Determining Placement of Intrusion Detectors for a Distributed Application through Bayesian Network Modeling

Gaspar Modelo-Howard, Saurabh Bagchi, and Guy Lebanon

School of Electrical and Computer Engineering, Purdue University
465 Northwestern Avenue, West Lafayette, IN 47907 USA
{gmodeloh,sbagchi,lebanon}@purdue.edu

Abstract. To secure today's computer systems, it is critical to have different intrusion detection sensors embedded in them. The complexity of *distributed* computer systems makes it difficult to determine the appropriate configuration of these detectors, i.e., their choice and placement. In this paper, we describe a method to evaluate the effect of the detector configuration on the accuracy and precision of determining security goals in the system. For this, we develop a Bayesian network model for the distributed system, from an attack graph representation of multi-stage attacks in the system. We use Bayesian inference to solve the problem of determining the likelihood that an attack goal has been achieved, *given* a certain set of detector alerts. We quantify the overall detection performance in the system for different detector settings, namely, choice and placement of the detectors, their quality, and levels of uncertainty of adversarial behavior. These observations lead us to a greedy algorithm for determining the optimal detector settings in a large-scale distributed system. We present the results of experiments on Bayesian networks representing two real distributed systems and real attacks on them.

Key words: Intrusion detection, detector placement, Bayesian networks, attack graph

1 Introduction

It is critical to provide intrusion detection to secure todays distributed computer systems. The overall intrusion detection strategy involves placing multiple detectors at different points of the system, at network ingress or combination points, specific hosts executing parts of the distributed system, or embedded in specific applications that form part of the distributed system. At the current time, the placement of the detectors and the choice of the detectors are more an art than a science, relying on expert knowledge of the system administrator.

The impact of the choice is significant on the accuracy and precision of the overall detection function in the system. The detectors are of different qualities, in terms of their false positive (FP) and false negative (FN) rates, some may have overlapping functionalities, and there may be many possible positions for

deploying a detector. Therefore the entire space of exploration is large and yet not much exists today to serve as a scientific basis for the choices. This paper is a step in that direction.

In the choice of the number of detectors, more is not always better. There are several reasons why an extreme design choice of a detector at every possible network point, host, and application may not be ideal. First, there is the economic cost of acquiring, configuring, and maintaining the detectors. Detectors are well-known to need tuning to achieve their best performance and to meet the targeted needs of the application (specifically in terms of the false positive-false negative performance balance). Second, a large number of detectors would mean a large number of alert streams under attack as well as benign conditions. These could overwhelm the manual or automated process in place to respond to intrusion alerts. Third, detectors impose a performance penalty on the distributed system that they are meant to protect. The penalty arises because the detectors typically share the computational cycles and the bandwidth along with the application. Fourth, a system owner may have specific security goals, e.g., detecting a security goal may be very important and requires high sensitivity, while another may need to be done with less tolerance for false positives.

The problem that we address in this paper is, given the security goals in a system and a model for the way multi-stage attacks can spread in the system, how can we automatically and based on scientific principles, select the right set of detectors and their placements. Right is determined by an application-specific requirement on the true positive (TP) - true negative (TN) rate of detection in the system. We explore the space of the configuration of the individual detectors, their placement on the different hosts or network points, and their number.

Our solution approach starts with attack graphs, which are a popular representation for multi-stage attacks [9]. Attack graphs are a graphical representation of the different ways multi-stage attacks can be launched against system. The nodes depict successful intermediate attack goals with the end nodes representing the ultimate goal of an attack. The edges represent the relation that one attack goal is a stepping stone to another goal and will thus have to be achieved before the other. The nodes can be represented at different levels of abstraction, thus the attack graph representation can bypass the criticism that detailed attack methods and steps will need to be known a priori to be represented (which is almost never the case for reasonably complex systems). Research in the area of attack graphs has included automation techniques to generate these graphs [11], [25], to analyze them [14], [21], and to reason about the completeness of these graphs [14].

We model the probabilistic relation between attack steps and the detectors using the statistical Bayesian network formalism. Bayesian network is particularly appealing in this setting since it enables computationally efficient inference for the unobserved nodes—the attack goals—based on the observed nodes—the detector alerts. The important question that Bayesian inference can answer for us is, given a set of detector alerts, what is the likelihood that an attack goal has been achieved. Further the Bayesian network can be relatively easily created

from an attack graph structure for the system, which we assume is given by existing methods.

We design an algorithm to systematically perform Bayesian inference and determine the accuracy and precision for determining that attack goals have been achieved. The algorithm then chooses the number, placement, and choice of detectors that gives the highest value of an application-specific utility function. We apply our technique to two specific systems—a distributed e-commerce system and a Voice-over-IP (VoIP) system and demonstrate the optimal choice under different conditions. The conditions we explore are different qualities of detectors, different level of knowledge of attack paths, and different threshold settings by the system administrator for determining if an attack goal is reached. Our exploration also shows that the value of a detector for determining an attack step degrades exponentially with distance from the site of the attack.

The rest of this document is organized as follows. Section 2 introduces the attack graphs model and provides a brief presentation of inference in Bayesian networks. Section 3 describes the model and algorithm used to determine an appropriate location for detectors. Section 4 provides a description of the systems used in our experiments. Section 5 presents a complete description of the experiments along with their motivations to help determine the location of the intrusion detectors. Section 6 presents related work and section 7 concludes the paper and discusses future work.

2 Background

2.1 Attack Graphs

An attack graph is a representation of the different methods by which a distributed system can be compromised. It represents the intermediate attack goals for a hypothetical adversary leading up to some high level attack goals. The attack goal may be in terms of violating one or more of confidentiality, integrity, or availability of a component in the system. It is particularly suitable for representing multi-stage attacks, in which a successful attack step (or steps) is used to achieve success in a subsequent attack step. An edge will connect the antecedent (or precondition) stage to the consequent (or postcondition) stage. To be accurate, this discussion reflects the notion of one kind of attack graph, called the exploit-dependency attack graph [11], [14], [25], but this is by far the most common type and considering the other subclass will not be discussed further in this paper.

