Interdicting Attack Graphs to Protect Organizations from Cyber Attacks: A Bi-Level Attacker-Defender Model

Author information blinded

Abstract

Today's organizations are inherently open and connected, sharing knowledge and ideas in order to remain innovative. As a result, these organizations are also more vulnerable to information theft through different forms of security breaches caused by hackers and competitors. One way of understanding the vulnerability of an information system is to build and analyze the attack graph of that system. The attack graph of an information system contains all the paths that can be used to penetrate the system in order to breach critical assets. Although existing literature provides an abundance of attack graph generation algorithms, more methods are required to help analyze the attack graphs. In this paper, we study how best to deploy security countermeasures to protect an organization by analyzing the vulnerability of the organization through the use of its attack graph. In particular, we present an approach to find an optimal affordable subset of arcs, called an interdiction plan, on an attack graph that should be protected from attack to minimize the loss due to security breaches. We formulate this problem as a bi-level mixed-integer linear program and develop an exact algorithm to solve it. Experiments show that the algorithm is able to solve relatively large problems. Two heuristic methods, one with and the other without a heuristic to solve the master problem and both limiting the master problem branch-and-bound tree to only one node solve the large problems remarkably well. Experiments also reveal that the quality of an interdiction plan is relatively insensitive with respect to the error in the estimate of the attacker's budget, and that the breach loss drops sharply at the beginning, then levels off before finally dropping sharply again with increases in the security budget.

Keywords: Attack graph, interdiction, bi-level, mixed-integer-programming, attacker-defender, cyber-security

1. Introduction

In order to increase operational efficiency and functionality, organizations, individuals, and devices are becoming more and more connected, spawning new phenomena such as the "Internet of Things." The result is the increasing vulnerability of information to theft and even disruption of services provided by critical infrastructures. According to the FBI Internet crime report for 2013 [14], more than 260,000 individuals reported complaints about their accounts being compromised with a total adjusted loss of more than \$781 million. This loss is an increase of 48.8% from \$581 million in 2012. In addition to the plethora of low-profile cyber attacks, there have been several recent high-profile attacks on corporations such as JPMorgan Chase & Co. and Target. The cyber attack on Target in 2013 resulted in the theft of 40 million credit card numbers and 70 million different pieces of personal information [34], and the attack on JPMorgan in 2014 compromised information from about 76 million households and 7 million small businesses [42]. Therefore, maximizing the security of cyber systems is becoming one of the most important tasks of information technology teams in many organizations.

The security of a network can be enhanced by hardening selected components of the network in various ways, such as deploying firewalls and intrusion prevention and/or detection systems, finding and patching vulnerabilities, and making configuration changes. One of the tools for analyzing the security of a network and finding ways to deploy countermeasures is the *attack graph*, which contains all the paths that can be used to penetrate a network to breach critical nodes (goal nodes). In this paper, we develop models and algorithms based on the attack graph of an organization to optimally deploy security countermeasures to protect its information assets.

Phillips and Swiler [32] first proposed attack graphs as an analysis tool. Attack graphs can be generated manually or automatically using the following inputs: a database of common attacks broken into atomic steps, information on network topology and configuration, and an attacker profile [32]. Swiler et al. [39] later proposed an automatic attack graph generation tool. Since then, many researchers have studied attack graphs and proposed a multitude of attack graph variations including attack trees [43], attack countermeasure trees [35], defense trees [9], and exploit dependency graphs [30]. For a detailed review of many other attack graph variations, refer to the studies by Lippmann and Ingols [23] and Kordy et al. [21]. Regardless of the type, an attack graph can be aggregated to different levels such as a graph of hosts (computers or devices) or a graph of subnets using different underlying network regularities [29]. As a result, the models and algorithms developed in this paper can be also applied with minor modifications to the different levels of aggregated graphs.

Our work complements the existing information security research on attack graph cut set generation. Based on the specific study, a cut set, which is also a minimum cut set, is usually defined as the set of initial security conditions, a set of exploits, or a set of arcs in general; the removal or hardening of such a cut set removes all the paths to the goal nodes. One implicit assumption behind cut set generation is that the defender has enough budget to deploy countermeasures on all the members in the cut set. However, the defender of a network almost always operates with a limited budget in real life. Therefore, in this paper, we relax the assumption of unrestricted budget and find a subset of arcs that removes only the most important subset of paths to the goal nodes to minimize the potential damage in the residual network. Albanese et al. [1] propose a formal cost model that estimates the cost if a critical node is breached. Their cost model can be used to select the minimum set of initial security conditions, which upon removal, also removes all the paths from initial conditions to the goal conditions. They also propose an approximation algorithm to find the minimum set. Alhomidi and Reed [3] use a genetic heuristic to find the minimum cut set in dependency attack graphs. This is one of the few studies that considers arc removal as opposed to node removal. The study by Jajodia et al. [19] is another one that uses arc removal for network hardening. Dewri et al. [12] propose a multi-objective optimization model to select a subset of security-hardening measures that minimizes the total damage and the total cost of security hardening. Their work is one of the few studies that constrain the total amount spent on security measures. Noel and Jajodia [29] develop a mechanism to cover (i.e., completely protect) an attack graph by placing the fewest number of Intrusion Detection Systems (IDS) sensors.

This work fits into the field of network interdiction, a subset of network optimization. The complete or partial removal of an arc is in general referred to as the *interdiction* of that arc in the network interdiction literature. The problem in this work is a network interdiction problem because the goal is to select a subset of arcs to be interdicted (completely removed). Our work extends the network interdiction literature by providing an effective new model and customized exact and heuristic algorithms motivated by an existing algorithmic framework, and then applying the model and algorithms in a new manner. Our model and algorithm can also be adapted for variations of existing network interdiction applications, such as interdiction of a nuclear weapons project to maximize the minimum completion time of the project [10, 33], interdiciton of an electric power grid to assess the vulnerability [36, 37], monitoring drinking water supply for quick detection of contamination [6, 7, 8, 26, 25, 41], interdiction in hazardous materials transportation to minimize the discharge damage and transportation cost [45], and interdiction of a nuclear smuggling network [24, 28, 31, 38]. From the modeling perspective, our work is most closely related to the work by Pan and Morton [31]; in both their work and our work, the network is interdicted to prevent some flow from reaching a set of destinations. However, Pan and Morton [31] interdict arcs in a nuclear material smuggling network to minimize a stochastic maximum reliability path, and our work interdicts arcs in a deterministic attack graph to minimize the maximum damage that can be caused through breach of critical assets (goal nodes) by an attacker. Although the selection of an origin destination pair by an attacker in their work is stochastic, they assume that the attacker selects only a single origin-destination pair in a specific realization. In our work, an attacker can use any one or more origins (initially vulnerable nodes) to initiate attacks and breach any one or more destinations (goal nodes).

In this paper, we study the strategic interaction between the defender of an organization and an attacker, modeling a two-player, sequential defender-attacker game with resource restrictions over an attack graph. Related to our work, Dewri et al. [13] propose a multi-objective optimization model that poses the interaction between the attacker and the defender as an arms race. They use a genetic heuristic to approximately solve the defender's problem and the attacker's problem, and they use competitive co-evolution to approximately solve the combined problem to find the Nash equilibrium. Zonouz et al. [46] propose a new automated response approach called the Response and Recovery Engine. In this engine, adversaries are modeled as opponents in a two-player Stackelberg stochastic game.

To the best of our knowledge, most of the existing literature related to cut set generation analyzes attack graphs composed of only one goal node. None of these studies considers multiple goal nodes that each have different costs of breach. Goal nodes are indeed different; for example, the loss due to the breach of a database server of an e-commerce portal that contains credit card information is probably not the same as the loss due to the breach of an internal mail server. None of these studies consider intermediate nodes as goal nodes. In reality, merging all the attack graphs of an organization produces intermediate goal nodes. Thus, inclusion of intermediate nodes as goal nodes can increase flexibility both in generation and analysis of attack graphs because the merged attack graphs can be analyzed without removing the intermediate goal nodes. There is a scarcity of rigorous mathematical models that analyze the attack graphs, especially the mathematical programming models, as most of the models are logic-based ad hoc models [11, 12]. There is also scarcity of literature that incorporates both the interest of the attacker and the interest of the defender in an attack graph setting. The existing studies are applicable only to the specific graph types used as the analysis platforms. Also, to the best of our knowledge, all of the proposed algorithms solving the defender-attacker games on attack graphs are either heuristic or simulation based. Since most of the studies develop methods to ensure complete security, only a few of the previous studies consider a limited budget for the defender, and none of the previous studies considers limited budgets for both the defender and the attacker.

In this paper, we develop an defender-attacker bi-level network interdiction model and formulate it using mixed-integer linear programming. We provide two alternative formulations for the inner problem. Our model has binary variables in both of the levels. As a result, there is no easy way to merge the two levels into a single-level formulation; thus, the bi-level formulation cannot be solved directly using a commonly used mixed-integer programming solver. Moreover, although bi-level programming models are aplenty in the network interdiction literature [10, 19], there are not many bi-level programming models with binary variales in both levels[10]. Therefore, we develop a customized exact algorithm to solve the model. Alderson et al. [2] provide a comprehensive discussion on the modeling and algorithmic strategies for quantifying the resilience of infrastructure systems to disruptive events. Our algorithm is based on the framework proposed by Brown et al. [10] and subsequently generalized by Alderson et al. [2]. However, our algorithm is different than theirs due to the different attacker problem (operational model in [2]) with the resultant unique attacker problem formulation. Moreover, the master problem formulation in the algorithm is also different than theirs becasue of the distinct attacker model. We also propose several enhancements to the base algorithm. Our path based formulation of the master problem is efficient mainly because it is implicitely able to exploit the tree structure of any solution of the attacker problem. We show through experimentation that our algorithm is capable of solving relatively large problems for different parameter combinations. We argue that with further enhancements to our algorithm, along with implementation of the existing graph simplification mechanisms [4, 17, 20, 22, 29], which reduces graph size by up to 99% in some cases, our algorithm has potential to solve even larger problems. We also show that both of the inner problem formulations perform well under specific parameter settings.

