A Knowledge-Based Bayesian Model for Analyzing a System after an Insider Attack

Qutaibah Althebyan, Brajendra Panda

Abstract Many consider insider attacks to be more severe than outsider attacks due to the nature of such attacks that involve people who have knowledge of their own organization. In this work, we presented a new model to evaluate and analyze a system after the occurrence of an insider attack. By evaluating and analyzing the system after detecting such attack, we classified systems' objects into a list of non affected objects and a list of affected objects. We also introduced a new graph called knowledge Bayesian attack graph (KBAG). KBAG represents possible candidate paths that malicious insiders may follow to achieve their goal of compromising critical objects. KBAG also enables us to calculate risk values for different objects using Bayesian inference techniques. These risk values will be considered as measurements for the likelihood of possible occurrence of other insider attacks that have not yet been detected by the underlying system.

1 Introduction

Nowadays, computer security issues have become a great concern for both individuals and organizations. Users have been increasingly concerned about sensitive information that can be revealed if they face an attack. Consequences of attacks can be as small as stealing small portion of an individual's critical information and as severe as destruction of an organization's entire information base. Several kinds of attacks may affect an organization. Among such attacks are insiders' attacks which are considered, according to many experts, among the most devastating ones.

Insider attacks involve individuals who work for an organization and who have justifiable privileges to access sensitive data in the organization. During this kind of attacks privileged individuals accumulate enough knowledge about their organiza-

Qutaibah Althebyan and Brajendra Panda
Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, Arkansas 72701, USA, e-mail: {qaltheb, bpanda}@uark.edu

tion. Both privileges and knowledge help individuals in planning successful attacks while making it difficult for the organization to discover and/or prevent them. In this paper, we assume that such an attack has been detected by the underlying system and it affected a specific object or set of objects. In our effort to resolve the problem and recover the system, we use Bayesian networks [1] as well as knowledge graphs (KGs) and dependency graphs (DGs), which were proposed in [2], to analyze the system that suffered from a successful attack. Analysis of the system includes identification of the source of the attack and after identifying the source, an evaluation of all objects in the system must be performed.

In general, an attack can be of many forms. However, in our work we focus on attacks that involve modification of a given document or a set of documents. It is important to mention that this work is suitable only for insider attacks. So, the reader should realize that attempts to use this work to evaluate a system suffering from an outsider attack may not be successful. Our model helps in narrowing down the investigation process by classifying objects into a list of unaffected objects and a list of affected objects. The set of aunaffected objects can then be made available to users while all affected objects go through further invistigation.

2 Our Model

Many definitions have been proposed for the insider. For example, Mark Maybury et al. [3] introduced malicious insider as ”one motivated to adversely impact an organization’s mission through a range of actions that compromise information confidentiality, integrity, and/or availability”. Boanerges Aleman-Meza1, Phillip Burns et al. [4] mentioned that ”insider threat refers to the potential malevolent actions by employees within an organization, a specific type of which relates to legitimate access of documents.” However, in our model, we define an insider as [2]: *An insider is an individual who has access to and has some knowledge of the organization’s information system.*

The above definition involves individuals with assigned privileges with a concentration on knowledge of insiders. Throughout this paper, we consider any piece of information as an object. Hence, usernames, passwords, dates, paper documents, digital documents, e-mails, etc. are all valid examples of objects. An insider upon accessing an object increases his/her knowledge. We assume that any knowledge an insider gains is saved in his/her knowledgebase, and it is saved as units. Usually insiders have knowledge about their own organization. They know other insiders in the organization and may know others’ responsibilities. This includes the knowledge of the overall network topology of the organization which makes it easy to know where to get information and where to find sensitive data. He/she also gains knowledge by accessing documents through his/her valid account. This accumulated knowledge gives him/her the advantage to extract sensitive data, which he/she is not authorized to get. However, he/she increases these chances by adding to his/her knowledgebase the knowledge of existing dependencies among various objects in the system.

Our model "Knowledge-Based Bayesian Graph Model" uses knowledge graphs (KGs), dependency graphs (DGs), and knowledge Bayesian attack graphs (KBAGs) to analyze the system after a detected attack that is initiated by an insider of the underlying system. Before going into details in describing our model, it is important to give brief descriptions of KGs and DGs in order to familiarize the reader with these concepts. Both KGs and DGs are used in constructing KBAGs and implementing algorithms in our model. We start with a brief description of KGs.