Recent advances in attack graph generation have been able to create graphs for systems of up to hundreds and thousands of hosts [11], [25].

For our detector-location framework, exploit-dependency attack graphs are used as the base graph from which we build the Bayesian network. For the rest of this paper, the vertex representing an exploit in the distributed system will be called an attack step.

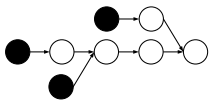

Fig. 1: Attack graph model for a sample web server. There are three starting vertices, representing three vulnerabilities found in different services of the server, from where the attacker can elevate the privileges in order to reach the final goal of compromising the password file.

2.2 Inference in Bayesian Networks

Bayesian networks [13] provide a convenient framework for modeling the relationship between attack steps and detector alerts. Using Bayesian networks we can infer which unobserved attack steps have been achieved based on the observed detector alerts.

Formally, a Bayesian network is a joint probabilistic model for n random variables $(x_1, \ldots, x_n)$ based on a directed acyclic graph $G = (V, E)$ where V is a set of nodes corresponding to the variables $V = (x_1, \ldots, x_n)$ and $E \subseteq VxV$ contains directed edges connecting some of these nodes in an acyclic manner. Instead of weights, the graph edges are described by conditional probabilities of nodes given their parents that are used to construct a joint distribution $P(V)$ or $P(x_1, \ldots, x_n)$.

There are three main tasks associated with Bayesian networks. The first is inferring values of variables corresponding to nodes that are unobserved given values of variables corresponding to observed nodes. In our context this corresponds to predicting whether an attack step has been achieved based on detector alerts. The second task is learning the conditional probabilities in the model based on available data which in our context corresponds to estimating the reliability of the detectors and the probabilistic relations between different attack steps. The third task is learning the structure of the network based on available data. All three tasks have been extensively studied in the machine learning literature and, despite their difficulty in the general case, may be accomplished relatively easily in the case of a Bayesian network.

We focus in this paper mainly on the first task. For the second task, to estimate the conditional probabilities, we can use characterization of the quality of detectors [20] and the perceived difficulty of achieving an attack step, say through risk assessment. We consider the fact that the estimate is unlikely to be perfectly accurate and provide experiments to characterize the loss in performance due to imperfections. For the third task, we rely on extensive prior work on attack graph generation and provide a mapping from the attack graph to the Bayesian network.

In our Bayesian network, the network contains nodes of two different types $V = V_a \bigcup V_b$. The first set of nodes V_a corresponds to binary variables indicating whether specific attack steps in the attack graph occurred or not. The second set of nodes V_b corresponds to binary variables indicating whether

a specific detector issued an alert. The first set of nodes representing attack steps are typically unobserved while the second set of nodes corresponding to alerts are observed and constitute the evidence. The Bayesian network defines a joint distribution $P(V) = P(V_a, V_b)$ which can be used to compute the marginal probability of the unobserved values $P(V_a)$ and the conditional probability $P(V_a|V_b) = P(V_a, V_b)/P(V_b)$ of the unobserved values given the observed values. The conditional probability $P(V_a|V_b)$ can be used to infer the likely values of the unobserved attack steps given the evidence from the detectors. Comparing the value of the conditional $P(V_a|V_b)$ with the marginal $P(V_a)$ reflects the gain in information about estimating successful attack steps given the current set of detectors. Alternatively, we may estimate the suitability of the detectors by computing classification error rate, precision, recall and Receiver Operating Characteristic (ROC) curve associated with the prediction of V_a based on V_b.

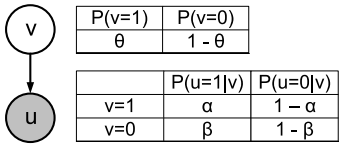

Fig. 2: Simple Bayesian network with two types of nodes: an observed node (u) and an unobserved node (v). The observed node correspond to the detector alert in our framework and its conditional probability table includes the true positive (α) and false positive (β).

Note that the analysis above is based on emulation done prior to deployment with attacks injected through the vulnerability analysis tools, a plethora of which exist in the commercial and research domains, including integrated infrastructures combining multiple tools.

Some attack steps have one or more detectors that specifically measure whether an attack step has been achieved while other attack steps do not have such detectors. We create an edge in the Bayesian network between nodes representing attack steps and nodes representing the corresponding detector alerts. Consider a specific pair of nodes $v \in V_a, u \in V_b$ representing an attack step and a corresponding detector alert. The conditional probability $P(v|u)$ determines the values $P(v = 1|u = 0), P(v = 0|u = 1), P(v = 0|u = 0), P(v = 1|u = 1)$. These probabilities representing false negative, false positive, and correct behavior (last two) can be obtained from an evaluation of the detectors quality.

3 System Design

3.1 Framework Description

Our framework uses a Bayesian network to represent the causal relationships between attack steps and also between attack steps and detectors. Such relation-

ships are expressed quantitatively, using conditional probabilities. To produce the Bayesian network[1], an attack graph is used as input. The structure of the attack graph maps exactly to the structure of the Bayesian network. Each node in the Bayesian network can be in one of two states. Each attack stage node can either be achieved or not by the attacker. Each detector node can be in one of two states: alarm generated state or not. The leaf nodes correspond to the starting stages of the attack, which do not need any precondition, and the end nodes correspond to end goals for an adversary. Typically, there are multiple leaf nodes and multiple end nodes.

The Bayesian network requires that the sets of vertices and directed edges form a directed acyclic graph (DAG). This property is also found in attack graphs. The idea is that the attacker follows a monotonic path, in which an attack step does not have to be revisited after moving to a subsequent attack step. This assumption can be considered reasonable in many scenarios according to experiences from real systems.

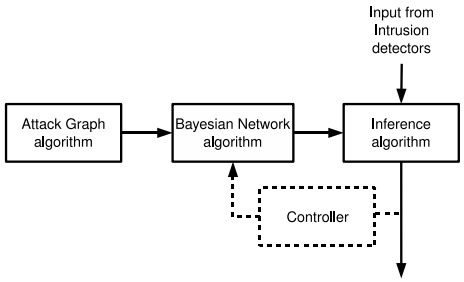

Fig. 3: A block diagram of the framework to determine placement of intrusion detectors. The dotted lines indicate a future component, controller, not included currently in the framework. It would provide for a feedback mechanism to adjust location of detectors.