2. Problem Description

A node in an attack graph might represent several different entities including an attack state, a security condition, a vulnerability, or an exploit. On the other hand, an arc might represent a change of state caused by an atomic action or attack by an attacker, exploit, etc. If the nodes in an attack graph represent security conditions, the tail node of an arc is a pre-condition, and the head node of that arc is a post-condition of that pre-condition. In the attack graphs used in this paper, a node represents a security condition, and an arc represents an attacker action or an exploit. An attacker action on any one of the pre-condition nodes of a post-condition node is enough to activate the post-condition node. This is in contrast with some variations of attack graphs in which some of the post-conditions are connected to pre-conditions via an AND logical gate; that is, the attacker must take action on all or a specific subset of the pre-conditions to activate a post-condition.

Given an attack graph, an attacker tries to maximize the total reward by compromising as many high-value goal nodes as possible starting from the initially vulnerable (initial security condition) nodes. A single path from an initially vulnerable node to a goal node is an attack path, and one or more attack paths constitute an attack plan. The defender or owner of the network selectively thwarts a subset of attacker actions or disables a subset of exploits. In this paper, a subset of attacker actions or exploits is represented by a subset of arcs. The subset of arcs selected by the defender constitutes an interdiction plan. The defender observes the results of different interdiction plans and attempts to minimize the maximum total reward possible for the attacker by interdicting or protecting the best subset of the arcs. We refer to this as the *optimal interdiction plan* of the defender. The defender's and the attacker's problems are represented in the outer and inner levels, respectively, in the bi-level formulation of the problem, which we call the MINMAXBREACH problem. Both the attacker and the defender have limited budgets. Losses associated with different nodes might have different values. The same applies to the costs of attacks through different arcs and the costs of security countermeasures to interdict different arcs. We use protection and interdiction of an arc interchangeably in this paper. Interdicting an arc is the same as protecting it from being attacked. Moreover, when a goal node is breached, the attacker gains a reward, and the defender suffers a loss equal in magnitude to the reward. Thus, the terms reward and loss used for a goal node also refer to the same quantity. We also refer to an initially vulnerable node as a vulnerability node to be concise.

Let the attack graph be denoted as $G = (N, A)$, where N is the set of nodes, and A is the set of arcs. N is partitioned into three subsets: N_I is the set of vulnerability nodes, N_R is the set of transition nodes, and N_T is the set of goal nodes. Nodes in N_T can act also as transition nodes. An attacker initiates his attacks using one or more of the nodes in N_I, continues his attacks by visiting one or more transition nodes in N_R, and then culminates his attacks by reaching one or more goal nodes in N_T, upon which he receives some reward. We assume that after a goal node is breached once, complete damage is done, and no further reward can be gained by attacking it again. Thus, we do not allow attacking a goal node more than once in the model. To move from one node to another, the attacker must attack using the arc connecting the two nodes, incurring a cost to attack. However, he incurs this arc attack cost only once even if the arc is used on more than one attack path. The defender may also incur a cost to employ a countermeasure on an arc to protect it from being attacked.

Figure 1 shows an example attack graph. Information inside the nodes includes the labels of the nodes and the losses due to breach separated by a hyphen (e.g., the label "6-5" denotes node 6, which has a loss of 5 if breached). Salmon nodes with dashed outlines are the vulnerability nodes, light blue nodes are the transition nodes, and green nodes are the goal nodes. To make the example simple, assume that the cost of countermeasures and attacks is 1 for all of the arcs in this graph. Also, assume that the budgets of the attacker and the defender are 3 and 0, respectively. So, the attacker attacks without any interdiction by the defender. In this case, an optimal attack plan for the attacker is to attack goal nodes 5 and 7 using the attack paths {(0,3),(3,5)} and {(0,3),(3,7)}, resulting in a total reward of 35. Now, assume that the defender's budget is 1, and the defender interdicts arc (3,5). In this case, an optimal attack plan is to attack goal nodes 4 and 5 using the attack paths {(0,2),(2,4)} and {(0,2),(2,5)}, resulting in a total reward of 25. By interdicting arc (3,5), the defender minimizes the maximum total reward achievable by the attacker. Note that in the first attack plan, the attacker uses arc (0,3) to breach two goal nodes. However, the attacker incurs the arc attack cost only once to attack the goal nodes using this arc. The same applies for arc (0,2) for the second attack plan.

3. Mathematical Formulations

In this section, we formulate the MINMAXBREACH problem as a bi-level mathematical programming model. The outer level represents the defender, and the inner level represents the attacker. Table 1 lists the notation used in the rest of this paper.

The objective of the outer problem is to minimize the maximum reward achievable by the attacker. The outer problem is formulated as follows.

Table 1: Notation.

(a) Parameters

Parameters	Description
$L(i)$	Set of arcs leaving node i
$E(i)$	Set of arcs entering node i
c_t^b	Loss due to breach of a goal node $t \in N_T$
c_{ij}^d	Cost of deployment of countermeasures on arc $(i,j) \in A$
c_{ij}^a	Cost of using arc $(i,j) \in A$ in one or more attack paths
B_d	Defender's budget
B_a	Attacker's budget
M	Large number enforcing the upper bound on variables
c_p^b	Loss due to breach of a goal node through path p
A_p	Set of arcs in path p
P_k	Set of paths in attack (iteration) k

(b) Variables

Variables	Description
z_t	1 if goal node t is breached, 0 otherwise
z_i^k	1 if node i is breached at iteration k, 0 otherwise
x_{ij}	1 if countermeasures are deployed on arc (i,j), 0 otherwise
y_{ij}	Number of goal nodes attacked through arc (i,j)
y_{ij}^t	1 if goal node t is attacked using arc (i,j), 0 otherwise
w_{ij}	1if arc (i,j) is used for one or more attacks, 0 otherwise
u_p	1if path p is removed, 0 otherwise
$\mathbf{x}, \mathbf{w}, \mathbf{y}, \tilde{\mathbf{y}}, \mathbf{z}, \tilde{\mathbf{z}}, \mathbf{u}$	Vector representations of the variables $x_{ij}, w_{ij}, y_{ij}, y_{ij}^t, z_t, z_i^k$, and u_p, respectively

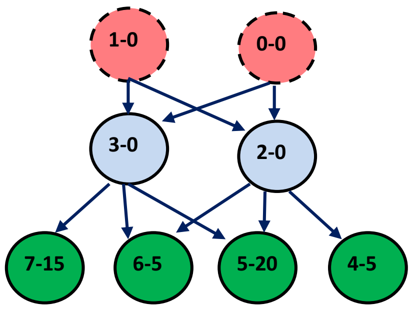


Figure 1: Example attack graph.

$$
\begin{aligned}
(\textsc{MinMaxBreach}) \quad & \min_{\mathbf{x}} \quad f(\mathbf{x}) && \text{(1a)}\\
& \text{s.t.} \quad \sum_{(i,j)\in\mathscr{A}} c^d_{ij} x_{ij} \le B_d && \text{(1b)}\\
& \qquad x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. && \text{(1c)}
\end{aligned}
$$

The function $f(\mathbf{x})$ in (1a) computes the maximum total reward achievable by the attacker for a given countermeasure vector $\mathbf{x}$. Constraint (1b) ensures that the total countermeasure cost does not exceed the budget.

The objective of the inner problem is to maximize the total reward possible by breaching the goal nodes given a specific solution $\hat{\mathbf{x}}$ of the outer problem. We refer to the following formulation of the inner problem as MAXBREACHBM. The BM at the end of MaxBreachBM indicates the presence of big M in this formulation.

$$
\begin{aligned}
(\textsc{MaxBreachBM}(\hat{\mathbf{x}}))\ f(\hat{\mathbf{x}}) = \quad & \max_{\mathbf{w},\mathbf{y},\mathbf{z}} \quad \sum_{t\in N_T} c^b_t z_t && \text{(2a)}\\
& \text{s.t.} \quad z_t - \sum_{(i,t)\in E(t)} y_{it} = 0 \quad \forall t \in N_T && \text{(2b)}\\
& \sum_{(i,j)\in L(i)} y_{ij} - \sum_{(j,i)\in E(i)} y_{ji} = 0 \quad \forall i \in N_R && \text{(2c)}\\
& y_{ij} \le M w_{ij} \quad \forall (i,j) \in A && \text{(2d)}\\
& w_{ij} \le 1 - \hat{x}_{ij} \quad \forall (i,j) \in A && \text{(2e)}\\
& \sum_{(i,j)\in A} c^a_{ij} w_{ij} \le B_a && \text{(2f)}\\
& z_t \le 1 \quad \forall t \in N_T && \text{(2g)}\\
& y_{ij} \ge 0 \quad \forall (i,j) \in A && \text{(2h)}\\
& w_{ij} \in \{0,1\} \quad \forall (i,j) \in A && \text{(2i)}
\end{aligned}
$$

The objective function (2a) calculates the total reward acquired by the attacker. Each constraint (2b) enforces that if at least one of the arcs going into a goal node is attacked, then the goal node is attacked. Each constraint (2c) ensures that an attack does not stop after visiting a transition node. Each constraint (2d) makes sure that if an arc is not used, no goal node is attacked through that arc.

Intuitively, the number of goal nodes attacked through an arc cannot be more than the number of all the goal nodes; therefore, we set $M = |N_T|$. Each constraint (2e) ensures that an arc is not used by the attacker if the arc is interdicted by the defender. Constraint (2f) ensures that the attacker does not spend more than the budget. Each constraint (2g) ensures that a goal node is not attacked more than once. Each pair of constraints (2d) and (2i) for an arc (i, j) ensures that even if more than one goal node is attacked through the arc, the attacker incurs the cost of attacking through (i, j) only once. Notice that the MAXBREACHBM formulation resembles a network design problem, especially because of the constraints (2d) accompanied by the constraints (2b) and (2c), which are are equivalent to the flow balance constraints of a network design problem. It is relatively straightforward to show that any solution of the MAXBREACHBM formulation produces binary values for the z_t variables and integer values for the y_{ij} variables, although we do not enforce any domain restrictions on these variables.

