_i.

We can classify the variables into 3 sets, namely A, B and C. Let C be the set of variables that are 0 and x_i. Let A be the variables of the left partition that have a value in the interval $(\sqrt{2}-1, 2-\sqrt{2})$. Let B be the variables of the right partition that have a value in the interval $(\sqrt{2}-1, 2-\sqrt{2})$.

Let k be the sum of the weights of the variables that are either in A or B. Since all the variables in A and B are at least $\sqrt{2}-1$, the objective function value of the LP is at least $(\sqrt{2}-1)k$. In order to obtain a $1+\frac{\sqrt{2}}{2}$-approximation algorithm, it suffices to show that we can cover *enough* edges by selecting vertices such that the sum of the weights of the selected vertices are at most $(1+\frac{\sqrt{2}}{2})(\sqrt{2}-1)k = (\frac{\sqrt{2}}{2}k)$.

It is *enough* to cover the following set of edges:

- All the edges between A and B,
- All the edges incident to x_i,
- A *large enough* subset of the remaining edges (j, l) such that $j \in A \cup B$ and $l \in C$. We call those edges the Type 3 edges. A subset of the Type 3 edges is large enough for our purposes if the sum of the weights of this subset of edges is at least $(2-\sqrt{2})$ times the sum of the weights of all the Type 3 edges.

We select the set of vertices for our partial cover as follows:

We add the vertex i to the partial cover. The selection of the rest of the vertices depends on the relation between $\sum_{j\in A} w(j)$, $\sum_{j\in B} w(j)$, and k. If both $\sum_{j\in A} w(j)$ and $\sum_{j\in B} w(j)$ are less than $\frac{\sqrt{2}}{2}k$, we select all the vertices of A if S covers more Type 3 edges than U per weight. Otherwise, we select all the vertices of B. Without loss of generality, assume A covers more Type 3 edges than B per weight and so we selected all the vertices of A. We then sort the vertices of B in descending order with respect to the number of Type 3 edges they cover per weight. We add them into our partial cover until *enough* Type 3 edges are covered.

We next consider the case where either $\sum_{j\in A} w(j)$ or $\sum_{j\in B} w(j)$ is bigger than $\frac{\sqrt{2}}{2}k$. Without loss of generality, assume $\sum_{j\in A} w(j) > \frac{\sqrt{2}}{2}k$. We then add all the vertices of B to our partial cover. We then sort the vertices of A in descending order with respect to the Type 3 edges they cover per weight. We add them into our partial cover until enough Type 3 edges are covered.

The sum of the weights of the added vertices to the partial cover in the main step of the algorithm is not necessarily bounded above by $\frac{\sqrt{2}}{2}k$. However, if we exclude the vertex i and the last vertex added to the cover, the sum of the weights of the rest of the vertices added to the partial cover in the main step of the algorithm is bounded above $\frac{\sqrt{2}}{2}k$. Therefore, we added to the partial cover a set of vertices whose total weight is at most $\frac{\sqrt{2}}{2}k$ plus an additional 2 vertices. Notice that none of these 2 vertices is an element of N and therefore, the ratio of their weight to the optimal solution is bounded by ϵ.

Therefore, our algorithms returns a $(1 + \frac{\sqrt{2}}{2} + 2\epsilon)$-approximate solution for at least one of the problems constructed in the preprocessing step. If we select ϵ as half of what it is then the solution will be a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximate solution.

6 Heuristic Approaches

In this section, we discuss the following 3 heuristic approaches. For ease of explanation, we assume that we want to delete a small subset of vertices such that the sum of the weights of the remaining edges is T.

6.1 Lightest Vertex Out Algorithm

In this subsection, we discuss our next design approach which we call the *Lightest Vertex Out Algorithm*. Since the deletion of vertices is reducing the functionality of the system, and we seek a solution that reduces the functionality of the system as small as possible, in this design approach we delete the vertices one-by-one in the ascending order with respect to their weight until the risk tolerance level is reduced below the threshold level, i.e., the sum of the weights of the remaining edges is at most T.

6.2 Heaviest Edge Out Algorithm

In this subsection, we discuss our next design approach which we call the *Heaviest Edge Out Algorithm*. Since we want to reduce the sum of the weights of the remaining edges below T, in this design approach we delete

Function LIGHTEST WEIGHT VERTEX OUT ALGORITHM()

1: **while** $\sum_{e \in E} p(e) \geq T$ **do**
2: Let i be the vertex with the smallest weight.
3: Remove i and all its incident edges.
4: **end while**

Algorithm 6.1: Lightest Vertex Out Algorithm

one of the the incident nodes of the edge with the highest weight and repeat this procedure until the sum of the weights of the remaining edges is below T. Since each edge is incident to 2 vertices, at each step of the algorithm we are left with the choice of one of the two incident vertices of the edge we want to remove. The algorithm always selects the incident vertex with the smallest weight.