A Bayesian network quantifies the causal relation that is implied by an edge in an attack graph. In the cases when an attack step has a parent, determined by the existence of an edge coming to this child vertex from another attack step, a conditional probability table is attached to the child vertex. As such, the probability values for each state of the child are conditioned by the state(s) of the parent(s). In these cases, the conditional probability is defined as the probability of a packet from an attacker that already achieved the parent attack step, achieving the child attack step. All values associated to the child are included in a conditional probability table (CPT). As an example, all values for node u in Figure 2 are conditioned on the possible states of its parent, node v. In conclusion, we are assuming that the path taken by the attacker is fully probabilistic. The attacker is following a strategy to maximize the probability of

[1] Henceforth, when we refer to a node, we mean a node in the Bayesian network, as opposed to a node in the attack graph. The clarifying phrase is thus implied.

success, to reach the security goal. To achieve it, the attacker is well informed about the vulnerabilities associated to a component of the distributed system and how to exploit it. The fact that an attack graph is generated from databases of vulnerabilities support this assumption.

The CPTs have been estimated for the Bayesian networks created. Input values are a mixture of estimates based on testing specific elements of the system, like using a certain detector such as IPTables [12] or Snort [28], and subjective estimates, using judgment of a system administrator. From the perspective of the expert (administrator), the probability values reflect the difficulty of reaching a higher level attack goal, having achieved some lower level attack goal.

A potential problem when building the Bayesian network is to obtain a good source for the values used in the CPTs of all nodes. The question is then how to deal with possible imperfect knowledge when building Bayesian networks. We took two approaches to deal with this issue: (1) use data from past work and industry sources and (2) evaluate and measure in our experiments the impact such imperfect knowledge might have.

For the purposes of the experiments explained in section 5, we have chosen the junction tree algorithm to do inference, the task of estimating probabilities given a Bayesian network and the observations or evidence. There are many different algorithms that could be chosen, making different tradeoffs between speed, complexity, and accuracy. Still, the junction tree engine is a general-purpose inference algorithm well suited for our experiments since it works under our scenario: allows discrete nodes, as we have defined our two-states nodes, in direct acyclic graphs such as Bayesian networks, and does exact inference. This last characteristic refers to the algorithm computing the posterior probability distribution for all nodes in network, given some evidence.

3.2 Algorithm

We present here an algorithm to achieve an optimal choice and placement of detectors. It takes as input (i) a Bayesian network with all attack vertices, their corresponding CPTs and the host impacted by the attack vertex; (ii) a set of detectors, the possible attack vertices each detector can be associated with, and the CPTs for each detector with respect to all applicable attack vertices.

Input: (i) Bayesian network $BN = (V, CPT(V), H(V))$ where V is the set of attack vertices, $CPT(V)$ is the set of conditional probability tables associated with the attack vertices, and $H(V)$ is the set of hosts affected if the attack vertex is achieved.

(ii) Set of detectors $D = (d_i, V(d_i), CPT[i][j])$ where d_i is the ith detector, $V(d_i)$ is the set of attack vertices that the detector d_i can be attached to (i.e., the detector can possibly detect those attack goals being achieved), and $CPT[i][j]$ $\forall j \in V(d_i)$ is the CPT tables associated with detector i and attack vertex j.

Output: Set of tuples $\theta = (d_i, \pi_i)$ where d_i is the ith detector selected and π_i is the set of attack vertices that it is attached to.

DETECTOR-PLACEMENT (BN, D)
1 System-Cost = 0
2 Sort all $(d_i, a_j), a_j \in V(d_i), \forall i$ by BENEFIT(d_i, a_j). Sorted list = L
3 Length$(L) = N$
4 for $(i = 1 to N)$
5 System-Cost = System-Cost + COST(d_i, a_j)
6 /* COST(d_i, a_j) can be in terms of economic cost, cost due to false alarms and missed alarms, etc. */
7 if (System-Cost > Threshold τ) break
8 if $(d_i \in \theta)$ add a_j to $\pi_i \in \theta$
9 else add $(d_i, \pi_i = a_j)$ to θ
10 end for
11 return θ

BENEFIT (d, a)
/* This is to calculate the benefit from attaching detector d to attack vertex a */
1 Let the end attack vertices in the BN be $F = f_i, i = 1, \ldots, M$
2 For each f_i, the following cost-benefit table exists
3 Perform Bayesian inference with d as the only detector in the network and connected to attack vertex a
4 Calculate for each f_i, the precision and recall, call them, Precision(f_i, d, a), Recall(f_i, d, a)
5 System-Benefit $= \sum_{i=1}^{M}$ [Benefit$_{f_i}$(True Negative) $\times$ Precision(f_i, d, a) + Benefit$_{f_i}$(True Positive) $\times$ Recall(f_i, d, a)]
6 return System-Benefit

The algorithm starts by sorting all combinations of detectors and their associated attack vertices according to their benefit to the overall system (line 2). The system benefit is calculated by the BENEFIT function. This specific design considers only the end nodes in the BN, corresponding to the ultimate attack goals. Other nodes that are of value to the system owner may also be considered. Note that a greedy decision is made in the BENEFIT calculation each detector is considered singly. From the sorted list, (detector, attack vertex) combinations are added in order, till the overall system cost due to detection is exceeded (line 7). Note that we use a cost-benefit table (line 2 of BENEFIT function), which is likely specified for each attack vertex at the finest level of granularity. One may also specify it for each host or each subnet in the system.

The worst-case complexity of this algorithm is $O(dv\ B(v, CPT(v)) + dv \log(dv) + dv)$, where d is the number of detectors and v is the number of attack vertices. $B(v, CPT(v))$ is the cost of Bayesian inference on a BN with v nodes and $CPT(v)$ defining the edges. The first term is due to calling Bayesian inference with up to d times v terms. The second term is the sorting cost and the third term is the cost of going through the for loop dv times. In practice, each detector will be applicable to only a constant number of attack vertices and therefore the

dv terms can be replaced by a constant times d, which will be only d considering order statistics.