Notice that constraints (2b) in this form do not allow intermediate goal nodes. Constraints (2b) are reformulated as follows to allow intermediate goal nodes.

$$z_t = \sum_{(i,t)\in E(t)} y_{it} - \sum_{(t,i)\in L(t)} y_{ti} \quad \forall t \in N_T \tag{3}$$

Initial computational experiments using our algorithm (see section 4) on small problems showed that the majority of computation time solving the problem is spent solving the inner problem. Initial investigation into the inner problem solutions also shows that its LP relaxations are not tight because of the big M in constraints (2d) despite using $|N_T|$ as the value of M. Therefore, in an effort to remove the big M, we reformulate the inner problem, which we refer to as MAXBREACHD (the D at the end of MAXBREACHD indicates that this is a disaggregated version of the MAXBREACHBM formulation). In fact, MAXBREACHD resembles a multi-commodity network design problem where goal nodes are analogous to the commodities. To be concise, we commonly refer to the MAXBREACHBM and the MAXBREACHD formulations as the MAXBREACH formulation whenever that is appropriate. The solution of MAXBREACH produces an attack plan, $\bar{A}$, for the attacker, that is, $w_{ij} = 1\,\forall (i, j) \in \bar{A}$. MAXBREACHD is formulated as follows.

$$
\begin{aligned}
(\textsc{MaxBreachD}(\hat{\mathbf{x}}))\ f(\hat{\mathbf{x}}) = \quad & \max_{\mathbf{w},\tilde{\mathbf{y}},\mathbf{z}} \sum_{t \in N_T} c_t^b z_t && \text{(4a)}\\
\text{s.t.} \quad & z_t - \sum_{(i,t)\in E(t)} y_{it}^t = 0 \quad \forall t \in N_T && \text{(4b)}\\
& \sum_{(i,j)\in L(i)} y_{ij}^t - \sum_{(j,i)\in E(i)} y_{ji}^t = 0 \quad \forall i \in N_T, \forall t \in N_T \backslash \{i\} && \text{(4c)}\\
& \sum_{(i,j)\in L(i)} y_{ij}^t - \sum_{(j,i)\in E(i)} y_{ji}^t = 0 \quad \forall i \in N \backslash (N_I \cup N_T), \forall t \in N_T && \text{(4d)}\\
& y_{ij}^t \le w_{ij} \quad \forall (i,j) \in A, \forall t \in N_T && \text{(4e)}\\
& w_{ij} \le 1 - \hat{x}_{ij} \quad \forall (i,j) \in A && \text{(4f)}\\
& \sum_{(i,j)\in A} c_{ij}^a w_{ij} \le B_a && \text{(4g)}\\
& z_t \le 1 \quad \forall t \in N_T && \text{(4h)}\\
& y_{ij}^t \ge 0 \quad \forall (i,j) \in A, \forall t \in N_T && \text{(4i)}\\
& w_{ij} \in \{0,1\} \quad \forall (i,j) \in A && \text{(4j)}
\end{aligned}
$$

The objective function (4a) of MAXBREACHD is the same as the objective function (2a) of MAXBREACHBM. Also, the constraints (4h), (4f), (4g), and (4j) in MAXBREACHD are directly equivalent to the constraints (2g), (2e), (2f), and (2i), respectively, in MAXBREACHBM. Each constraint (4b) enforces that if attack is sent for goal node t using at least one of its incoming arcs, then goal node t is attacked. Each constraint (4c) ensures that any attack sent for a goal node t does not stop at any other goal node. Each constraint (4d) requires that an attack does not terminate at a transition node. Constraints (4e) have the same meaning as constraints (2d) except that the new constraints are enforced for attacks sent for each of the goal nodes. Similar to the MAXBREACHBM formulation, any solution of the MAXBREACHD formulation produces binary values for the z_t and the y_{ij}^t variables without any additional domain restrictions on this variables. Because one attack, at most, can be sent for a goal node, constraints (4e) do not require a big M, unlike constraints (2d) in MAXBREACHBM.

However, MAXBREACHD eliminates the big M at the expense of adding a large number of variables and constraints because the number of y_{ij}^t variables is much higher than the number of y_{ij} variables in MAXBREACHBM. The index t in y_{ij}^t means that attacks through an arc are now disaggregated into attacks aimed at each of the goal nodes. Also, MAXBREACHD has constraints for all t in constraints (4c) and (4d). Refer to sections 5 and 6 for a comparative analysis of the performances of MAXBREACHBM and MAXBREACHD formulations.

4. Solution Approach

The interdiction problem solved in this paper is computationally hard because the inner problem alone is NP-hard as Theorem 1 proves. Bi-level mixed-integer programs with binary variables only in the outer level can be converted to a single level by taking the dual of the inner level and merging the inner level with the outer level [44, 18]. Because our bi-level formulation has binary variables in both levels, the inner problem has a nonzero duality gap, requiring us to take a different approach. Thus, we develop a customized algorithm to solve our model. The framework of our algorithm is motivated by the algorithm in [10]. The algorithm requires two models to generate the upper and lower bounds. We refer to the model generating the upper bound as the sub-problem and the model generating the lower bound as the master-problem. The algorithm alternates between the master-problem and the sub-problem in an effort to reduce the optimality gap (i.e., the gap between the two bounds) at every iteration. The master-problem finds an optimal interdiction plan for a given set of alternative attack plans generated by the sub-problem through a specific iteration, whereas the sub-problem finds an optimal attack plan for a given interdiction plan generated by the master-problem.

The solution of the sub-problem (MAXBREACH) for a feasible interdiction plan produces an upper bound. However, there is no easy way to generate a good lower bound. Thus, to generate the lower bound, we adopt a model (MINATRISK) from a study by Nandi and Medal [27] on interdicting networks to prevent the spread of infections.

Theorem 1. *The attacker problem (*MAXBREACH*) is NP-hard.*

Proof. If there are only two levels in an attack graph, the initially vulnerable nodes in the upper level and the goal nodes in the lower level, the attack graph simplifies into a directed bipartite graph as shown in figure 2.

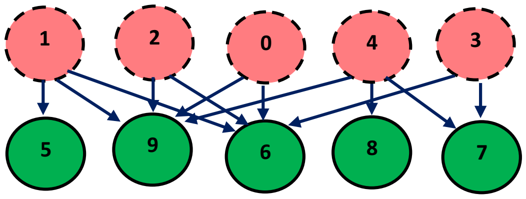


Figure 2: Bipartite attack graph

The attacker problem on a bipartite graph is a binary knapsack problem. Let, s_t be a binary variable representing the breach of a goal node t in an attack plan. If $s_t = 1$, goal node t is breached, and it costs the attacker an amount, $c_t^m = min_{(i,t)\in E(i)}(c_{it}^a)$. Then, the binary knapsack attacker problem can be formulated as follows: $\max \sum_{t\in N_T} c_t^b s_t : \sum_{t\in N_T} c_t^m s_t \leq B_a,\ s_t \in \{0,1\}\ \forall t \in N_T$. We know that the binary knapsack problem is NP-hard. Therefore, the attacker problem on a general attack graph is NP-hard. □

Lemma 1. *An attack is a tree in the attack graph.*

Proof. In a residual graph after interdiciton, every node can be breached through zero or more minimum cost paths. The attacker incurs additional cost by breaching any node through more than one path because of using additional arcs. However, the attacker does not gain any additional reward. Thus, in an attacker solution, any node will always be breached through at most one path which will result in a tree. Constraints (2c) and (2g) in the MAXBREACHBM formulation, and the constraints (4c), (4d), and (4h) in the MAXBREACHD formulation ensure that any node is breached at most once in an attacker solution. □

4.1. Upper Bound

The optimal objective value of MAXBREACH($\hat{\mathbf{x}}^k$) at iteration k, $f(\hat{\mathbf{x}}^k)$ is an upper bound for our algorithm at this iteration. MAXBREACH($\hat{\mathbf{x}}^k$) is the sub-problem for our algorithm. If UB is the minimum of $f(\hat{\mathbf{x}}^k)$ found through iteration k,

$$UB \quad \leq f(\hat{\mathbf{x}}^k) \tag{5}$$

4.2. Lower Bound

The MINATRISK model in [27] minimizes the number of susceptible nodes at risk of infection from infected nodes. The idea in the MINATRISK model is that for a specific interdiction plan of the defender, if the attacker is able to build a path from any of the vulnerability nodes to a goal node, that goal node is at risk of breach. We reformulate the MINATRISK model as follows in an effort to improve it before using the model for our purpose. In our reformulation, we also incorporate the fact that the attacker has a limited budget as opposed to no attacker budget restriction in the MINATRISK model. We refer to the new formulation as MINBREACHNODE($\bar{A}^k$) generated at iteration k of the algorithm. Here, $\bar{A}^k$ is the set of arcs associated with the attack plan generated by solving the MAXBREACH problem at iteration k of the algorithm. MINBREACHNODE($\bar{A}^k$) is the master-problem for our algorithm. Suppose, $L(\bar{A}^k)$ is the optimal objective value, and $\hat{\mathbf{x}}^k$ is the optimal solution of MINBREACHNODE($\bar{A}^k$). Then, $\hat{\mathbf{x}}^k$ is the new interdiction plan generated at iteration k of the algorithm. $L(\bar{A}^k)$ is a lower bound for our algorithm at iteration k because $\hat{\mathbf{x}}^k$ is generated considering only a subset of the alternative attack plans. If LB is the maximum of $L(\bar{A}^k)$ found through iteration k,

$$LB \geq L(\bar{A}^k) \tag{6}$$

$$\begin{aligned}
(\text{MINBREACHNode}(\bar{A}^k))\, L(\bar{A}^k) = \quad & \min_{\eta,\mathbf{x},\tilde{\mathbf{z}}} \quad \eta && \text{(7a)}\\
\text{s.t.} \quad & \eta \geq \sum_{t\in N_T^k} c_t^b z_t^k \qquad \forall k \in K && \text{(7b)}\\
& z_j^k \geq z_i^k - x_{ij} \quad \forall (i,j) \in \bar{A}^k, \forall k \in K && \text{(7c)}\\
& z_i^k = 1 \quad \forall i \in N_I^k, \forall k \in K && \text{(7d)}\\
& \sum_{(i,j)\in\cup_{k\in K}\bar{A}^k} c_{ij}^d x_{ij} \leq B_d && \text{(7e)}\\
& z_i^k \geq 0 \qquad \forall i \in N^k, \forall k \in K && \text{(7f)}\\
& x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \cup_{k\in K}\bar{A}^k && \text{(7g)}
\end{aligned}$$

The objective function (7a) and the constraints (7b) ensure that the objective of this model is the maximum total reward out of all the total rewards from all the attacks generated through iteration k. Each constraint (7c) ensures that if a node i is at risk of breach in attack (iteration) k, and there exists an arc (i,j), this arc must be interdicted for node j to not be at risk of breach through node i in this attack. All the incoming arcs of node j which have tail nodes that are at risk of breach must be interdicted for node j to be saved from breach. Each constraint (7d) requires a vulnerability node used in attack k to be already at risk of breach in this attack. Constraint (1b) is the defender's budget constraint. According to Lemma (1), an attack is a tree in the attack graph. A distinct set of nodes and arcs represent the associated attack tree. In order to differentiate one attack tree from another in the model, a new set of variables is generated for the nodes used in this attack tree, and new sets of constraints (7c) and (7d) are generated to represent the connectivity among these nodes. Note that a new variable z_j^k is generated for a node j if it is used in attack k. Any solution of the MINBREACHNODE formulation produces binary values for the z_i^k variables without enforcing the binary domain restrictions on these variables.

