Knowledge Extraction and Management for Insider Threat Mitigation

Qutaibah Althebyan and Brajendra Panda

Computer Science and Computer Engineering Department
University of Arkansas, Fayetteville, AR 72701, U.S.A.

Abstract. This paper presents a model for insider threat mitigation. While many of the existing insider threat models concentrate on watching insiders' activities for any misbehavior, we believe that considering the insider himself/herself as a basic entity before looking into his/her activities will be more effective. In this paper, we presented an approach that relies on ontology to extract knowledge from an object. This represents expected knowledge that an insider might gain by accessing that object. We then utilized this information to build a model for insider threat mitigation which ensures that only knowledge units that are related to the insider's domain of access or his/her assigned tasks will be allowed to be accessed by such insiders.

1 Introduction

Security issues are increasingly becoming cumbersome for both individuals and organizations. Individuals want to ensure that their private data remains unexposed to others. At the same time, organizations aim to increase their productivity by trusting their employees while maintaining confidentiality, integrity, and/or availability [1] of information. However, no organization can fully trust its employees because some employees, having malicious intentions, might be waiting for opportunities to access information violating the organization's security policy. Hence, the insider threat problem [1, 2, 3, 4, 5] is becoming one of the most severe problems that needs to be addressed and resolved in an effective manner. In our work, we define an insider as a person with legitimate access to the underlying system. In an organization's effort to stop insiders' attacks, it must ensure that an insider accesses only documents that are relevant to his/her domain of access and his/her assigned tasks [6]. Moreover, the nature of insiders, who are privileged to access many resources of the organization, enables them to get access to many important assets of the organization. Through the broad familiarity with their organization besides their privileges, insiders can accumulate knowledge of many assets of their organization. This knowledge gives insiders, with malicious intentions, opportunities to obtain sensitive information about organizations' assets by either accessing sensitive assets or by predicting information about sensitive objects through their knowledge of related objects. Hence, an organization is responsible for making sure that certain insiders' knowledge does not increase to a point by which they can get access to organization's sensitive data.

Althebyan Q. and Panda B. (2008).
Knowledge Extraction and Management for Insider Threat Mitigation.
In *Proceedings of the 6th International Workshop on Security in Information Systems*, pages 99-110
DOI: 10.5220/0001741800990110

In this paper, we address the problem of insider threats and present a model that enables any organization control its insiders' activities and stop them from accessing objects that they are not allowed to access. In fact, we divided our work into two parts. Section 2 deals with extracting knowledge from objects that an insider requests access to. We believe that to solve the problem of the insider threat, we need to consider the insider as our basic entity that should be dealt with before looking into his/her activities. In our effort to extract knowledge from an object (especially documents) we followed an ontological approach that enabled us to extract knowledge which will be saved as knowledge units from a given object. Then, in section 4 we use insider's accumulated knowledge besides other information to build a model to mitigate insider threats. In our model we make sure that individuals of an organization do not get access to any object unless it is relevant to their domain of access. In fact, we deal with individuals who try to obtain information from organization's other domains which might help them to gain access or uncover information about sensitive data of their own organization.

2 Extracting Knowledge from Objects using Ontology

We assume that the underlying system has different domains that organization's individuals are allowed to get access to. An insider might be allowed to gain access to more than one domain if his/her assigned tasks need that. However, he/she is not allowed to simultaneously access more than one account that belongs to only one domain.

Domains in the underlying system are categorized and named according to their specialization. For example, a domain that is related to finance is called finance-domain, a domain that is related to employees and their information might be named as human-resource-domain and so on. So, the domain that an insider is accessing can be specified and determined. Usually an insider has privileges through which he/she can access different objects in his/her organization. Throughout this paper, we use 'object' and 'document' to mean the same. So, both words are interchangeably used in this paper. An insider upon accessing an object increases his/her knowledge. We assume that any knowledge an insider gains is saved in his/her knowledgebase. For example, consider an insider who got access to an important document that has sensitive information. Then this person ascertained different facts from this document. These facts are added to the knowledgebase of this individual and are saved in his/her knowledgebase as different knowledge units. This section aims to extract knowledge units from objects an insider has access to. In our paper we follow an ontological approach to extract such knowledge units.

For each domain of access a predefined ontology of topics is determined in advance. That is, for every object an insider accesses, the domain of access is identified. For each domain a set of ontology of topics are predefined. These topics are terms that are related to the content of the corresponding object. We are interested in computing the relevance of the content of a document to the domain of access. Hence, the relevance of these related ontological terms to the content of the corresponding objects is computed. We assume that an insider upon accessing an object, which belongs to a specific domain, might be interested in information that is

related to that domain. For example, an insider accessing a document that belongs to a finance domain, might be interested in information related to that domain, namely, salaries of employees (other examples are valid).

2.1 Document Categorization

According to our model specifications, we distinguish between two kinds of documents (although other kinds of documents might exist):

1. Documents in which their contents are stored as tables
2. Documents in which their contents are stored as plain text

The first type of documents is easy to extract knowledge units from. Information in the tables can be distinguished by a row/column entry. That is, the first row/column entry has a named entry. Hence, each entry represents a knowledge unit. Knowledge units can be combined to give higher knowledge units. For example, consider a document *d* has the following content that contains a single table (employee-salary table) of the form as shown in table 1:

Table 1. Employee-salary table.