2.1 Knowledge Graphs (KGs)

A Knowledge Graph [2], denoted by KG, is a graph that represents different knowledge units of an insider. Knowledge units are any piece of information an insider gets access to and are saved in his/her knowledgebase. Knowledge units are denoted by K_i. A given insider has a specific KG that is initiated the first time he/she accessed the organizations' resources. It is updated after each access to any object of the system. *Definition: A Knowledge Graph* is a directional graph represented by G(V, E) where V is:

- A set of non-leaf nodes representing the user's knowledge that is gained from objects in the underlying system
- A set of leaf nodes representing the objects that the user has had access to

E is the set of edges among vertices of the graph. An edge E = (V_i, V_j) exists in G iff the knowledge unit in V_j can be obtained from V_i. V_i, V_j, respectively belong to any of the following set of vertices:

- An object O_i and its corresponding knowledge unit K_i
- Two knowledge units K_i and K_j
- A knowledge unit K_i and the vertex of the composed knowledge CK, where the composed knowledge CK is a node in the KG that represents the total knowledge accumulated for the corresponding insider, and equals to the union of knowledge units that exist in the knowledgebase of the insider

Fig. 1 illustrates an example of Knowledge Graph of an insider S_i:

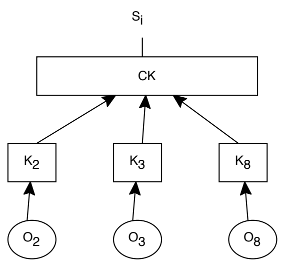


Fig. 1 An Example of a Knowledge-Graph

So, a KG contains knowledge units K$_i$'s that are saved in S$_i$ knowledgebase. We used ontological methods to extract knowledge facts (saved as units) from documents' contents using the concept of relatedness of documents to the current active domain of the corresponding insider. All knowledge units that are extracted from documents related to the user's current domain constitute the insider's knowledge. However, this does not constitute his/her total knowledge. In fact, an insider can obtain a new higher knowledge by combining existing knowledge units. To illustrate, consider the following example of a document d$_1$ represented by table 1(a):

Table 1 Original and related attributes of d$_1$ to the domain of access for S$_i$

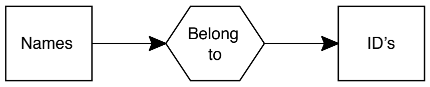
	Name	ID	Age	Salary
1	A	1	30	$1000
⋮	…			
100	K	100	27	$1600

(a) D$_1$'s Attributes.

	Name	ID
1	A	1
⋮	…	
100	K	100

(b) Related Attributes.

Table 1(b) contains the attributes that are related to the current insider's domain. S$_i$ is interested only in names and id's of employees. Since there are 100 names then the set of related attributes (knowledge units) are: U$_1$ $\rightarrow$ A, U$_2$ $\rightarrow$ B,..., U$_{100}$ $\rightarrow$ k. Therefore, 100 knowledge units U$_1$,..., U$_{100}$ are saved in his/her knowledgebase. Another 100 knowledge units (100 ID's) are also saved in his/her knowledge units. That is: U$_{101}$ $\rightarrow$ 1, U$_{102}$ $\rightarrow$ 2,..., U$_{200}$ $\rightarrow$ 100 are another 100 knowledge units. Moreover, S$_i$ can extract a higher level of knowledge from the 200 knowledge units he/she already gained. This can be obtained by combining existing knowledge units which will be represented as new knowledge units. Fig. 2 (a) illustrates the combining of two knowledge units to form a new higher-level knowledge unit:



(a)Combining two knowledge units.

Name-Id
A, 1
…
K, 100

(b) Knowledge unit Name-Id.

Fig. 2 Relation of combining two knowledge units and the corresponding new knowledge unit.

The table in Fig. 2 (b) illustrates the process of combining the two knowledge units: Name and ID which results in knowledge unit "Name-ID". This associates names of employees with their id's. This adds 100 new knowledge units to his/her knowledgebase, giving a total of 300 knowledge units.