Each constraint (7b) performs a similar function as an optimality cut in Benders decomposition [5] in that it forces the master-problem to move closer to the optimal solution at each iteration. However, whereas each optimality cut in Benders decomposition cuts away a fraction of the feasible region of the master-problem, each constraint (7b) creates a new feasible region in a higher dimension by creating new variables (arcs). When a sufficient number of variables and constraints has been included, the solution of the master-problem is the optimal solution.

Theorem 2. *The master-problem (*MINBREACHNODE*) provides a valid lower bound.*

Proof. According to Lemma (1), an attack is a tree in the attack graph. The sets of constraints (7b), (7c), and (7d) in the MINBREACHNODE formulation above adds the attack tree from a specific attack to the attack graph used by the master problem. All the distinct attack trees are represented separately in this formulation, thereby implicitely taking the attacker budget into consideration. The MINBREACHNODE formulation adds k trees through iteration k. Because of adding only a subset of all the possible alternative attack trees through iteration k, the objective value of the MINBREACHNODE model provides a lower bound for the defender problem. □

4.3. *Algorithm* MINMAX

Input: Parameter values for MINMAXBREACH and tolerance $\varepsilon \geq 0$.
Output: Subset of arcs $\mathbf{x}^*$ on which to deploy countermeasures with a maximum optimality gap of ε.

1. Upper bound, $UB := \infty$, lower bound $LB := 0$, current interdiction plan, $\mathbf{x}^* := \hat{\mathbf{x}}^1 := 0$, iteration counter $k := 1$.
2. Given $\hat{\mathbf{x}}^k$, solve the sub-problem MAXBREACH($\hat{\mathbf{x}}^k$) to determine the attacker's optimal attack plan, $\hat{\mathbf{w}}^k$ and the associated $\bar{A}^k$.
3. If $f(\hat{\mathbf{x}}^k) < UB$, $UB := f(\hat{\mathbf{x}}^k)$. Set $\hat{\mathbf{x}}^K$ as the new best interdiction plan. $\mathbf{x}^* := \hat{\mathbf{x}}^k$.
4. If $UB - LB \leq \varepsilon$, go to END.
5. Generate new variables z_i^k for the nodes used in the current attack plan. Add a constraint (7b) and sets of constraints (7c), (7d), and (7f) corresponding to the sets of nodes and arcs used in this attack.
6. Solve MINBREACH($\bar{A}^{k+1}$). If $L(\bar{A}^{k+1}) > LB$, $LB := L(\bar{A}^{k+1})$ and $\mathbf{x}^* := \hat{\mathbf{x}}^{k+1}$. If $UB - LB \leq \varepsilon$, go to END.
7. $k = k + 1$, go to 2.
8. END: return $\mathbf{x}^*$ as the $\varepsilon-$optimal solution.

Lemma 2. *The master problem produces a new solution at each iteration until convergence.*

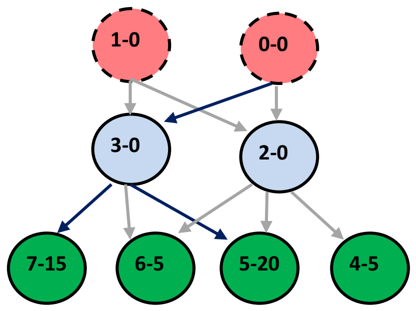
Proof. Let us assume that the master problem solution $\hat{\mathbf{x}}^k$ from iteration k is the same as the master problem solution $\hat{\mathbf{x}}^q$ from a prior iteration q. Then, the sub-problem solution $\hat{\mathbf{w}}^q$ from iteration q will repeat meaning that the attack plan represented by $\hat{\mathbf{w}}^q$ is not interdicted by the defender at iteration k. Then, the master problem objective function value $L(\bar{A}^k)$ from iteration k will be at least as large as the sub-problem objective function value $f(\hat{X}^q)$ from iteration q. Since, $LB \geq L(\bar{A}^k) \geq f(\hat{X}^q)$ and $UB \leq f(\hat{X}^q)$. Thus, $LB \geq UB$, and the algorithm terminates at this point. Hence, the master problem must produce a new solution at each iteration partially or completely interdicting each of the attack plans generated so far until convergence. □

Theorem 3. *The algorithm* MINMAX *converges within a finite number of iterations.*

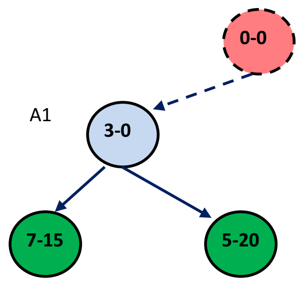
Proof. According to Lemma (2), the master problem will keep producing a new solution at each iteration until the last iteration. If there are $\mathbb{K}$ possible alternative interdiction plans, in the worst case, the algorithm will run through $\mathbb{K}$ iterations adding all the possible attack plans associated with the $\mathbb{K}$ interdiction plans. At this point, the algorithm must converge because of exhausting all the possibilities. Therefore, the MINMAX algorithm will terminate within a finite number ($\mathbb{K}$) of iterations. In reality, the algorithm usually terminates within a small fraction of the $\mathbb{K}$ maximum iterations. □

4.4. Algorithm Example

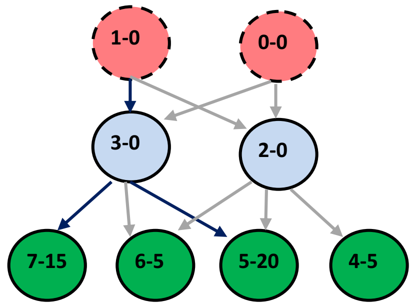
Figure 3 demonstrates how the algorithm works using the example attack graph in figure 1. The defender's budget is 1, and the attacker's budget is 3 in this demonstration. Each sub-figure in figure 3 represents a solution of either the master-problem or the sub-problem. See figure 1 for the original colors of the nodes and their meanings. The dark blue solid arcs in figure 3 are either used to attack (in the attacker's solutions) or not interdicted by the defender (in the defender's solution). The dashed arcs in the defender's solutions are interdicted by the defender. The gray arcs in the attacker's solutions are not used by the attacker. The input graph in each of the sub-figures related to the attacker's solutions is the whole graph except the arcs interdicted by the defender in this iteration. The input graphs in each of the sub-figures related to the defender's solutions are the whole graphs in that sub-figure. There is no sub-figure in figure 3 for defender's solution at iteration 1 because the algorithm begins with no interdiction.



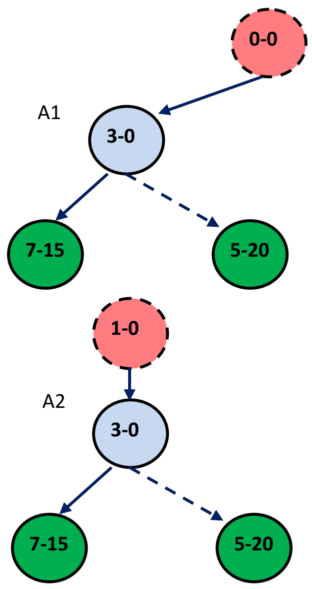
(a) Iteration 1: Attacker's solution. The attacker uses the original graph in figure 1 as the input and chooses this attack plan. Upper bound = 35.



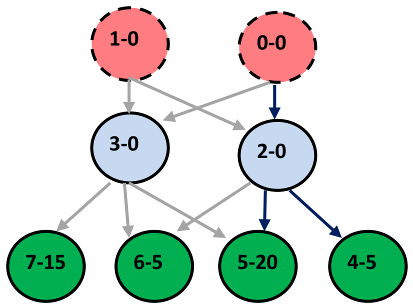
(b) Iteration 2: Defender's solution. The defender observes the attack plan in sub-figure 3a and interdicts arc (0,3). Lower bound = 0.



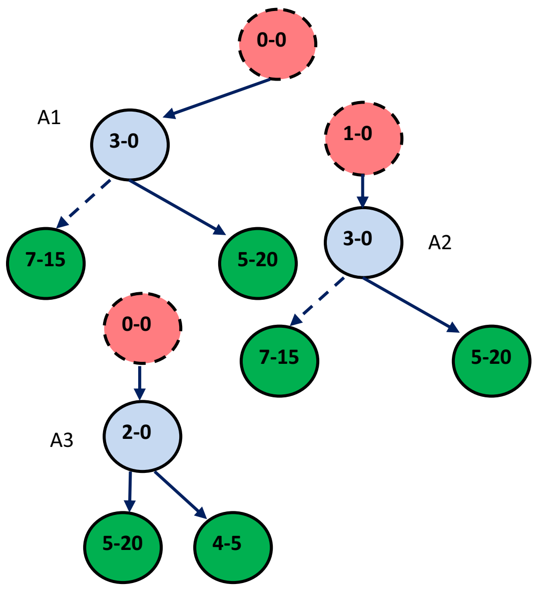
(c) Iteration 2: Attacker's solution. The attacker uses the original attack graph in figure 1 except the interdicted arc (0,3) as the input and produces this attack plan. Upper bound = 35.



(d) Iteration 3: Defender's solution. The defender takes into account all the attack plans in sub-figures 3a and 3c and interdicts arc (3,5). Lower bound = 15.



(e) Iteration 3: Attacker's solution. The attacker uses the original graph in figure 1 except the interdicted arc (3,5) as the input and produces this attack plan. Upper bound = 25.



(f) Iteration 4: Defender's solution. The defender takes into account all the attack plans in sub-figures 3a, 3c, 3e and interdicts arc (3,7). Lower bound = 25.

Figure 3: Algorithm in action.

At iteration 4, the algorithm terminates as the lower bound (25) and the upper bound (25) become equal. An optimal interdiction plan is to interdict arc (3,7), and a corresponding optimal attack plan is to attack using the set of arcs {(0,2),(2,5),(2,4)} comprised of the attack paths {(0,2),(2,5)} and {(0,2),(2,4)}.