The reader would have observed that the presented algorithm is greedy-choice of detectors is done according to a pre-computed order, in a linear sweep through the sorted list L (the for loop starting in line 4). This is not guaranteed to provide an optimal solution. For example, detectors d_2 and d_3 taken together may provide greater benefit even though detector d1 being ranked higher would have been considered first in the DETECTOR-PLACEMENT algorithm. This is due to the observation that the problem of optimal detector choice and placement can be mapped to the 0-1 knapsack problem which is known to be NP-hard. The mapping is obvious consider $D \times A$ (D: Detectors and A: Attack vertices). We have to include as many of these tuples so as to maximize the benefit without the cost exceeding , the system cost of detection.

4 Experimental Systems

We created three Bayesian networks for our experiments modeling two real systems and one synthetic network. These are a distributed electronic commerce (e-commerce) system, a Voice-over-IP (VoIP) network, and a synthetic generic Bayesian network that is larger than the other two. The Bayesian networks were manually created from attack graphs that include several multi-step attacks for the vulnerabilities found in the software used for each system. These vulnerabilities are associated with specific versions of the particular software, and are taken from popular databases [6], [23]. An explanation for each Bayesian network follows.

4.1 E-Commerce System

The distributed e-commerce system used to build the first Bayesian network is a three tier architecture connected to the Internet and composed of an Apache web server, the Tomcat application server, and the MySQL database backend. All servers are running a Unix-based operating system. The web server sits in a demilitarized zone (DMZ) separated by a firewall from the other two servers, which are connected to a network not accessible from the Internet. All connections from the Internet and through servers are controlled by the firewall. Rules state that the web and application servers can communicate, as well as the web server can be reached from the Internet. The attack scenarios are designed with the assumption that the attacker is an external one and thus her starting point is the Internet. The goal for the attacker is to have access to the MySQL database (specifically access customer confidential data such as credit card information node 19 in the Bayesian network of Figure 4).

As an example, an attack step would be a portscan on the application server (node 10). This node has a child node, which represents a buffer overflow vulnerability present in the rpc.statd service running on the application server (node 12). The other attack steps in the network follow a similar logic and represent

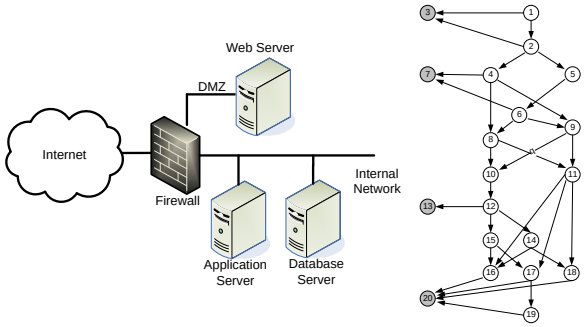

Fig. 4: Network diagram for the e-commerce system and its corresponding Bayesian network. The white nodes are the attack steps and the gray nodes are the detectors.

other phases of an attack to the distributed system. The system includes four detectors: IPtables, Snort, Libsafe, and a database IDS. As shown in Figure 4, each detector has a causal relationship to at least one attack step.

4.2 Voice-over-IP (VoIP) System

The VoIP system used to build the second network has a few more components, making the resulting Bayesian network more complex. The system is divided into three zones: a DMZ for the servers accessible from the Internet, an internal network for local resources such as desktop computers, mail server and DNS server, and an internal network only for VoIP components. This separation of the internal network into two units follows the security guidelines for deploying a secure VoIP system [18].

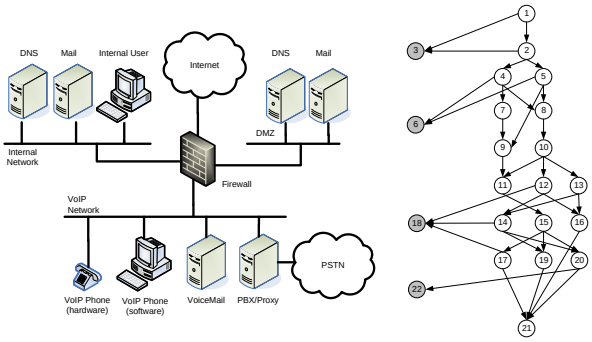

Fig. 5: VoIP system and its corresponding Bayesian network.

The VoIP network includes a PBX/Proxy, voicemail server and software-based and hardware-based phones. A firewall provides all the rules to control

	Attack = True	*Attack = False*
Detection = True	*TP*	*FP*
Detection = False	*FN*	*TN*

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

Fig. 6: Parameters used for our experiments: True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN), precision, and recall.

the traffic between zones. The DNS and mail servers in the DMZ are the only accessible hosts from the Internet. The PBX server can route calls to the Internet or to a public-switched telephone network (PSTN). The ultimate goal of this multi-stage attack is to eavesdrop on VoIP communication. There are 4 detectors Iptables, and three network IDSs on the different subnets.

A third synthetic Bayesian network was built to test our framework for experiments where a larger network, than the other two, was required. This network is shown in Figure 7(a).

5 Experiments

The correct number, accuracy, and location of the detectors can provide an advantage to the systems owner when deploying an intrusion detection system. Several metrics have been developed for evaluation of intrusion detection systems. In our work, we concentrate on the precision and recall. Precision is the fraction of true positives determined among all attacks flagged by the detection system. Recall is the fraction of true positives determined among all real positives in the system. The notions of true positive, false positive, etc. are shown in Figure 6. We also plot the ROC curve which is a traditional method for characterizing detector performanceit is a plot of the true positive against the false positive.