2.2 Dependency Graphs (DGs)

A *Dependency graph* (DG) [2] is a global hierarchal graph that shows all dependencies among various objects in the system. Usually, objects, especially documents, are created depending on other objects of the same system. This dependency relationship is an important one through which insiders may predict contents of important objects that they do not have privileges to access. By following dependencies among different objects, insiders may locate critical objects, and hence get a chance to obtain critical information for which they are not authorized to access. A DG contains nodes for all objects in the system. Our concept of a DG is similar to the System Dependency Graph, SDC, presented by Larsen and Harrold [6] for modeling an object-oriented program.

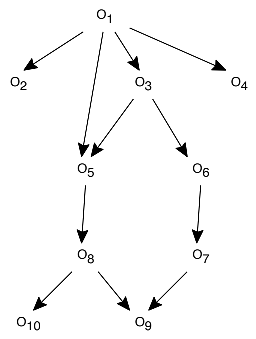


Fig. 3 An Example of a Dependency Graph

Fig. 3 represents an example of a DG of objects. It is important to mention that the DG is a dynamic graph that is updated for every newly created object in the system. A new edge is added between the new object and the object (or objects) which the new object relies on. Also, the DG is updated periodically to reflect operations affecting the objects. A DG is a directional graph in which the direction of the edge indicates the dependency direction which can be verified from Fig. 3. Throughout this paper, we follow the top down approach for representing dependencies among different objects. So, if we consider the dependency relation between any two objects as a function F, then we can verify the following relations from Fig. 3 to be true: $O_3 = F(O_1)$, $O_4 = F(O_1)$, $O_5 = F(O_1, O_3)$,... etc.

It is important to clarify the meaning of dependency relations that exist among nodes in a DG. For example, in Fig. 3 the following relation $O_5 = F(O_1, O_3)$ exists. This relation indicates that O_5 is a function of O_1 and O_3. In reality this relation has a meaning that is consistent with the context for which this relation has been created. Generally speaking, this relation means that some information in O_1 and O_3 has been used to derive some information in O_5. This contributes to some or all knowledge units that O_5 contains. An example of the above argument is:

- Object O_1 contains salary rates of all sets of employees in the organization. Employees are distinguished into several sets that involve categorizing employees with the same rank into the same set
- Object O_3 contains the total number of employees in each set or group
- Object O_5, as a result, uses the information in both objects O_1 and O_3 to calculate the total amount of money consumed for salaries out of the total budget of the organization

Having the knowledge of both O_1 and O_3 gives an insider the ability to ”get the knowledge” of object (O_5) probably without having the right privilege to access O_5.

Before we go into describing KBAGs, it is necessary for the reader to have some familiarity with the concept of Bayesian networks. We start by giving a brief description of Bayesian networks. We then describe KBAGs.

2.3 Bayesian Networks

Bayesian network (or Bayes net) is modeled as a directed acyclic graph over a set of variables which associates these variables with a set of conditional probability distributions, one for each variable [5]. It is a graphical representation which shows probabilistic relationships among a set of variables and is used in modeling situation with uncertainty. Fig. 4 shows a simple Bayesian network with three variables A, B and C. Bayesian network uses the available prior probabilities or knowledge for cal-

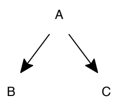


Fig. 4 A simple Bayesian netowrk with three variables

culating new probability values using Bayesian inference techniques. For example, if we consider Fig. 4, then the joint probability distribution function for the above three variables is P(A, B, C), and this joint probability distribution function equals to:

$$P(A, B, C) = P(A/B)*P(C/B)*P(B)$$

The above formula can be extended to any number of variables. It also draws some dependency relations among variables in which a change in the state (value in our case) in one variable results in a change in the state of one or more other variables. The change takes effect when one or more variables of the Bayesian network have a dependent relation with the variable that has been changed. For example, any change in the value of variable B results in an effect in variable A, affecting the joint probability distribution function. Moreover, any change in variable B (for instance) will not have any direct effect in variable C because both B and C are independent.

However, the joint function is still affected by this change because of the dependency relation between C and A which (i.e., A) has a dependency relation with B. In the following section we introduce KBAG which is a basic component in our model of evaluating and analyzing an attack initiated by an insider of the underlying system.