Notice from figures 3b, 3d, and 3f that the number of sub-graphs input into the MinBreach model is growing at each iteration. Eventually, the MinBreach model finds an optimal interdiction plan by solving for a sufficiently large number of sub-graphs of the original graph.

4.5. *Enhancements to the* MINMAX *Algorithm*

Computational experiments show that the basic MINMAX algorithm takes too long to terminate for graphs having more than 100 nodes. The reason is that the computation time of the master problem increases exponentially because of adding a large number of variables and constraints at each iteration. Thus, we add the following enhancements.

4.5.1. *Path Based Formulation of the Master Problem (S)*

We exploit the fact that an attacker solution can be represented by a distinct set of paths instead of a distinct set of nodes and arcs as in the MinBreachNode formulation. Moreover, since an attack plan is a tree according to Lemma 1, there can be a maximum of $|N_T|$ paths in each attack plan with a maximum of one path to each goal node. At each iteration of the algorithm, we run a search algorithm on the attacker solution to find the set of paths used in this attack. If a path p found at an iteration is new, a variable u_p and associated constraints are generated and added to the master problem. If a path was used in a previous attack, we just attach the path to the current attack. Following is the path based formulation of the master problem.

$$
\begin{aligned}
(\textsc{MinBreachPath}(\bar{A}^k))\, L(\bar{A}^k) = \quad & \min_{\eta,\mathbf{u},\mathbf{x}} \quad \eta && \text{(8a)}\\
\text{s.t.} \quad & \eta \geq \sum_{p\in P_k} c^b_p(1-u_p) \qquad \forall k \in K && \text{(8b)}\\
& u_p \leq \sum_{(i,j)\in A_p} x_{ij} \quad \forall p \in \cup_{k\in K} P_k && \text{(8c)}\\
& \sum_{\forall(i,j)\in\cup_{k\in K}A^k} c^d_{ij} x_{ij} \leq B_d && \text{(8d)}\\
& u_p \leq 1 \quad \forall p \in \cup_{k\in K} P_k && \text{(8e)}\\
& u_p \geq 0 \quad \forall p \in \cup_{k\in K} P_k && \text{(8f)}\\
& x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \cup_{k\in K} A^k && \text{(8g)}
\end{aligned}
$$

Each constraint (8b) calculates the reward for attack k, and all constraints (8b) together ensure that the minimized reward objective is at least as large as the maximum of the rewards through iteration k. Each pair of the constraints (8c) and (8e) for a path, p together ensures that at least one of the arcs on a path must be removed to remove the path. Constraint (8d) is the defender budget constraint. Constraints (8f) and (8g) are the sign restriction and binary constraints, respectively. Any solution of the MINBREACHPATH formulation produces binary values for the u_p variables without the binary domain restriction on these variables. The MINBREACHPATH formulation usually has significantly fewer variables and constraints than the MINBREACHNODE formulation. Computational experiments demonstrate the superiority of the MINBREACHPATH formulation over the MinBreachNode formulation.

4.5.2. *Add Multiple Sub-Problem Solutions to the Master Problem (Ms)*

If only one sub-problem solution is added to the master problem solution, only a slightly different master problem is solved at each iteration requiring the algorithm to run through many iterations to provide the master problem enough sub-problem information for convergence. To overcome this issue along with the fact that Gurobi (the commercial solver used in this paper) is able to return multiple optimal and sub-optimal solutions from a solution, we add multiple sub-problem solutions to the master problem at each iteration. Adding multiple sub-problem solutions indeed reduces the number of iterations and the average computation time. Experiments show that adding 33% of the available solutions produces good results.

4.5.3. Statibilize Master Problem Solutions (TR)

One problem with the MINBREACH formulation is that it produces very divergent solutions at initial iterations of the algorithm slowing the convergence of the algorithm. In an effort to stabilize the master problem solution, we added a trust region cut at each of the 20 initial iterations of the algorithm. Suppose $\hat{\mathbf{x}}^k$ is the master problem solution from iteration k and $\hat{X}_1^k = \{(i,j) : \hat{x}_{ij}^k = 1\}$. Then, we add the following trust region cut to the master problem at the next iteration.

$$\sum_{(i,j)\notin \hat{X}_1^k} x_{ij} + \sum_{(i,j)\in \hat{X}_1^k} (1 - x_{ij}) \leq 0.33 \times 2 \times |\hat{X}_1^k| \tag{9}$$

The left hand side of constraint (9) calculates the Hamming distance [16] between the interdiction plan from iteration k and the interdiction plan from iteration $k+1$. The right hand side of constraint (9) ensures that a maximum of one-third of all the arcs interdicted at the current iteration is replaced at the next iteration. Master problem with the trust region cut does not provide a valid lower bound. Thus, we update the lower bound only after we stop adding the trust region cut. Experiments show that although the impact of adding trust region cut is not significant, the average computation time is slightly lower with the trust region cut in the master problem.

4.5.4. Apply Heuristic to Solve the Master Problem (Hf)

Our heuristic to solve the master problem is based on the greedy addition heuristic proposed by Toyoda [40]. Our heuristic selects a set of arcs to be interdicted using the following steps.

1. Initialize the set of arcs to be interdicted, $Xh = \emptyset$, $tBudget = 0$.
2. Evaluate a metric, $Score_{ij} = \frac{Saved_{ij}}{SecurityCost_{ij}}$ for each of the arcs not selected yet and the interdiction of the arc does not exceed the budget. Here, *Saved* is the difference between current maximum total reward gained by the attacker and the maximum reward gained if the arc is interdicted. And, *SecurityCost* is the arc security cost.
3. Select the arc (i,j) with the maximum $Score_{ij}$ calculated in step 2 and add to Xh. $tBudget = tBudget + SecurityCost_{ij}$.
4. If $tBudget < Bd$, go to step 2, else return Xh.
   The above heuristic (Hf) can be used to approximately solve the master problem at each iteration of the MINMAX algorithm. The solution Xh generated by the above heuristic can be used directly as the master problem solution, or it can be used as a warm start for the master problem solver. Xh is not a good solution for the master problem in most of the cases, and thus, we do not use Xh directly as a master problem solution in the MINMAX algorithm.

5. Computational Experiments

In this section, we perform experiments to ascertain the effects of the model parameters and different topological attributes of attack graphs on the computation times and the loss due to breach.

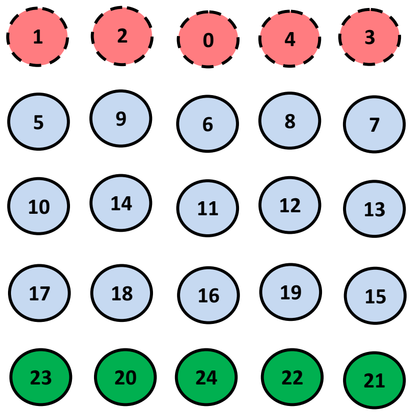
All the experiments are performed on synthetic attack graphs generated using the approach described in the following paragraph. All the experiments are carried out on a laptop with an Intel core i7 2.70 GHz processor and 16 GB RAM. The algorithms are implemented in Python 3.4 with Gurobi [15] as the solver for both the MAXBREACH and the MINBREACH problems.

In most of the attack graphs in the literature, nodes are organized in a hierarchical topology with nodes in one level having directed arcs incident to nodes in the subsequent level [22]. A level usually means the level of access acquired into the system. Thus, if an attack path has five levels, an attacker will have to acquire five successive levels of access, possibly by using five different exploits, in order to reach a goal node. This is very similar to attack trees except that any node might have more than one incoming arc, making it a graph. With that in mind, we generate the graphs for this paper randomly using the following method. This method uses the number of nodes, number of levels, probability of an arc between any pair of nodes in subsequent levels (*pd*), and probability of an arc between any pair of nodes in the same level (*ps*) as inputs, and generates the graph topology as the output.

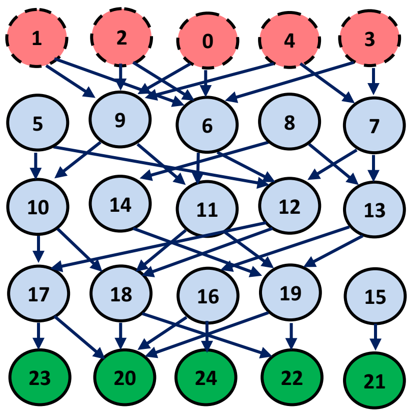
1. The nodes are arbitrarily divided into r levels. All the nodes in the first level are designated as vulnerability nodes, and all the nodes in the last level are designated as the goal nodes. Thus, all the nodes in the intermediate levels are transition nodes. In this way, the minimum distance between the vulnerability nodes and the goal nodes equals $r-1$ levels.
2. Directed arcs are generated randomly from each level to its subsequent level with probability *pd* for any specific arc.
3. Arcs are also generated randomly between any pair of nodes within a level with probability *ps* for any specific arc.
4. One incoming arc is generated randomly to each of the transition and goal nodes without any incoming arc through step 3. Tail nodes of the arcs generated in step 4 are selected randomly from the corresponding prior levels. Step 4 makes sure that all the transition and goal nodes have at least one incoming arc.

In the graphs generated using the four steps above, any of the vulnerability nodes will not have any incoming arcs, and some of the goal nodes might have outgoing arcs because of step 3. Probabilities *pd* and *ps* are tuned manually to generate the desired number of arcs. Figure 4 shows a five-level graph generated using our approach.

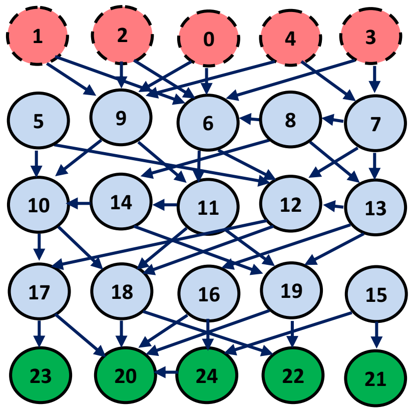
Table 2 shows the different parameters and their values used in the experiments. We perform experiments on 4 different network sizes (defined by the number of nodes) to show how the computation time is impacted by the size of the graph. The number of arcs in each of the graphs is approximately 2.15 times the number of nodes. Three and two different level values are used for graphs with 50 nodes and 100 nodes, respectively to examine the variation of computation time with respect to the number of levels. Loss due to the breach of goal nodes, the costs of attack on arcs, and the costs of countermeasures on arcs are generated using uniform distribution with different parameter values. Table 3, 4, and 5 present the results from the experiments.