Function HEAVIEST EDGE OUT ALGORITHM()

1: **while** $\sum_{e \in E} p(e) \geq T$ **do**
2: Let (i, j) be the edge with the highest weight.
3: **if** $w(i) \leq w(j)$ **then**
4: Remove i and all its incident edges.
5: **else**
6: Remove j and all its incident edges.
7: **end if**
8: **end while**

Algorithm 6.2: Heaviest Edge Out Algorithm

6.3 Best Return for Cost Algorithm

In this subsection, we discuss our last design approach which we call the *Best Return for Cost Algorithm*. Since we want to reduce the sum of the weights of the remaining edges below T, by deleting a set of vertices such that the sum of the weights of the vertices deleted is as small as possible, in this design approach we delete the vertex with the highest return for cost and repeat this procedure until the the sum of the weights of the remaining edges is below T.

Function BEST RETURN FOR COST ALGORITHM()

1: Let $|\delta(i)|$ denote the sum of the weights of the incident edges of each vertex i.
2: **while** $\sum_{e \in E} p(e) \geq T$ **do**
3: Let i be the vertex for which $\frac{|\delta(i)|}{w(i)}$ is maximized.
4: Remove i and all its incident edges.
5: **end while**

Algorithm 6.3: Best Return for Cost Algorithm

7 Empirical Study

To demonstrate the utility of our algorithms in a wide range of conditions, we compare the performance of our design approaches on randomly generated bipartite graphs. We produced random bipartite graphs using the Erdos Renyi random graph generation. In this model, a random bipartite graph is given by $G(S_1, S_2, p)$ where S_1 and S_2 are the first and second bipartite sets respectively and p is the probability of setting an edge between a vertex in S_1 and a vertex in S_2 . Edge weights and node weights are both drawn from an independent uniform distribution in the interval $[0, 1]$. Each experiment is repeated 10 times and the average cost for each algorithm is reported. The aforementioned algorithms are implemented in Matlab using the optimization toolbox where necessary.

The Lightest Vertex Out (*LVO*) heuristic requires the vertices to be weighted, whereas the Heaviest Edge Out (*HEO*) heuristic requires the edges to be weighted.

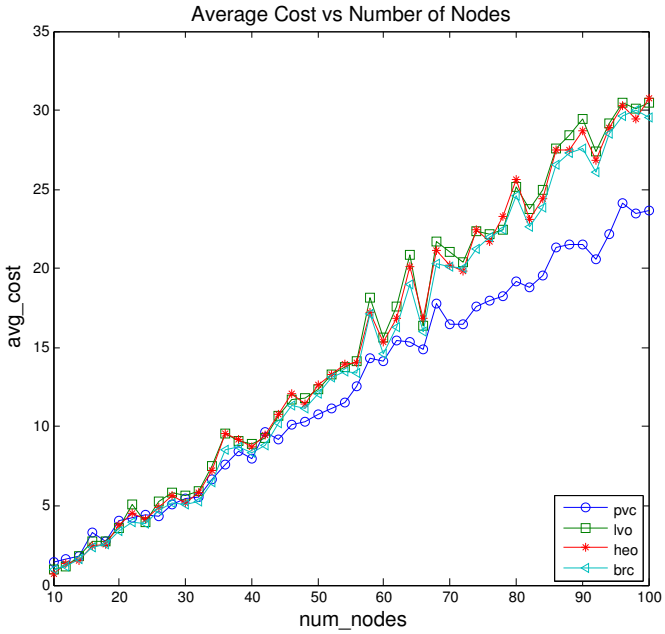
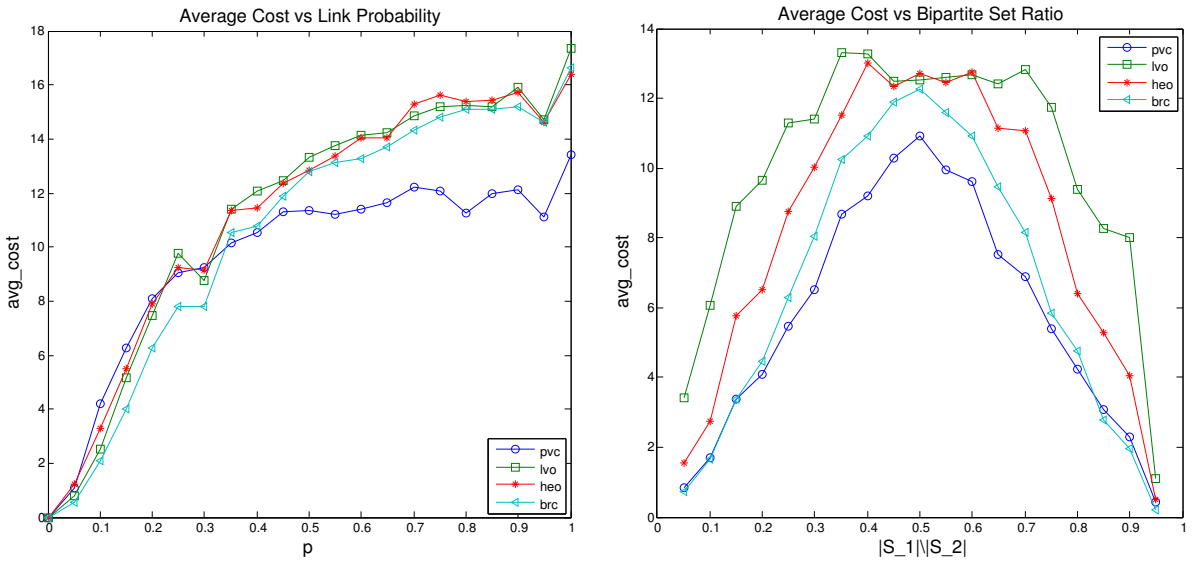