For the experiments we create a dataset of 50,000 samples or attacks, based on the respective Bayesian network. We use the Matlab Bayesian network toolbox [3] for our Bayesian inference and sample generation. Each sample consists of a set of binary values, for each attack vertex and each detector vertex. A one (zero) value for an attack vertex indicates that attack step was achieved (not achieved) and a one (zero) value for a detector vertex indicates the detector generated (did not generate) an alert. Separately, we perform inference on the Bayesian network to determine the conditional probability of different attack vertices. The probability is then converted to a binary determination whether the detection system flagged that particular attack step or not, using a threshold. This determination is then compared with reality, as given by the attack samples which leads to a determination of the systems accuracy. There are several experimental parameters which specific attack vertex is to be considered, the threshold, CPT values, etc. and their values (or variations) are mentioned in the appropriate experiment. The CPTs of each node in the network are manually configured according to the authors experience administering security for

distributed systems and frequency of occurrences of attacks from references such as vulnerability databases, as mentioned earlier.

5.1 Experiment 1: Distance from Detectors

The objective of experiment 1 was to quantify for a system designer what is the gain in placing a detector close to a service where a security event may occur. Here we used the synthetic network since it provided a larger range of distances between attack steps and detector alerts.

The CPTs were fixed to manually determined values on each attack step. Detectors were used as evidence, one at a time, on the Bayesian network and the respective conditional probability for each attack node was determined. The effect of the single detector on different attack vertices was studied, thereby varying the distance between the node and the detector. The output metric is the difference of two terms. The first term is the conditional probability that the attack step is achieved, conditioned on a specific detector firing. The second term is the probability that the attack step is achieved, without use of any detector evidence. The larger the difference is, the greater is the value of the information provided by the detector. In Figure 7(b), we show the effect due to detector corresponding to node 24 and in Figure 7(c), we consider all the detectors (again one at a time). The effect of all the detectors shows that the conclusions from node 24 are general.

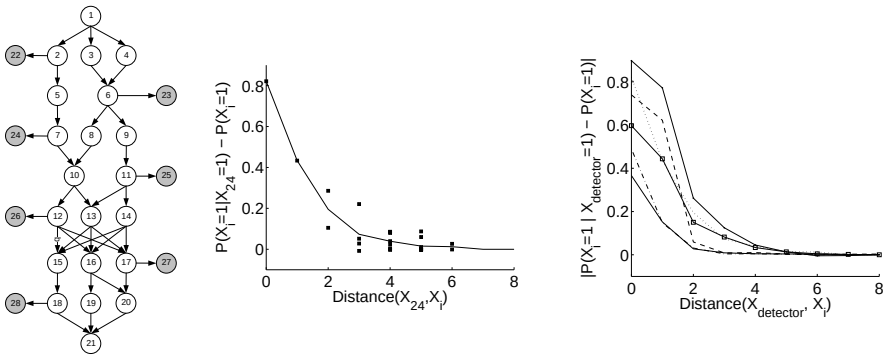

Fig. 7: Results of experiment 1: Impact of distance to a set of attack steps. (a) Generic Bayesian network used. (b) Using node 24 as the detector (evidence), the line shows mean values for rate of change. (c) Comparison between different detectors as evidence, showing the mean rate of change for case.

The results show that a detector can affect nodes inside a radius of up to three edges from the detector. The change in probability for a node within this radius, compared to one outside the radius, can be two times greater when the detector is used as evidence. For all Bayesian networks tested, the results were consistent to the three edges radius observation.

5.2 Experiment 2: Impact of Imperfect Knowledge

The objective of experiment 2 was to determine the performance of the detection system in the face of attacks. In the first part of the experiment *(Exp 2a)*, the effect of the threshold, that is used in converting the conditional probability of an attack step into a binary determination, is studied. This corresponds to the practical situation that a system administrator has to make a binary decision based on the result of a probabilistic framework and there is no oracle at hand to help. For the second part of the experiment *(Exp 2b)*, the CPT values in the Bayesian network are perturbed by introducing variances of different magnitudes. This corresponds to the practical situation that the system administrator cannot accurately gauge the level of difficulty for the adversary to achieve attack goals. The impact of the imperfect knowledge is studied through a ROC curve.

For Exp 2a, precision and recall were plotted as a function of the threshold value. This was done for all the attack nodes in the Bayesian network and the results for a representative sample of six nodes are shown in Figure 8. We used threshold values from 0.5 to 0.95, since anything below 0.5 would imply the Bayesian network is useless in its predictive ability.

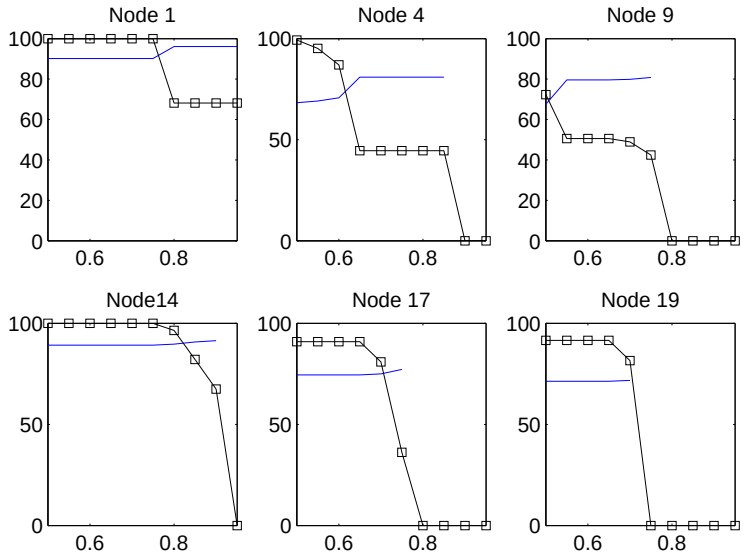

Fig. 8: Precision and recall as a function of detection threshold, for the e-commerce Bayesian network. The line with square markers is recall and other line is for precision.

Expectedly, as the threshold is increased, there are fewer false positives and the precision of the detection system improves. The opposite is true for the recall of the system since there are more false negatives. However, an illuminating

observation is that the precision is relatively insensitive to the threshold variation while the recall has a sharp cutoff. Clearly, the desired threshold is to the left of the cutoff point. Therefore, this provides a scientific basis for an administrator to set the threshold for drawing conclusions from a Bayesian network representing the system.