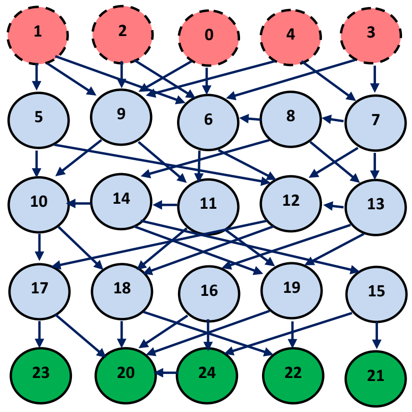
(a) Step 1: Nodes are divided into 5 levels with the vulnerability nodes in the first level, the goal nodes in the fifth level, and the transition nodes in the other levels.



(b) Step 2: Arcs from each level to its subsequent level are generated.



(c) Step 3: Arcs with both tail and head nodes in the same level are generated.



(d) Step 4: Incoming arcs are generated for transition nodes 5 and 15 without incoming arcs through step 3.

Figure 4: Attack graph generated using our approach. Node colors have the same meaning as in figure 1.

Parameters	Values
Network size (nodes)	50, 100, 150, 200
Network size and number of levels combinations (size, levels)	(50,5), (50,7), (50,10), (100,2), (100,5), (150,5), (200,5)
Arcs	$\approx 2.15 \times$Nodes
Network sizes and defender budget combinations (size, low budget, <intermediate budget>, <high budget>)	(50, 75), (100, 100, 150, 250), (150, 275), (200, 375)
Network sizes and attacker budget combinations (size, low budget, <high budget>)	(50, 125), (100, 150, 300), (150, 325), (200, 425)
Loss due to breach of the goal nodes	~Uniform(500, 1500), ~Uniform(1000, 2000)
Cost of attacks on arcs	~Uniform(10, 30), ~Uniform(30,50)
Cost of countermeasures on arcs	~Uniform(10, 30), ~Uniform(30, 50)

Table 2: Parameters and their values used in the experiments.

Table 3: Growth of computation time with graph size. Four random instances of graphs are used for each graph size. Levels = 5 and other parameters at their low levels. Values in italic mean that those graphs cannot be solved in 2 hours. Iters, Ttime, and masTime as the column labels are short for number of iterations, total runtime, and master problem runtime, respectively.

Graph Params		MNLHf			EM		
Nodes	Arcs	Iters	Ttime	masTime	Iters	Ttime	masTime
50	118	22	4.8	0.4	22	6.7	0.6
50	126	21	10.1	0.4	25	12.6	1.1
50	119	21	20.6	0.3	21	19.1	0.3
50	116	21	7.3	0.4	22	7	0.5
100	237	22	62.8	0.9	23	80.1	2.8
100	235	20	73.5	0.8	20	87.6	1.8
100	230	20	50.6	0.5	23	58.4	1
100	248	25	55.2	1.3	37	114.3	27.3
150	374	167	1180.2	195.7	*39*	*7200*	*7056*
150	374	123	772.9	94.8	*35*	*7200*	*7012*
150	345	97	4557.6	51.7	*29*	*7200*	*7009*
150	352	109	918.5	81.3	*33*	*7200*	*6953*
200	431	230	2937.2	487.5	*21*	*7200*	*7046*
200	437	445	5190.9	1256.7	*23*	*7200*	*7062*
200	425	208	1661	313.5	*22*	*7200*	*7023*
200	424	131	992.2	181.6	*24*	*7200*	*6984*

Table 4: Quality of solution and computation time (clock seconds) of the heuristic methods (MNLHf and MNL) compared to the exact method (EM). Master model NodeLimit = 1, Gurobi initial heuristic NodeLimit (subMIPNodes) = 2000. . BreachLoss, masTime, and %Higher are, respectively, short for breach loss, master problem runtime, and the percentage by which the breach loss from that computational method is higher than the breach loss from EM.

Graph Params		EM		MNLHf			MNL		
Nodes	Levels	BreachLoss	masTime	BreachLoss	masTime	%Higher	BreachLoss	masTime	%Higher
50	5	3990	0.6	4163	0.4	4.34	4436	0.3	11.18
50	5	4893	1.1	5062	0.4	3.45	5062	0.3	3.45
50	5	2999	0.3	2999	0.3	0	2999	0.3	0
50	5	3564	0.5	3564	0.4	0	3564	0.4	0
50	7	2273	0.4	2273	0.2	0	2306	0.3	1.45
50	7	1995	0.3	1995	0.2	0	1995	0.3	0
50	7	1226	0	1226	0.1	0	1226	0.1	0
50	7	2343	0.5	2343	0.3	0	2517	0.3	7.43
100	5	5122	2.8	5122	0.9	0	5122	0.7	0
100	5	5097	1.8	5191	0.8	1.84	5097	0.6	0
100	5	4749	1	4749	0.5	0	4749	0.6	0
100	5	5839	27.3	5839	1.3	0	5839	0.9	0

6. Discussion

Figure 5 compares the average computation times from the different exact computational methods. In this figure, All, N, S, SHf, SMs, and STR have the following meanings: All - path based (MINBREACHPATH) method with all the enhancements, N - node based (MINBREACHNODE) method, S - path based method without any other enhancements, SHf - path based method with master problem heuristic (Hf), SMs - path-based method with the addition of multiple attacker solutions at an iteration (Ms), and STR - path based method with the addition of trust region cut to the master problem (TR). This figure clearly demonstrates that the path based method is much superior to the node based method. Average computation time seems to increase slightly if Hf is applied with the path based method. However, table 4 shows that the heuristic method with Hf usually finds more optimal solutions and higher quality solutions (if not optimal) compared to the heuristic method without the master problem heuristic. Thus, application of Hf can be very useful when the heuristic method is used. Addition of multiple sub-problem solutions seems to slightly decrease the average computation time. Addition of trust region cuts also seems to slightly decrease the average computation time. Application of all the enhancements on the path based method decreases the average computation time compared to the application of only one of the enhancements on the path based method. Note however that the computational methods and the enhancements only impact the computation time of the master problem.

Based on the findings above, in all the subsequent experiments, we use the following three variations of the MINMAX algorithm to solve the MINMAXBREACH problem based on whether the master problem is solved to optimality, and whether Hf is used to provide a warm start. We refer to the MINMAX algorithm as the exact method (EM) when Hf is not used, and the master problem is solved to optimality. Following two adaptations of the MINMAX algorithm are the heuristic methods. (1) We use Hf to provide a warm start to the master problem and limit the

Table 5: Computational performance of the MaxBreachBM and the MaxBreachD formulations of the sub-problem. The last two columns contain average computation times of the sub-problem per iteration averaged over a maximum of 10 iterations.

Graph Params			Defender Budget	Attacker Budget	Average sub-problem runtime (clock seconds)	
Nodes	Arcs	Levels			MaxBreachBM	MaxBreachD
50	118	5	75	150	0.31	0.26
50	126	5	75	150	0.67	0.45
50	119	5	75	150	0.71	0.27
50	116	5	75	150	0.22	0.18
50	117	7	75	150	1.31	0.25
50	116	7	75	150	0.67	0.21
50	112	7	75	150	1.18	0.95
50	109	7	75	150	0.38	0.22
50	117	10	75	150	11.7	3.63
50	124	10	75	150	4.04	2.42
50	126	10	75	150	5.5	2
50	123	10	75	150	51.4	5.5
100	237	5	250	375	5	0.65
100	235	5	250	375	12.1	1.09
100	230	5	250	375	2.65	1.14
100	248	5	250	375	4.88	1.63
100	220	2	250	375	0.15	0.38
100	225	2	250	375	0.17	0.39
100	236	2	250	375	0.22	0.38
100	224	2	250	375	0.17	0.5
150	374	5	275	325	3.6	1.56
150	374	5	275	325	5.3	1.73
150	345	5	275	325	5.25	2.63
150	352	5	275	325	6	2.5

number of nodes in the master problem branch and bound tree explored by the optimization solver to 1 [15] . We refer to this heuristic method as the node limit heuristic with heuristic-based master problem warm start (MNLHf). (2) We do not use Hf but limit the number of nodes explored by the optimization solver to 1. We refer to the second heuristic method as the node limit heuristic (MNL). Neither of the heuristic methods is guaranteed to produce a valid lower bound. Nevertheless, we use the lower bound from these heuristic methods to terminate the algorithm accepting the fact that the final solution might not be optimal. Other enhancements S, MS, and TR are used in all the three variations of the algorithm. Node based master problem formulation is used only in the experiments demonstrating the superiority of the path based formulation (figure 5).

From the *Time* columns in table 3, we see that total computation time increases sharply with respect to graph size, especially when the exact method is used. Graphs with 150 and 200 nodes cannot be solved in two hours using the exact method. However, the heuristic method solves most

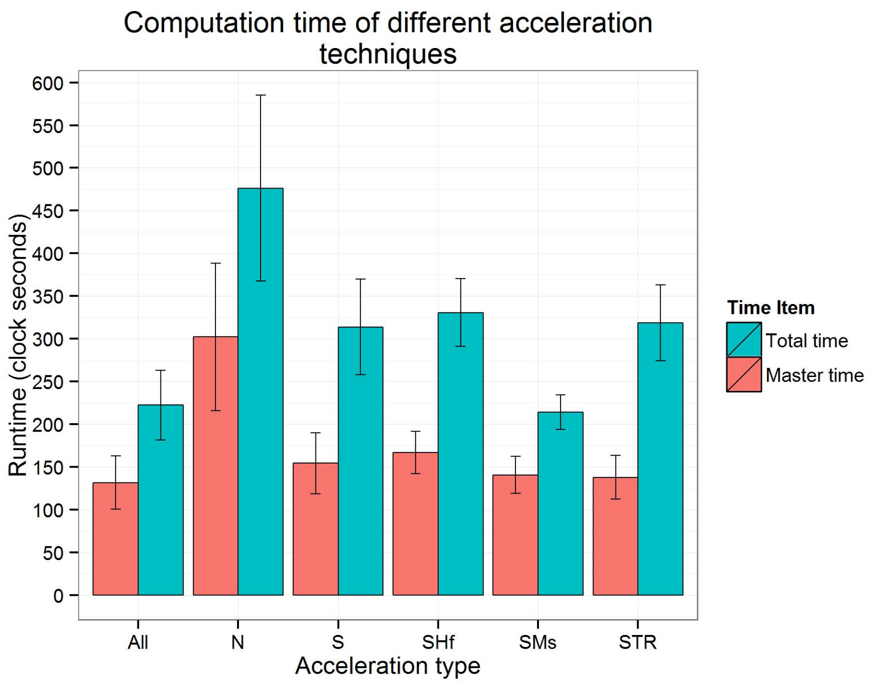


Figure 5: Average computation times on 4 attack graphs for each of the base or accelerated computation techniques. Nodes = 100, defender budget = 150, and attacker budget =150. The vertical line at the top of each histogram bar equals twice of the standard deviation of the corresponding time item.

of the 150-node and 200-node graphs in less than 15 minutes.