2.4 Knowledge Bayesian Attack Graphs (KBAGs)

Knowledge Bayesian Attack Graph (KBAG) is a top-down directed graph that shows probabilistic relationships among different objects (representing nodes) of the graph. It involves similar characteristics of both Bayesian networks and attack graphs [7, 8, 9]. Each node of a KBAG (represents an object) is assigned a probability value that represents the probability of a successful attack at that node. That is, this probability represents a measurement which quantifies the fact that a given node of KBAG had been attacked (maliciously modified). Different levels of conditional probabilities are assigned to different nodes. More details for how to create KBAGs and how to assign probability values for their nodes are presented in section 3.2.

A KBAG is created using information from the knowledge graph (KG) of an insider and the dependency graph (DG) of different objects in the system. It consists of a set of variables (representing nodes) that represents a set of objects that can be accessed by an insider. These nodes will be associated with probability values as mentioned earlier. It also consists of a set of directed edges that indicates a directed dependency from the parent node to its descendent nodes. Hence, a path from one node to another node represents a dependency relation that exists between the two nodes. Conditional dependency has only the top-down direction which is the direction from the parent node to children nodes.

A candidate path in a KBAG represents a series of accesses for objects. This series of accesses may lead to a successful attack to any node in the underlying system. So, a path from a root node to a descendent intermediate or leaf node represents a candidate for a successful attack. By creating this graph and identifying all possible paths from a root node to either a leaf node or a critical node (a critical node is a node that has critical information and will be called ”hot node”) we can learn how insiders initiate a set of accesses or actions for launching an attack. Further, we can narrow down the list of affected objects by calculating risk values for all nodes in the KBAG. More details on how to create and implement KBAGs is provided in section 3.1.

3 Evaluating and Analyzing the System after an Insider Attack

In this section, we introduce a new algorithm for evaluating and analyzing a system after undergoing an insider attack. We assume that the system has been affected by an attack launched by an insider of the system. Although our model can be extended to cover all kinds of attacks, we limit ourselves to cover only attacks that involve modefying an object. In what follows we introduce a new algorithm which uses KBAGs to calculate risk values for different objects in the system. Through risk values we can find a list of possible affected objects by the same insider in case other attacks have been performed to other objects.

3.1 Creating Knowledge Bayesian Attack Graph (KBAG)

A KBAG is a directed graph that consists of:

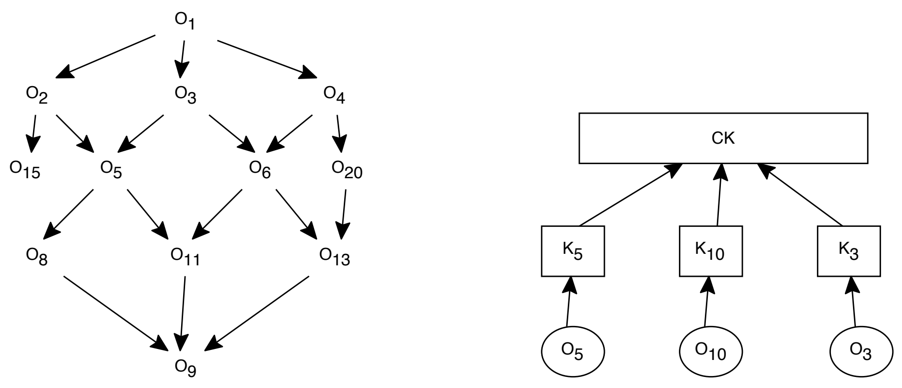
1. A set of nodes each of which representing an object of the system
2. Top-down directed edges among different nodes of KBAG; and
3. Probability values assigned to all nodes.

To build KBAG, both KG and DG are compared in parallel. Nodes that belong to the knowledgebase of the insider create the new set of nodes. That is, part of the nodes of the considered KG represent the set of nodes of the KBAG. Directed edges are obtained by traversing the DG of different objects in the system. That is, we are computing kind of transitivity closure of the DG. After building the KBAG, probability values are assigned to different nodes of the KBAG. More details on assigning these probability values follows in section 3.2.