Figure 4: Effect of the size of the random graph $G(S_1, S_2, p)$ where $|S_1| = |S_2| = N/2$ and $p = 0.5$ is depicted. APP, LVO, HEO and brc stands for the $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm, Lightest Vertex Out algorithm, Heaviest Edge Out algorithm and Best Return for Cost algorithm, respectively. The $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm presented in this paper scales better than the other algorithms.

We start our experiments by considering the effect of the increasing number of nodes and comparing the average costs of the given algorithms. In each experiment we picked $T = 0.25 * num\_nodes$ and considered the problem of covering T edges. The first experiment involves the effect of size of the random bipartite graph of the form $G(S_1, S_2, p)$ where $|S_1| = |S_2| = N/2$ and $p = 0.5$ with varying N. In Figure 4, we observe that the heuristics Lightest Vertex Out (*LVO*), Heaviest Edge Out (*HEO*) and Best Return for Cost (*BRC*) all perform similarly, whereas the $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm presented in this paper scales better as the number of nodes in the solution increases.



(a) Link probability vs cost

(b) Bipartite ratio vs cost

Figure 5: The effect of edge probability on the cost for a graph of size 100 with equal bipartite set cardinalities, $|S_1| = |S_2|$ is depicted in Figure 5(a). As the link probability increases, the graph becomes denser. Effect of the ratio between the bipartite sets on the cost for a graph of size 100 with $p = 0.5$ is shown in Figure 5(b). APP, LVO, HEO and BRC stand for the $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm, the Lightest Vertex Out algorithm, the Heaviest Edge Out algorithm, and the Best Return for Cost algorithm, respectively.

We also consider the effect of the probability of setting a link between the vertices by varying p for a graph of size 100 with $|S_1| = |S_2|$. As the probability p increases, the graph becomes denser. We observe that the $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm presented in this paper performs better than the heuristic algorithms for all the p values considered (Figure 5(a)).

For the same-sized random graphs, $N = 100$, with probability of building an edge $p = 0.5$, we also vary the ratio between the bipartite sets, $R = S_1/S_2$ and plot the total cost as a function of R. We repeat this experiment 10 times as well to account for the randomness and report the average total cost in Figure 5(b). We see that the cost of the WPVCB problem depends on the balance of the bipartite sets. When the sets have similar sizes, the cost of the solution increases as well. Like previous experiments, we see a similar performance gain with the $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm when compared to the aforementioned heuristics.

8 Conclusion

We studied the problem of managing the risk of information systems. We presented a model that uses the computation of risk to derive changes in the security posture of a system. Our model corresponds to a slight generalization of the well-known Partial Vertex Cover problem on bipartite graphs. We provided several design approaches and an experimental comparison of our design approaches on instances of the problem.

Our future work for our intrusion response problem is twofold. The model described in this paper is nonadap-

tive – that is, each time we are given a risk tolerance level we recompute a solution that reduces the risk level of the system below the risk tolerance level. As the first future work direction, we want to introduce the adaptive version of this problem. In the adaptive version, there is a cost associated with changing the encapsulation status of an asset as well as a cost with changing the exposure status of a permission. Thereby, our algorithms will favor a solution that does not change the security posture much from the previous configuration.