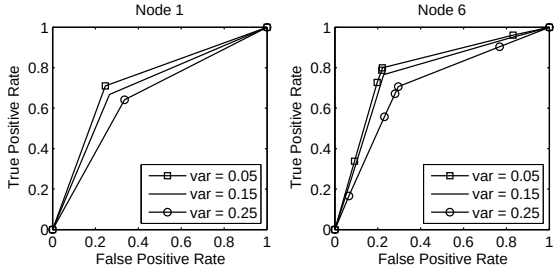

Fig. 9: ROC curves for two attack steps in e-commerce Bayesian network. Each curve corresponds to a different variance added to the CTP values.

In experiment 2b we introduced variance to the CPT values of all the attack nodes, mimicking different levels of imperfect knowledge an admin may have about the adversarys attack strategies. When generating the samples corresponding to the attacks, we used three variance values: 0.05, 0.15, and 0.25. Each value could be associated with a different level of knowledge from an administrator: expert, intermediate, and nave, respectively. For each variance value, ten batches of 1,000 samples were generated and the detection results were averaged over all batches.

In Figure 9, we show the ROC curves for nodes 1 and 6 of the e-commerce system, with all four detectors in place. Expectedly, as the variance increases, the performance suffers. However, the process of Bayesian inference shows an inherent resilience since the performance does not degrade significantly with the increase in variance. For node 1, several points are placed so close together that only one marker shows up. On the contrary, for node 6, multiple well spread out TP-FP value pairs are observed. We hypothesize that since node 1 is directly connected to the detector node 3, its influence over node 1 dominates that of all other detectors. Hence fewer number of sharp transitions are seen compared to node 6, which is more centrally placed with respect to multiple detectors.

Experiment 2c also looked at the impact of imperfect knowledge when defining the CPT values in the Bayesian network. Here we progressively changed the CPT values for several attack steps in order to determine how much we would deviate from the correct value. We used two values 0.6 and 0.8 for each CPT cell (only two are independent) giving rise to four possible CPT tables for each node. We plot the minimum and maximum conditional probabilities for a representative attack node for a given detector flagging. We change the number of CPTs that we perturb from the ideal values. Expectedly as the number of CPTs

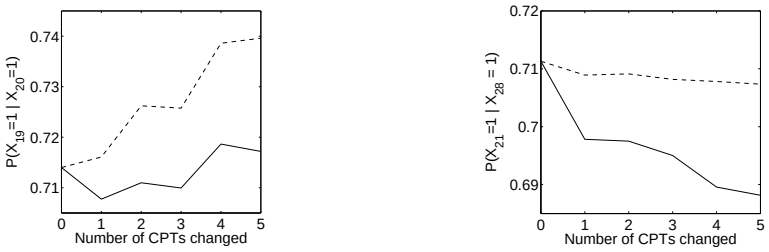

Fig. 10: Impact of deviation from correct CPT values, for the (a) e-commerce and (b) generic Bayesian networks.

changed increases, the difference between the minimum and the maximum increases, but the range is within 0.03. Note that the point at the left end of the curve for zero CPTs changed gives the correct value.

Both experiments indicate that the BN formalism is relatively robust to imperfect assumptions concerning the CPT values. This is an important fact since it is likely that the values determined by an experienced system administrator would still be somewhat imperfect. Overall, as long as the deviation of the assumed CPTs from the truth is not overwhelming, the network performance degrades gracefully.

5.3 Experiment 3: Impact on Choice and Placement of Detectors

The objective of experiment 3 was to determine the impact of selecting the detectors and their corresponding locations. To achieve this, we ran experiments on the e-commerce and the VoIP Bayesian networks to determine a pair of detectors that would be most effective. This pair, called the optimal pair, is chosen according to the algorithm described in Section 3.2. The performance of the optimal pair is compared against additional pairs selected at random. We show the result using the ROC curve for the two ultimate attack goals, namely node 19 and node 21 in the e-commerce and the VoIP systems.

To calculate the performance of each pair of detectors, we created 10,000 samples from each Bayesian network, corresponding to that many actual attacks. Then we performed Bayesian inference and calculated the conditional probability of the attack step, given the pair of detectors. We determined the true positive rate and false positive rate by sweeping across threshold values.

Results show that the pair of detectors determined from the algorithm performs better than the other randomly selected pairs. Figure 11a shows the situation in which a single detector (d_{20}) attached to two attack nodes (x_{19}, x_{18}) performs better than two detectors (d_{13} and d_7, or d_{12} and d_3). The placement of the detector d_{20} affects the performance. This can be explained by the fact that node 18 is more highly connected in the attack graph and therefore attaching detector d_{20} to that node, rather than node 16, provides better predictive performance.

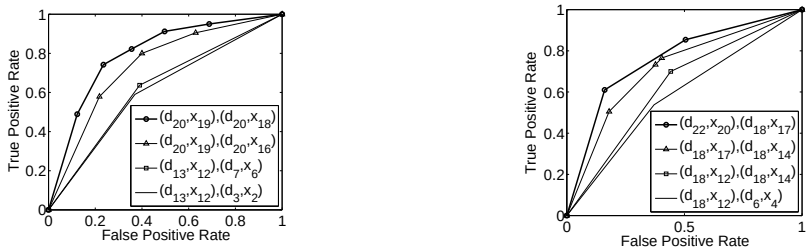

Fig. 11: ROC curves for detection of attack steps, using pairs of detectors, in the e-commerce network (left) and the VoIP network (right).

There is a cost of adding detectors to a system, but there is also a cost of having a detector attached to more attack nodes, in terms of the bandwidth and computation. Thus adding further edges in the Bayesian network between a detector node and an attack node, even if feasible, may not be desirable. For the VoIP network, detector pair d_{22} and d_{18} performs best. This time two separate detectors outperform a single high quality detector (d_{18}) connected to two nodes.

Further details on all experiments performed, including all the probability values used for the Bayesian networks, are available at [22]. These are omitted here due to space constraints and the interested party is welcome to further read. All the experiments validate the intuition behind our algorithm that the greedy choice of the detectors also gives good results when multiple detectors are considered together and over the entire Bayesian network.