The algorithm below illustrates creating KBAG. In this algorithm, we assume that an attack has been detected which affected a specific object identified as O_{hacked}. A list of hot nodes is determined. A hot node is a node that has high importance value and its corresponding access list is limited to some high privileged users. This indicates that hot nodes contain very sensitive information that should not be accessed except by very high privileged users of the organization. After determining hot nodes, all paths from different nodes in the KBAG to these hot nodes are specified. The paths are possible candidates for attack paths. We assume that attackers try one of these paths to get access to one or more of these hot nodes in their effort to compromise one of them. It is important to mention that hot nodes are not the only nodes that will be investigated. In fact, our model considers identifying attacks that might have been carried out on any node in the underlying system and yet have not been detected. However, identifying hot nodes guides us in building KBAG. It is also right (according to our assumption) that insiders try to compromise several nodes (to accumulate enough knowledge) before attacking a node that has sensitive information. So, although our model use hot nodes to build KBAGs, the reader should realize that all nodes of KBAGs are considered. This means that all nodes of the KBAG will be evaluated and investigated for any expected attack that might have been carried out without being detected by the underlying system. Consider the following segments of a DG which is illustrated in Fig. 5(a) and the segment of a KG of an insider which is illustrated in Fig. 5(b).

Fig. 6 shows the corresponding partially constructed KBAG (yet without implementing the last step, i.e., assigning probability values for the nodes). It shows

```
Algorithm : Creating KBAG {
/* Given that $O_{hacked}$ is the object that has been maliciously modified */
/* KG($S_i$) is the knowledge graph of insider $S_i$ */
/* DG is the dependency graph of objects in the system */
 Create a new graph G* = (V*, E*) where
 V* =          /* V* is empty */
 E* =          /* E* is empty */
 V* = V*  $O_{hacked}$
 V* = V*  $O_{hot}$
 for every insider $S_i$ do
          for each vertex $v_j$  KG($S_i$)
            V* = V* U  $v_j$
 for each vertex vk  belongs to V* {
          Starting with the vertex $v_k$
          Traverse DG and add edges between vertices $v_k$ and $v_j$ ($v_k$, $v_j$)  G* such that
          there is a direct edge connecting $v_k$ and $v_j$ in DG}
 for each vertex $v_i$ belongs to V* do{
          if $v_i$ does not have any incoming or leaving edge from it then{
            $v_i$ is an isolated vertex
            V* = V* - vi } }
 for each vertex $v_i$  belongs to V* do {
          if there is no direct edge between $v_i$ and $O_{hot}$ in G* then do {
          Traverse DG
          if there is a path from $v_i$ to {$v_j$}and a path from {$v_j$} to $O_{hot}$ such that
          /* {$v_j$} represents one or more vertices */
          both vertices $v_i$ and $v_j$ ($v_i$, $v_j$)  G* then add a series of edges
          between $v_i$ and $O_{hot}$ adding edges among all intermediate nodes
          {$v_j$} between $v_i$ and $O_{hot}$ } }
G* is the new KBAG
}
```



(a) An example of a DG. (b) A snapshot of a KG.

Fig. 5 A DG and a KG of a specific insider.

the partial KBAG with nodes (objects) and edges (dependency relationships among nodes). Once the KBAG is constructed, probability values will be assigned to each node. From Fig. 6 we can observe the following facts:

- A KBAG may, in some cases, consist of only one node. This case happens when an attacked object does not have any dependency relationship with any other

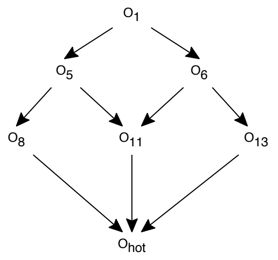


Fig. 6 A partially constructed KBAG.

object in the system. In such a case, the insider can not launch an attack on any other node if he/she relies on dependency relationships exist among objects

- The risk of compromising a set of descendent nodes does not depend on the number of the ongoing edges from the parent node of these descendent nodes. For example, consider O_5, O_8, and O_{11} in Fig. 6. O_5 is the parent node of both O_8 and O_{11}. Consider that the probability of compromising O_5 is high (say 0.75), then, the risk value of compromising O_5 is also high. This high value results in a high risk of compromising all descendent nodes of O_5 (O_8, and O_{11}). The same example can be applied to cases where the probability value of O_5 is low which results in low chances of compromising any of the descendent nodes. So, if the probability of a node is high, then probability of compromising all its descendent nodes is high
- The risk of compromising a descendent node depends on the probability values of compromising all its parent nodes. That is, the risk of compromising a specific node depends on the number of incoming edges to this node. To illustrate, consider O_5, O_6, O_{11}. These objects have the relation $O_{11} = F(O_5, O_6)$. Consider that their probability values are (0.2, 0.3, 0.7), respectively. The risk values can be expressed as (low, low, high), respectively. Then compromising O_{11} is very low. Since compromising O_{11} is low, then compromising O_5 is also low. The fact that compromising O_6 is high does not contribute much in the likelihood of compromising O_{11}. However, it can be deduced that compromising O_{13} is likely to be very high because the probability of compromising O_6 is high and O_{13} is a descendent of O_6 (as explained in the previous case)