In fact, experimentation on 100-node graphs shows that the heuristic method is able to solve the problem very quickly for most of the parameter combinations. The median computation time of all the parameter combinations is less than 2 minutes. Therefore, the heuristic method has the potential to solve relatively large problems within a reasonable amount of time, especially for suitable combinations of parameters and topology of the graph.

One important observation is that the time taken by the algorithm to solve the sub-problem is much greater than the time taken to solve the master-problem during the intitial iterations of the algorithm. However, as the number of iterations increases, the master problem becomes bigger sharply increasing its computation time. We see from table 3 that when the exact method is used, the majority of the total time is spent solving the master problem. In contrast, when the heuristic method is used, the majority of the total time is spent solving the sub-problem. Comparision of the masTime columns under the exact and the heuristic methods makes it clear that the heuristic method significantly decreases the computation time of the master problem.

From table 4, we see that 9 and 8 out of the 12 solutions using the heuristic method with Hf and the heuristic method without Hf, respectively are optimal. So, a large fraction of the solutions from both of the heuristic methods are optimal. However, the heuristic method with Hf seems to find optimal solutions more frequently and solutions with higher quality if it does not find the optimal solution.

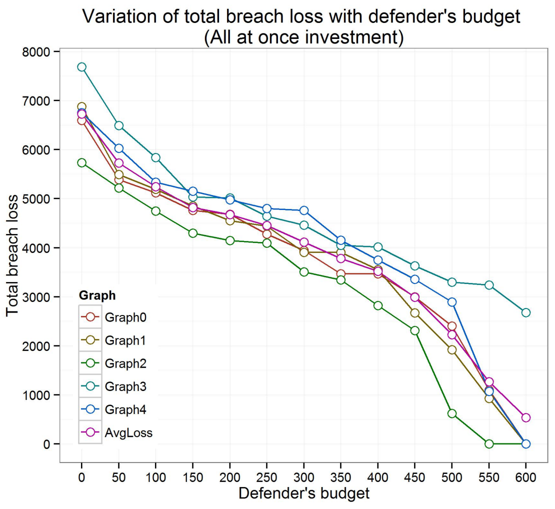
Comparing the computation times of the graphs with the same number of nodes and levels, it is clear that significant variation remains. Although the number of nodes and the levels are the same, their topologies are different because the graphs are generated randomly. In fact, topology plays an important role not just in computation times but also in the total breach loss.

We see from table 5 that the average computation time of the MaxBreachBM formulation increases with the number of levels. Upon examination of the computation times of the 50-node graphs, it is apparent that the computation times of the 50-node graphs with 10 levels are much higher on average than the computation times of the 50-node graphs with 5 levels. Because of the higher number of levels, the average distance between the vulnerability nodes and the goal nodes is much longer in the 10-level graphs than in the 5-level graphs. Thus, the 10-level graphs have more paths than the 5-level graphs, and the paths are longer, making it relatively more difficult for the attacker to decide whether to attack a specific node. When the graph size is significantly large relative to the number of levels, computation time of the MaxBreachBM formulation decreases significantly with graph size. Computation times of the 100-node graphs with 2 levels are significantly smaller than the computation times of the 50-node graphs with 10 levels. Because of the small number of levels, there are relatively more short paths and possibly fewer paths from vulnerability nodes to the goal nodes in the 100-node graphs with 2 levels, making it easier for the attacker to decide whether to attack some goal nodes. However, increasing the number of levels while keeping the number of nodes unchanged does not monotonically increase the computation time because that makes the graph easier for the defender to defend given the same defense budget.

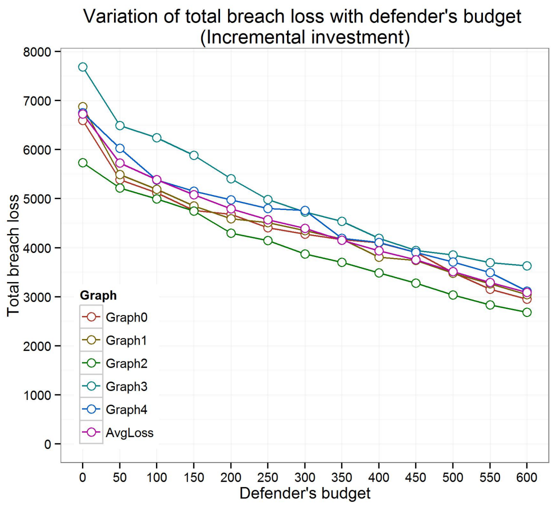
Average computation time of the MAXBREACHD formulation is shorter than the average computation time of the MAXBREACHBM formulation under many network size and number of level combinations, especially when the number of levels is large relative to the graph size. However, variation of computation times is also much more for the MaxBreachD formulation. This is probably an indication that the MAXBREACHD model is more affected by the topology of the attack graph than the MAXBREACHBM model. Computation times of the MAXBREACHD formulation are smaller than the computation times of the MAXBREACHBM formulation for the 100-node graphs with 2 levels in table 5. Recall that for the MAXBREACHD formulation, the number of constraints and variables depends on the number of goal nodes, and as the number of goal nodes increases, the number of constraints and variables increases. Because of the smaller number of levels, the 100-node graph with 2 levels have 50 goal nodes instead of 20 goal nodes as in the 100-node graphs with 5 levels. Therefore, we can conclude that the MAXBREACHD formulation usually performs worse than the MAXBREACHBM formulation in cases when the number of goal nodes is relatively high, or, in other words, the number levels is smaller for the same graph size.

Although there are some random variation in the plots of figure 6a, the common pattern of relationships between the total breach loss and the amount of the defender's budget is that the breach loss drops sharply with increase in budget at the beginning, and it then forms a concave shape until the breach loss becomes zero. This relationship becomes even clearer from the plot of the average breach losses. This relationship between the breach loss and the defender's budget implies that the defender will need to spend much more to reduce per unit of breach loss after the initial set of relatively easy security challenges are successfully confronted. Decision makers should investigate breach losses for a wide range of defense budgets before allocating a specific budget to ensure high return on investment.

The shapes of the plots in figure 6b are quite different than the shapes in figure 6a. In figure 6b, to find the deployment for a specific budget, 100 units for example, the deployment is first optimized for 50 units, and then again the additional deployment is optimized for the additional 50 units keeping the initial deployment unchanged. In contrast, the deployment is optimized for the



(a) All at once investment. The whole budget is spent at once meaning that, for any specific budget, countermeasure deployment is optimized for that whole budget.



(b) Incremental investment. The whole budget is spent in increments of 50 meaning that countermeasure deployment is optimized incrementally for each additional 50 units of budget on top of the existing deployment.

Figure 6: Variation of breach loss with defender's budget. This chart is generated using five randomly generated graphs with 100 nodes. Graph0, Graph1, Graph2, Graph3, and Graph4 have 228, 232, 228, 221, and 238 arcs, respectively. All the parameters except the defender's budget are at their low levels. Average breach loss for a specific budget is determined by averaging the breach losses over all the graphs.

whole budget of 100 units in figure 6a. It is clear from these figures that if additional investment is to be made, countermeasure deployment should be re-optimized for the whole budget instead of optimizing only for the additional budget to achieve the maximum benefit possible from a specific budget.

Finally, it might not be easy to know or estimate the budget of the attacker correctly. Suppose the attacker's actual budget is an unknown amount between 100 and 200, but the defender's estimate of the attacker's budget is 150. Assume also that the defender's budget is 250, the other parameters are at their low levels, and the five graphs are the same as those used in figure 6. Values in table 6 are the extra breach losses due to error in the estimates of the attacker's budget. The second column corresponds to the extra breach losses due to a 50% overestimate of the attacker's

Table 6: Sensitivity of total breach loss with respect to error in the estimated attacker's budget.

Graphs	Extra Loss (%) for 50% Overestimate	Extra Loss (%) for 25% Underestimate
Graph0	11.34	10.14
Graph1	27.62	4.93
Graph2	33.54	3.82
Graph3	13.1	1.37
Graph4	21.03	14.5

budget (the actual attacker budget is 100, and the defender's perception of the attacker's budget is 150), and the third column corresponds to the extra breach losses due to a 25% underestimate of the attacker's budget (the actual attacker budget is 200, and the defender's perception of the attacker's budget is 150). For example, with respect to Graph0, the defender incurs an extra breach loss of 11.34% due to a 50% overestimate, and an extra breach loss of 10.14% due to a 25% underestimate of the attacker's budget. The average extra loss is 14.17% corresponding to an average error of 37.5%. Therefore, the quality of an interdiction plan is relatively insensitive on average with respect to the error in the defender's knowledge about the attacker's budget. Moreover, the extra losses due to the underestimate are significantly smaller than the extra losses due to the overestimate of the attacker's budget. Hence, it is preferable to have an underestimate rather than an overestimate of the attacker's budget to reduce the extra losses due to estimation error.

7. Conclusion

This paper presents the breach of information resources of an organization by an attacker and the defensive measures of the organization as a defender-attacker bi-level network interdiction model. The inner level represents the attacker trying to maximize the total reward, and the outer level represents the defender trying to minimize the maximum total reward achievable by the attacker. Both the inner- and the outer levels are formulated as mixed-integer linear programs. We provide two alternate formulations for the inner problem. As both of the levels have binary variables, we also develop a customized algorithm to solve the model. The path based method with the enhancements is much faster than the node based method. A heuristic method with or without the application of master problem heuristic providing warm start is capable of solving relatively large problems for various parameter settings. The following are some of the most important findings. 1) The computation time of the two sub-problem formulations does not monotonically increase with graph size. In fact, average computation time of the MaxBreachBM formulation decreases with graph size when the graph size is significantly large relative to the number of levels. 2) The majority of computation time is spent solving the sub-problem when the heuristic method is used. 3) The MAXBREACHD formulation without big M is a computationally better formulation than the MAXBREACHBM formulation with the big M when the number of levels is relatively high for a specific graph size. 4) When additional investment is made, countermeasure deployment should be re-optimized for the whole budget instead of optimizing for only the additional budget. 5) Breach loss drops sharply with increase in the defense budget at the beginning, then levels off before finally dropping sharply again with increasing defense budget. And 6) Quality of an interdiction plan is relatively insensitive with respect to the error in the estimate of the attacker's budget.