6 Related Work

Bayesian networks have been used in intrusion detection to perform classification of events. Kruegel et al. [17] proposed the usage of Bayesian networks to reduce the number of false alarms. Bayesian networks are used to improve the aggregation of different model outputs and allow integration of additional information. The experimental results show an improvement in the accuracy of detections, compared to threshold-based schemes. Ben Amor et al. [4] studied the use of nave Bayes in intrusion detection, which included a performance comparison with decision trees. Due to similar performance and simpler structure, nave Bayes is an attractive alternative for intrusion detection. Other researchers have also used nave Bayesian inference for classifying intrusion events [29].

To the best of our knowledge, the problem of determining an appropriate location for detectors has not been systematically explored by the intrusion detection community. However, analogous problems have been studied to some extent in the physical security and the sensor network fields.

Jones *et al.* [15] developed a Markov Decision Process (MDP) model of how an intruder might try to penetrate the various barriers designed to protect a physical facility. The model output includes the probability of a successful intrusion and

the most likely paths for success. These paths provide a basis to determine the location of new barriers to deter a future intrusion.

In the case of sensor networks, the placement problem has been studied to identify multiple phenomena such as determining location of an intrusion [1], contamination source [5], [27], and atmospheric conditions [16]. Anjum *et al.* [1] determined which nodes should act as intrusion detectors in order to provide detection capabilities in a hierarchical sensor network. The adversary is trying to send malicious traffic to a destination node (say, the base node). In their model, only some nodes called tamper-resistant nodes are capable of executing a signature-based intrusion detection algorithm and these nodes cannot be compromised by an adversary. Since these nodes are expensive, the goal is to minimize the number of such nodes and the authors provide a distributed approximate algorithm for this based on minimum cut-set and minimum dominating set. The solution is applicable to a specific kind of topology, widely used in sensor networks, namely clusters with a cluster head in each cluster capable of communicating with the nodes at the higher layer of the network hierarchy.

In [5], the sensor placement problem is studied to detect the contamination of air or water supplies from a single source. The goal is to detect that contamination has happened and the source of the contamination, under the constraints that the number of sensors and the time for detection are limited. The authors show that the problem with sensor constraint or time constraint are both NP-hard and they come up with approximation algorithms. They also solve the problem exactly for two specific cases, the uniform clique and rooted trees. A significant contribution of this work is the time efficient method of calculating the sensor placement. However, several simplifying assumptions are made—sensing is perfect and no sensor failure (either natural or malicious) occurs, there is a single contaminating source, and the flow is stable.

Krause *et al.* [16] also point out the intractability of the placement problem and present a polynomial-time algorithm to provide near-optimal placement which incurs low communication cost between the sensors. The approximation algorithm exploits two properties of this problem: submodularity, formalizing the intuition that adding a node to a small deployment can help more than adding a node to a large deployment; and locality, under which nodes that are far from each other provide almost independent information. In our current work, we also experienced the locality property of the placement problem. The proposed solution *learns* a probabilistic model (based on Gaussian processes) of the underlying phenomenon (variation of temperature, light, and precipitation) and for the expected communication cost between any two locations from a small, short-term initial deployment.

In [27], the authors present an approach for determining the location in an indoor environment based on which sensors cover the location. The key idea is to ensure that each resolvable position is covered by a unique set of sensors, which then serves as its signature. They make use of identifying code theory to reduce the number of active sensors required by the system and yet provide unique localization for each position. The algorithm also considers robustness,

in terms of the number of sensor failures that can be corrected, and provides solutions in harsh environments, such as presence of noise and changes in the structural topology. The objective for deploying sensors here is quite different from our current work.

For all the previous work on placement of detectors, the authors are looking to detect events of interest, which propagate using some well-defined models, such as, through the cluster head *en route* to a base node. Some of the work (such as [16]) is focused on detecting natural events, that do not have a malicious motive in avoiding detection. In our case, we deal with malicious adversaries who have an active goal of trying to bypass the security of the system. The adversaries' methods of attacking the system do not follow a well-known model making our problem challenging. As an example of how our solution handles this, we use noise in our BN model to emulate the lack of an accurate attack model.

There are some similarities between the work done in alert correlation and ours, primarily the interest to reduce the number of alerts to be analyzed from an intrusion. Approaches such as [24] have proposed modeling attack scenarios to correlate alerts and identify causal relationships among the alerts. Our work aims to closely integrate the vulnerability analysis into the placement process, whereas the alert correlation proposals have not suggested such importance.

The idea of using Bayes theorem for detector placement is suggested in [26]. No formal definition is given, but several metrics such as accuracy, sensitivity, and specificity are presented to help an administrator make informed choices about placing detectors in a distributed system. These metrics are associated to different areas or sub-networks of the system to help in the decision process.

Many studies have been done on developing performance metrics for the evaluation of intrusion detection systems (IDS), which have influenced our choice of metrics here. Axelsson [2] showed the applicability of estimation theory in the intrusion detection field and presented the Bayesian detection rate as a metric for the performance of an IDS. His observation that the base rate, and not only the false alarm rate, is an important factor on the Bayesian detection rate, was included in our work by using low base rates as part of probability values in the Bayesian network. The MAFTIA Project [8] proposed precision and recall to effectively determine when a vulnerability was exploited in the system. A difference from our approach is that they expand the metrics to consider a set of IDSes and not only a single detector. The idea of using ROC curves to measure performance of intrusion detectors has been explored many times, most recently in [7], [10].

Extensive work has been done for many years with attack graphs. Recent work has concentrated on the problems of generating attack graphs for large networks and automating the process to describe and analyze vulnerabilities and system components to create the graphs. The NetSPA system [11] uses a breath-first technique to generate a graph that grows almost linearly with the size of the distributed system. Ou et al. [25] proposed a graph building algorithm using a formal logical technique that allows to create graphs of polynomial size to the network being analyzed.