From the above cases, one can conclude that what contributes more to the likelihood of compromising a node is the number of incoming edges to the node, not the number of outgoing edges from that node (taking into consideration the probability values of nodes). The following section illustrates the process of assigning these probability values to the corresponding KBAG.

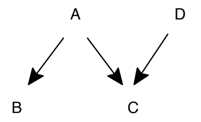
3.1.1 Assigning Probabilities to KBAGs' Nodes

To use Bayesian networks in modeling KBAG, probability values need to be assigned to each node in the KBAG. Our proposed approach uses the fact that an attack which affected node O_{hacked} has been detected. It also aims to make sure that no other nodes in the system have been maliciously modified without being detected. In the proposed approach we follow a similar procedure used by [10] to assign different probability values for all nodes of the corresponding KBAG, where two cases can be classified:

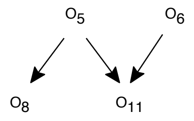
- Base Case: Indicates the probability of only one node and takes the form P(A = yes/no). This shows the probability of an attack of node A. Here, "yes" means that the node has been compromised and "no" indicates that node A is not compromised. An example is: $P(O_1{=}yes) = 0.3$. This probability means that the probability of a successful attack on O_1 equals to 0.3.
- Inductive Case: Indicates the probability of one or more nodes given the knowledge of a fact of one or more other nodes. This probability takes the form: P(A= yes/no / B= yes/no). This indicates a dependency relation among nodes in the KBAGs and is a conditional dependency from the parent node to the child node as indicated in the definition of KBAGs. So, the above probability shows the probability of an attack at node A by a specific insider given that this node has dependency with another node B that may or may not be compromised by the insider. An example is: $P(O_i = no / O_j = yes) = 0.41$. It means that the probability of node O_i is not compromised considering the dependency of its parent node O_j which has been compromised equals to 0.41.
  Note: the two objects O_i and O_j have the parent/child relationship as indicated earlier in the definition of KBAGs.

Therefore, probability by the inductive case (which draws the probability of a node X_i in the KBAG) is given by: $P(X_i) = P(X_i / parent(X_i))$

Expert knowledge and past observations are used to assign the initial probability values to nodes in a KBAG. To clarfiy how initial values are assigned for different nodes, let us consider the following example:



(a) A KBAG of four nodes.



(b) A segment of KBAG given in Fig. 6.

Fig. 7 Two KBAGs of four nodes

The following probability values are associated with nodes of the KBAG of Fig. 7(a):

P(A) = 0.6, P(B) = 0.4, P(C) = 0.2, P(D) = 0.7, P(B = yes / A = yes) = 0.3, P(C = yes / A = yes, D = no) = 0.4

Below we explain what the above probabilities mean:

- P(A)= 0.6 means that 60% of insiders trying to access node A were able to access it. If 100 insiders try to access this node, then on average 60 of them will be able to access it.
- P(B), P(C), and P(D) have the same meaning.
- P(B = yes / A = yes) = 0.3 means that if given that node A had been compromised, then there is a probability of 0.3 for node B to be also compromised. In other words, according to system previous knowledge and experience if node A had been compromised, there is a probability of 0.3 for node B to be also compromised. That is, system experience indicates that 30% of attackers try to compromise node B after compromising node A.
- P(C = yes / A = yes, D = no) = 0.4 means that if given the fact that node A had been compromised and node D had not been compromised, then there is a probability of 0.4 for node C to be compromised. In other words if an attacker wants to compromise node C then he has a choice of following a path from A going to C or following a path from D going to C. However, according to system experience and knowledge 40% of attackers follow the path from A to C.

Based on the above discussion, the system realizes that the attacker most likely follows the path with the higher probability value. In fact, the above fact will be captured in our method of calculating risk values. That is, in the method of calculating risk values, higher probability values contributes more in the new risk values and hence may give more insight for a node to be compromised by an insider. The above discussion gives a clear idea about the way of calculating new risk values and hence a new evidence for a node to be compromised. This method of calculating risk values has been adapted from [10].