In this work, we focus on accelerating the master problem solution. Further research should emphasize on speeding up the sub-problem solution. Decomposition techniques such as Benders decomposition and Lagrangian relaxation aided by additional inequalities should be investigated. This work can be extended to have actual security countermeasures with multiple levels of defense of arcs (interdiction) rather than binary defense. It will be valuable to relax the implicit assumptions that the attacker has complete information about the topology of the attack graph, and that the countermeasures are capable of protecting the arcs completely. It will also be more realistic to consider the interdiction of arcs as stochastic; that is, the interdiction effects are probabilistic. The models developed in this paper are risk neutral. It will be interesting to analyze the problem with the objective of minimizing the probability of large losses. Finally, it will be worthwhile to perform experiments on real attack graphs rather than the synthetic ones used in this paper.

References

[1] Albanese, M., Jajodia, S., Noel, S., Jun. 2012. Time-efficient and cost-effective network hardening using attack graphs. In: Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on. IEEE, pp. 1–12.

[2] Alderson, D. L., Brown, G. G., Carlyle, W. M., 2014. Assessing and Improving Operational Resilience of Critical Infrastructures and Other Systems. Tutorials in Operations Research, 180–215.

[3] Alhomidi, M., Reed, M., 2013. Finding the minimum cut set in attack graphs using genetic algorithms. In: Computer Applications Technology (ICCAT), 2013 International Conference on. IEEE, pp. 1–6.

[4] Ammann, P., Wijesekera, D., Kaushik, S., 2002. Scalable, Graph-based Network Vulnerability Analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. CCS '02. ACM, New York, NY, USA, pp. 217–224.

[5] Benders, J. F., Dec. 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.

[6] Berry, J., Carr, R. D., Hart, W. E., Leung, V. J., Phillips, C. A., Watson, J.-P., http://ascelibrary, R. M., 2009. Designing Contamination Warning Systems for Municipal Water Networks Using Imperfect Sensors. Journal of Water Resources Planning and Management 135 (4), 253–263.

[7] Berry, J., Hart, W. E., Phillips, C. A., Uber, J. G., Watson, J.-P., 2006. Sensor Placement in Municipal Water Networks with Temporal Integer Programming Models. Journal of water resources planning and management 132 (4), 218–224.

[8] Berry, J. W., Boman, E., Riesen, L. A., Hart, W. E., Phillips, C. A., Watson, J. P., Murray, R., 2010. User's manual: TEVA-SPOT toolkit version 2.4. Tech. rep., Technical Report EPA/600/R-08/041, National Homeland Security Research Center, Office of Research and Development, US Environmental Protection Agency.

[9] Bistarelli, S., Fioravanti, F., Peretti, P., Apr. 2006. Defense trees for economic evaluation of security investments. In: Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on. IEEE, Washington, DC, USA, pp. 8 pp.–423.

[10] Brown, G. G., Carlyle, W. M., Harney, R. C., Skroch, E. M., Wood, R. K., 2009. Interdicting a Nuclear-Weapons Project. Operations Research 57 (4), 866–877.

[11] Chen, F., Tu, R., Zhang, Y., Su, J., Apr. 2009. Two Scalable Analyses of Compact Attack Graphs for Defending Network Security. In: Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on. Vol. 1. IEEE, pp. 627–632.

[12] Dewri, R., Poolsappasit, N., Ray, I., Whitley, D., 2007. Optimal Security Hardening Using Multi-objective Optimization on Attack Tree Models of Networks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. CCS '07. ACM, New York, NY, USA, pp. 204–213.

[13] Dewri, R., Ray, I., Poolsappasit, N., Whitley, D., Mar. 2012. Optimal security hardening on

attack tree models of networks: a cost-benefit analysis. International Journal of Information Security 11 (3), 167–188.

[14] FBI, 2013. 2013 Internet Crime Report. Tech. rep., FBI.

[15] Gurobi, O., 2015. Gurobi optimizer reference manual. URL: http://www. gurobi. com.

[16] Hamming, R. W., Apr. 1950. Error Detecting and Error Correcting Codes. Bell System Technical Journal 29 (2), 147–160.

[17] Ingols, K., Lippmann, R., Piwowarski, K., Dec. 2006. Practical Attack Graph Generation for Network Defense. In: Computer Security Applications Conference, 2006. ACSAC 06. 22nd Annual. IEEE, pp. 121–130.

[18] Israeli, E., Wood, R. K., Sep. 2002. Shortest-path network interdiction. Networks 40 (2), 97–111.

[19] Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J., Nov. 2011. Cauldron mission-centric cyber situational awareness with defense in depth. In: Military Communications Conference, 2011 - MILCOM 2011. IEEE, pp. 1339–1344.

[20] Jajodia, S., Noel, S., O'Berry, B., 2005. Topological Analysis of Network Attack Vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (Eds.), Managing Cyber Threats. Vol. 5 of Massive Computing. Springer US, pp. 247–266.

[21] Kordy, B., Piètre-Cambacédès, L., Schweitzer, P., 2014. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. Computer science review 13, 1–38.

[22] Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R., Oct. 2006. Validating and Restoring Defense in Depth Using Attack Graphs. In: Military Communications Conference, 2006. MILCOM 2006. IEEE. IEEE, pp. 1–10.

[23] Lippmann, R. P., Ingols, K. W., 2005. An Annotated Review of Past Papers on Attack Graphs. Tech. Rep. PR-IA-1, Massachusetts Inst of Tech Lexington Lincoln Lab.

[24] Morton, D. P., Pan, F., Saeger, K. J., Jan. 2007. Models for nuclear smuggling interdiction. IIE Transactions 39 (1), 3–14.

[25] Murray, R., 2009. US Environmental Protection Agency uses operations research to reduce contamination risks in drinking water. Interfaces 39 (1), 57–68.

[26] Murray, R., Haxton, T., Janke, R., Hart, W. E., Berry, J., Phillips, C., 2010. Sensor network design for drinking water contamination warning systems: A compendium of research results and case studies using the TEVA-SPOT software. US Environmental Protection Agency, Washington, DC, EPA/600/R-09 141.

[27] Nandi, A. K., Medal, H. R., 2016. Methods for removing links in a network to minimize the spread of infections. Computers & Operations Research 69, 10–24.

[28] Nehme, M. V., 2009. Two-person games for stochastic network interdiction: models, methods, and complexities. Ph.D. thesis, The University of Texas at Austin.

[29] Noel, S., Jajodia, S., Sep. 2008. Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs. J. Netw. Syst. Manage. 16 (3), 259–275.

[30] Noel, S., Jajodia, S., O'Berry, B., Jacobs, M., Dec. 2003. Efficient minimum-cost network hardening via exploit dependency graphs. In: Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, pp. 86–95.

[31] Pan, F., Morton, D. P., Oct. 2008. Minimizing a stochastic maximum-reliability path. Networks 52 (3), 111–119.

[32] Phillips, C., Swiler, L. P., 1998. A Graph-based System for Network-vulnerability Analysis. In: Proceedings of the 1998 Workshop on New Security Paradigms. NSPW '98. ACM, New

York, NY, USA, pp. 71–79, phillipsSwiler1998.

[33] Reed, B. K., 1994. Models for Proliferation Interdiction Response Analysis. Master's thesis, Naval Postgraduate School Monterey CA.

[34] Riley, M., Elgin, B., Lawrence, D., Matlack, C., Mar. 2014. Missed Alarms and 40 Million Stolen Credit Card Numbers: How Target Blew It. Bloomber Businessweek.

[35] Roy, A., Kim, D. S., Trivedi, K. S., 2010. Cyber Security Analysis Using Attack Countermeasure Trees. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. CSIIRW '10. ACM, New York, NY, USA.

[36] Salmeron, J., Wood, K., Baldick, R., May 2004. Analysis of Electric Grid Security Under Terrorist Threat. IEEE Transactions on Power Systems 19 (2).

[37] Salmeron, J., Wood, K., Baldick, R., Feb. 2009. Worst-Case Interdiction Analysis of Large-Scale Electric Power Grids. Power Systems, IEEE Transactions on 24 (1), 96–104.

[38] Sullivan, K. M., Morton, D. P., Pan, F., Cole Smith, J., Mar. 2014. Securing a border under asymmetric information. Naval Research Logistics 61 (2), 91–100.

[39] Swiler, L. P., Phillips, C., Ellis, D., Chakerian, S., 2001. Computer-attack graph generation tool. In: DARPA Information Survivability Conference and Exposition II, 2001. DISCEX 01. Proceedings. Vol. 2. IEEE, pp. 307–321 vol.2.

[40] Toyoda, Y., 1975. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. Management Science 21 (12), 1417–1427.

[41] Watson, J.-P., Murray, R., Hart, W. E., 2009. Formulation and optimization of robust sensor placement problems for drinking water contamination warning systems. Journal of Infrastructure Systems 15 (4), 330–339.

[42] Weise, E., Oct. 2014. JP Morgan reveals data breach affected 76 million households. USA TODAY.

[43] Weiss, J. D., 1991. A system security engineering process. Vol. 249. Proceedings of the 14th National Computer Security Conference.

[44] Wood, R. K., 1993. Deterministic network interdiction. Mathematical and Computer Modelling 17 (2), 1–18.

[45] Yates, J., 2013. Network Interdiction Methods and Approximations in a Hazmat Transportation Setting. In: Batta, R., Kwon, C. (Eds.), Handbook of OR/MS Models in Hazardous Materials Transportation. Vol. 193 of International Series in Operations Research & Management Science. Springer New York, pp. 187–243.

[46] Zonouz, S. A., Khurana, H., Sanders, W. H., Yardley, T. M., Jun. 2009. RRE: A game-theoretic intrusion Response and Recovery Engine. In: Dependable Systems and Networks, 2009. DSN 09. IEEE/IFIP International Conference on. IEEE, pp. 439–448.