7 Conclusions and Future Work

Bayesian networks have proven to be a useful tool in representing complex probability distributions, such as in our case of determining the likelihood that an attack goal has been achieved, given evidence from a set of detectors. By using attack graphs and Bayesian inference, we can quantify the overall detection performance in the systems by looking at different choices and placements of detectors and the detection parameter settings. We also quantified the information gain due to a detector as a function of its distance from the attack step. Also, the effectiveness of the Bayesian networks can be affected by imperfect knowledge when defining the conditional probability values. Nevertheless, the Bayesian network exhibits considerable resiliency to these factors as our experiments showed.

Future work should include looking at the scalability issues of Bayesian networks and its impact on determining the location for a set of detectors in a distributed system. The probability values acquisition problem can be handled by using techniques such as the recursive noisy-OR modeling [19] but experimentation is required to determine its benefits and limitations for our scenario.

Acknowledgments. Gaspar Modelo-Howard was partly supported by an IFARHU-SENACYT Scholarship from the Republic of Panama. Saurabh Bagchi was partly supported in this work by an endowment grant from Purdue's Center for Education and Research in Information Assurance and Security (CERIAS).

References

1. Anjum, F., Subhadrabandhu, D., Sarkar, S., Shetty, R.: On Optimal Placement of Intrusion Detection Modules in Sensor Networks. In: 1st IEEE International Conference on Broadband Networks, pp. 690–699. IEEE Press, New York (2004)
2. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. ACM Trans. Inf. Syst. Secur. 3-3, 186–205 (2000)
3. Bayes Net Toolbox for Matlab, `http://www.cs.ubc.ca/~murphyk/Software`
4. Ben Amor, N., Benferhat, S., Elouedi, Z.: Naive Bayes vs decision trees in intrusion detection systems. In: 19th ACM Symposium on Applied computing, pp. 420–424. ACM Press, New York (2004)
5. Berger-Wolf, T., Hart, W., Saia, J.: Discrete Sensor Placement Problems in Distribution Networks. J. Math. and Comp. Model. 42, 1385-1396 (2005)
6. Bugtraq Vulnerability Database, `http://www.securityfocus.com/vulnerabilities`
7. Cardenas, A., Baras, J., Seamon, K.: A Framework for the Evaluation of Intrusion Detection Systems. In: 27th IEEE Symposium on Security and Privacy, 15 pp. IEEE Press, New York (2006)
8. Dacier, M. (ed.): Design of an Intrusion-Tolerant Intrusion Detection System. Research Report, Maftia Project (2002)
9. Foo, B., Wu, Y., Mao, Y., Bagchi, S., Spafford, E.: ADEPTS: Adaptive Intrusion Response using Attack Graphs in an E-Commerce Environment. In: International Conference on Dependable Systems and Networks, pp. 508–517, (2005)

10. Gu, G., Fogla, P., Dagon, D., Lee, W., Skoric, B.: Measuring Intrusion Detection Capability: An Information-Theoretic Approach. In: 1st ACM Symposium on Information, Computer and Communications Security, pp. 90–101. ACM Press, New York (2006)
11. Ingols, K., Lippmann, R., Piwowarski, K.: Practical Attack Graph Generation for Network Defense. In: 22nd Annual Computer Security Applications Conference, pp. 121–130. IEEE Press, New York (2006)
12. IPTables Firewall, `http://www.netfilters.org/projects/iptables`
13. Jensen, F.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2001)
14. Jha, S., Sheyner, O., Wing, J.: Two Formal Analyses of Attack Graphs. In: 15th IEEE Computer Security Foundations Workshop, pp. 49–63. IEEE Press, New York (2002)
15. Jones, D., Davis, C., Turnquist, M., Nozick, L.: Physical Security and Vulnerability Modeling for Infrastructure Facilities. Technical Report, Sandia National Laboratories (2006)
16. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J.: Near-optimal Sensor Placements: Maximizing Information while Minimizing Communication Cost. In: 5th International Conference on Information Processing in Sensor Networks, pp. 2–10. ACM Press, New York (2006)
17. Krügel, C., Mutz, D., Robertson, W., Valeyr, F.: Bayesian Event Classification for Intrusion Detection. In: 19th Annual Computer Security Applications Conference, pp.14–23. IEEE Press, New York (2003)
18. Kuhn, D., Walsh, T., Fires, S.: Security Considerations for Voice Over IP Systems. Special Publication 800-58, National Institute of Standards and Technology (2005)
19. Lemmer, J., Gossink, D.: Recursive Noisy OR - A Rule for Estimating Complex Probabilistic Interactions. In: IEEE Trans. Syst. Man. Cybern. B. 34, 2252–2261 (2004)
20. Lippmann, R., et al.: Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In: 1st DARPA Information Survivability Conference and Exposition, pp. 81–89, (2000)
21. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking Attack Graphs. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp.127–144. Springer, Heidelberg (2006)
22. Modelo-Howard, G.: Addendum to Determining Placement of Intrusion Detectors for a Distributed Application through Bayesian Network Modeling, `http://cobweb.ecn.purdue.edu/~dcsl/publications/detectors-location_addendum.pdf`
23. National Vulnerability Database, `http://nvd.nist.gov/nvd.cfm`
24. Ning, P., Cui, Y., Reeves, D.: Constructing Attack Scenarios through Correlation of Intrusion Alerts. In: 9th ACM Conference on Computers & Communications Security, pp. 245–254, (2002)
25. Ou, X., Boyer, W., McQueen, M.: A Scalable Approach to Attack Graph Generation. In: 13th ACM Conference on Computer & Communications Security, pp. 336–345, (2006)
26. Peikari, C., Chuvakin, A.: Security Warrior. O'Reilly, New York (2004)
27. Ray, S., Starobinski, D., Trachtenberg, A., Ungrangsi, R.: Robust Location Detection with Sensor Networks. In: IEEE J. on Selected Areas in Comm., 22, pp. 1016–1025 (2004)
28. Snort Intrusion Detection System, `http://www.snort.org`
29. Valdes, A., Skinner, K.: Adaptive, Model-based Monitoring for Cyber Attack Detection. In: Debar, H., Me, L., Wu, S. (eds.) RAID 2000. LNCS, vol. 1907, pp. 80–92. Springer, Heidelberg (2000)