3.2 *Calculating Risk Values for Nodes of KBAGs with an Example Scenario*

Risk value for a specific node A in a KBAG can be achieved by calculating a new probability value of that node given the occurrence of an attack at a descendant node that has dependency with it. To illustrate the process of dynamically updating new probability values for nodes based on the occurrence of an attack that affected other nodes in the KBAG, we consider a segment of the KBAG we gave earlier, illustrated in Fig. 7(b):

The example shows how to compute the inferred probability at node O_5 for observed evidence at node O_{11}. ”Observed evidence” means that the object O_{11} had been successfully attacked. In light of previous experience and system knowledge and according to our previous discussion we assume that initial probability values for the nodes of the KBAG are:

P(O_5 = yes) = 0.4, P(O_6 = yes) = 0.1, P(O_{11} = yes) = 0.3, P(O_8 = yes / O_5 = yes) = 0.3, P(O_8 = yes / O_5 = no) = 0.2, P(O_{11} = yes / O_6 = yes, O_5 = no) = 0.3, P(O_{11} = yes / O_6 = yes, O_5 = yes) = 0.15

If an evidence of an attack has been detected at O_{11} by insider S_i, then the probability that an attack at O_5 was carried out can be calculated by computing the probability P(O_5/O_6, O_8, O_{11}); this is computed as:

$$P(O_5/O_6,O_8,O_{11}) = \frac{P(O_5,O_6,O_8,O_{11}}{\sum P(O_5,O_6,O_8,O_{11})} \tag{1}$$

$$= \frac{P(O_{11} = yes/O_6 = yes, O_5 = yes) * P(O_8 = yes/O_5 = yes) * P(O_5 = yes)}{\sum P(O_{11}/O_6,O_5) * P(O_8/O_5) * P(O_5)} \tag{2}$$

= 0.4929

This new probability value of O_5 (represents risk value) is compared with a threshold value. If it is less than the threshold value, it does not indicate a high risk of compromising that node. However, if it is equal to or more than the threshold value, then it indicates a very high risk of an attack at that node which causes an alarm to be raised. Legal actions should be initiated to resolve this situation.

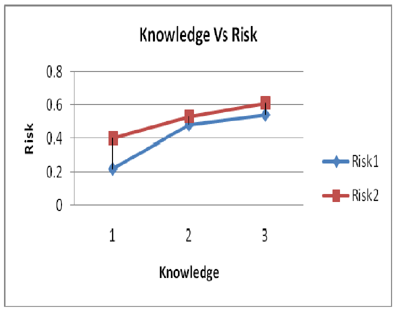
This process of calculating risk values for different nodes in the KBAGs which involves updating probability values is repeated for all nodes. It shows which nodes may be compromised (maliciously modified), and which other nodes do not have any indication of a compromise. The above process of dynamically updating probability values takes care of any risk that may arise from an insider that accesses several documents and tries to access some critical documents. If the risk reaches a sensitive point that is high enough for the insider to compromise a critical object, the above procedure will detect that and raise an alarm. Raising an alarm informs legal parties about the situation. Hence, legal actions should be initiated to resolve the problem. The following table 2 describes different risk values for nodes of the KBAG described in Fig. 7(b), given that O_{11} has been attacked (maliciously modified). Risk values have been calculated using the above procedure. As can be seen from table 2, risk values might increase having the fact that other nodes might have been compromised. From table 2, we conclude that increased knowledge of a malicious insider increases his/her chances of compromising more objects in the underlying system. Hence, a direct relation exists between the knowledge of a malicious insider and the risk of compromising more objects. This can also be captured by Fig. 8(b) which illustrates results of the table in Fig. 8(a). In Fig. 8, risk1 represents risk values of O_6 considering the fact that one, two or three ascendant or sibling nodes of O_6 have been compromised. For example, risk value of O_6 considering that one of the ascendant nodes (O_6 and its parent O_1 results in 2 nodes) equals

Table 2 Calculated risk values of nodes of KBAG given in Fig. 7(b)

Risk	P(O_5)	P(O_5/O_{15})	P(O_6/O_{15})	P(O_8/O_1)	P(O_6/O_1)	P(O_5/O_{15},O_6,O_1)	P(O_6/O_{15},O_5,O_1)
Value	0.4	0.55	0.49	0.53	0.48	0.61	0.54