As our second future research direction, we want to establish the computational complexity of the Partial Vertex Cover problem on bipartite graphs. Although the Vertex Cover problem is known to be polynomial time solvable on bipartite graphs, the computational complexity of the Partial Vertex Cover problem on bipartite graphs is currently unknown. Our research will also focus on obtaining approximation algorithms for the Partial Vertex Cover problem on bipartite graphs with a provable bound less than $(1 + \frac{\sqrt{2}}{2})$. In this paper, we presented a $(1 + \frac{\sqrt{2}}{2} + \epsilon)$-approximation algorithm for the Partial Vertex Cover problem on bipartite graphs, but the question of whether there is a polynomial time algorithm that guarantees a better solution on all instances of the problem has both theoretical and practical (as demonstrated by this paper) merits for research.

References

[Bil03] Daniel Bilar. *Quantitative Risk Analysis of Computer Networks*. Ph.D. Thesis, Dartmouth College, 2003.

[BYFMR07] Reuven Bar-Yehuda, Guy Flysher, Julián Mestre, and Dror Rawitz. Approximation of partial capacitated vertex cover. *15th Annual European Symposium on Algorithms*, *Lecture Notes in Computer Science*, Volume 4698, pages 335–346, Springer, 2007.

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, Volume 162(1), pages 439–485, 2005.

[FIP74] Guidelines for automatic data processing physical security and risk management, 1974.

[FIP79] Guidelines for automatic data processing risk analysis, 1979.

[GCK06] Ashish Gehani, Surendar Chandra, and Gershon Kedem. Augmenting storage with an intrusion response primitive to ensure the security of critical data. *1st ACM Symposium on Information, Computer and Communications Security*, 2006.

[Geh03] Ashish Gehani. Support for Automated Passive Host-based Intrusion Response. *Ph.D. Thesis*, Duke University, 2003.

[Geh04] Ashish Gehani. Performance-sensitive real-time risk management is NP-hard. *Workshop on Foundations of Computer Security*, affiliated with the *19th IEEE Symposium on Logic in Computer Science*, 2004.

[GeK04] Ashish Gehani and Gershon Kedem. RheoStat: Real-time Risk Management. *7th International Symposium on Recent Advances in Intrusion Detection*, 2004.

[GK04] Ashish Gehani and Gershon Kedem. Real-time access control reconfiguration. *International Infrastructure Survivability Workshop*, affiliated with the *25th IEEE International Real-Time Systems Symposium (RTSS)*, 2004.

[GNW05] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of generalized vertex cover problems. *9th WADS*, *Lecture Notes in Computer Science*, Volume 3608, pages 36–48. Springer, 2005.

[HS02] Eran Halperin and Aravind Srinivasan. Improved approximation algorithms for the partial vertex cover problem. *APPROX*, *Lecture Notes in Computer Science*, Volume 2462, pages 161–174, Springer, 2002.

[Kar72] Richard Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, Plenum Press, 1972.

[Kar09] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms*, Volume 5(4), 2009.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal Computer System Science*, Volume 74, pages 335–349, May 2008.

[Mes09] Julin Mestre. A primal-dual approximation algorithm for partial vertex cover: Making educated guesses. *Algorithmica*, Volume 55(1), pages 227–239, 2009.

[Nat88] National Bureau of Standards. *1st Computer Security Risk Management Model Builders Workshop*, Martin Marietta, Denver, Colorado, May 1988.

[Nat89] National Institute of Standards and Technology. *2nd Computer Security Risk Management Model Builders Workshop*, AIT Corporation, Ottawa, Canada, June 1989.

[Nat90] National Institute of Standards and Technology. *3rd Computer Security Risk Management Model Builders Workshop*, Los Alamos National Laboratory, Santa Fe, New Mexico, August 1990.

[Nat91a] National Institute of Standards and Technology. *Description of Automated Risk Management Packages That NIST/NCSC Risk Management Research Laboratory Has Examined*, 1991.

[Nat91b] National Institute of Standards and Technology. *4th Computer Security Risk Management Model Builders Workshop*, University of Maryland, College Park, Maryland, August 1991.

[Nat96] National Institute of Standards and Technology. *Guidelines for Automatic Data Processing Physical Security and Risk Management*, 1996.

[PY91] Christos Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, Volume 84, pages 127–150, July 1991.

[Soo02] Kevin SooHoo. *Guidelines for Automatic Data Processing Physical Security and Risk Management*. Ph.D. Thesis, Stanford University, 2002.