Knowledge	Risk1(for O_6)	Risk2 (for O_5)
1	0.22	0.40
2	0.48	0.53
3	0.54	0.61

(a)Knowledge vs Risk relation



(b) Knowledge vs Risk

Fig. 8 Knowledge VS Risk

0.48. The same can be applied to risk2 to get the same meaning of risk1. Knowledge indicates that the insider gets the knowledge of the corresponding node(s). For example two in the knowledge column of Fig. 8(a) indicates that the insider gets the knowledge of two objects (O_6 and O_1 as an example)

4 Conclusion

In this paper, we introduced a new model of evaluating and analyzing a system after an insider attack. In this work, we introduced a new graph called Knowledge Bayesian Attack Graph (KBAG) that uses Bayesian network concepts as well as knowledge graphs (KGs) and dependency graph (DG) of different objects in the underlying system. KBAG helps in reasoning about attacks on the underlying system because its paths are considered as candidate paths that (according to our assumptions) malicious insiders may follow to achieve their goals of compromising critical objects. It also helps in calculating risk values for each node in the KBAG. The model should calculate and update risk values regularly using Bayesian inference techniques. Regularly updating risk values reflects the likelihood of the possible occurrence of an attack on other nodes by the same insider. Risk values also help in classifying objects into a list of non affected objects and a list of possible affected objects. Consequently, an increased risk of a node in the KBAG indicates a possible compromise of that node by the insider. It is important to mention that this work is suitable only for insider attacks. So, the reader should realize that attempts to use this model to outsider attacks may not be successful. The compromises that are suspected by our model, that may impose a great risk on the system, represent compromises that are not detected by the underlying system. So, our model helps in uncovering these compromises and also helps in understanding the ways that malicious insiders may use to launch their attacks.

Acknowledgements This work has been supported in part by US AFOSR under grant FA 9550-04-1-0429. We are thankful to Dr. Robert. L. Herklotz for his support, which made this work possible. We are also thankful to the anonymous reviewers for their valuable remarks.

References

1. D. Burroughs, L. Wilson and G. Cybenko. Analysis of Distributed Intrusion Detection System Using Bayesian Methods. In Proceedings of the twenty first IEEE International Performance, Computing and Communications Conference, 2002.
2. Q. Althebyan, B. Panda. A Knowledge-Base Model for Insider Threat Prediction. In Proceedings of the 2007 IEEE Workshop on Information Assurance United States Military Academy, West Point, NY 20-22 June 2007.
3. M. Maybury, P. Chase, B. Cheikes, D. Brackne, S. Matzner, T. Hetherington, B. Wood, C. Sibley, J. Marin, T. Longstaff, L. Spitzner, J. Haile, J. Copeland, S. Lewandowski. Analysis and Detection of Malicious Insiders. In Proceedings of the 2005 International Conference on Intelligence Analysis. Sheraton Premiere, McLean, VA. May 2-4.
4. B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, A. Sheth. An Ontological Approach to the Document Access Problem of Insider Threat. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics, ISI 2005, Atlanta, Georgia, USA, May 19-20, 2005, 486-491.
5. J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, San Mateo, CA, 1988.
6. L Larsen, M. Harrold. Slicing Object-Oriented Software. In Proceedings of the 18th International Conference in Software Engineering, March 1996.
7. L.P. Swiler, C. Philips, D. Ellis, S. Chakerian. Computer-Attack Graph Generation Tool. In Proceedings of the IEEE Symposium on Security and Privacy 2001.
8. A.P. Moore, R.J. Ellison, R.C. Linger. Attack Modeling for Information Security and Survivability. Technical Note, CMU/SE1-2001-TN-001, March 2001.
9. O. Sheyner, J. Joshua Haines, S. Jha, R. Lippmann, J.M. Wing. Automated Generation and Analysis of Attack Graphs. In Proceedings of the IEEE Symposium on Security and Privacy, 2002.
10. R. Dantu, P. Kolan. Risk Management Using Based Behavior Bayesian Networks. In Proceedings of IEEE International Conference on Intelligence and Security Informatics, ISI'2005, Lecture Notes in Computer Science (LNCS), pages 115-126, Springer-Verlag, May, 2005.