Name	ID	Salary
A	1	$1000
B	2	$2000
C	3	$3000

The above table contains a set of (Name, ID, Salary) of three employees in the organization. Hence, at least three knowledge units can be extracted from the above table.

$K_1 \rightarrow$ Name(Employee) $K_2 \rightarrow$ ID(Employee) $K_3 \rightarrow$ Salary(Employee)

Higher knowledge units can also be extracted by combining existing knowledge units [7]. Examples of combining knowledge units are:

$K_{12} \rightarrow$ Name-ID(Employee) $K_{123} \rightarrow$ Name-ID-Salary(Employee)

The second type of documents needs extra processing to extract different knowledge units. In fact we use a measure that quantifies the relevance of the content of a document to the domain of access, as explained in the next section.

2.2 Relevance Factor

The relevance factor is a measurement that quantifies how much the content of an object (especially documents) is related to the domain of access for a specific insider. This means that the relevance factor of an object measures how much the several facts of the object are related to the domain of access of the corresponding insider. To calculate the relevance factor of the object, we followed a procedure presented in [8] and borrowed some of their ideas which involve computing the relevance of a

document in performing a focused crawling in the web. We adapted some of their ideas to fit in our work to calculate the relevance of the content of an object to the domain of access of the corresponding insider. Relevance of the content of an object *f* to the domain of access *D* will be computed as:

$$R(f) = \sum P(c|f) \tag{1}$$

where:
R(f) represents the relevance of the content of document *f* to the domain of access

$P(c|f)$ represents probability that object *f* mentioned topic *c*

In equation (1) the probability $P(c|f)$ is computed as:

$$P(c|f) = P(parent(c)|f)P\big(c\big|f, parent(c)\big) \tag{2}$$

Using Bayes rule and ontology the last conditional probability can be computed as:

$$P\big(c\big|f, parent(c)\big) = \frac{P\big(c\big|parent(c)\big)P(f|c)}{\sum_{c',parent(c')=parent(c)} P(f|c')} \tag{3}$$

where the sum ranges over the siblings *c'* of *c*. Finally $P(f|c)$ in equation (3) can be computed using a Bernoulli binomial distribution model:

$$P(f|c) = \binom{n(f)}{n(f,t)} \prod_{t \in f} \theta(c,t)^{n(f,t)} \tag{4}$$

$$P(f|c) = \prod_{t \in f} \left[\frac{n(f)!}{n(f,t)! * \big(n(f) - n(f,t)\big)!}\right] * \theta(c,t)^{n(f,t)} \tag{5}$$

where:
t: represents terms(words) in the document *f*,
n(f): represents the number of words in *f*,
n(f, t): represents the number of times the *t* appears in *f*,
$\theta(c,t)$: represents the probability of *t* in topic *c*.

For the probability $P(f|c)$ in (4) we used the Bernoulli binomial distribution which fits the specifications of our problem and works well in our model.

Consider the following example to verify our procedure of extracting knowledge units. An example of existing ontology has been used in this case. Three documents are created in plain text. That is, they do not have named entities in tables. All documents belong to the same domain, which is the domain of the finance department as indicated in table 2 and are accessed by the same insider. We apply the above procedure to extract different knowledge units from these documents.

Table 2. Number of words and domain of access of three documents.

	Doc1 (project_total_cost)	Doc2 (employee_salary)	Doc3 (hourly_rates)
Domain	Finance Department	Finance Department	Finance Department
Words	53	200	79

Table 2 contains information about three documents (project_total_cost, employee_salary, hourly_rates) for different employees in an organization. As discussed earlier, we are going to extract different knowledge units from different documents that belong to a specific domain for a given insider. Our procedure uses an existing ontology that describes the given domain of access. So, a set of related terms (called topics in our work) which pertains to the underlying domain of access can be obtained from this ontology. Using our discussed approach, relevance values are computed using the above equations (1) - (5). These relevance values represent values that measure the relevance of the content of each document to the domain of access. The following tables (table 3, table 4, and table 5) show several calculated relevance values for different ontological topics for documents of table 2.

Table 3 shows that two topics (Name and cost) of a project are found to be relevant and one topic is found to be not relevant. Hence, these two relevant topics are represented as two new knowledge units and are saved in the corresponding insider's knowledgebase as new knowledge units. A possible representation might be as:

$K_1 \rightarrow$ Project_Name (for Name attribute) $K_2 \rightarrow$ Cost

Table 4 shows that all the three ontological topics are found to be relevant and hence, are saved in the insider's knowledgebase as new knowledge units. They might be represented as:

$K_3 \rightarrow$ Employee_Name(for Name attribute) $K_4 \rightarrow$ Employee_ID (for ID attribute)
$K_5 \rightarrow$ Salary

Table 5 shows that two of the ontological topics are found to be relevant and hence, are represented as new knowledge units. They might be represented as:

$K_6 \rightarrow$ Employee_Rank (for Rank attribute) $K_7 \rightarrow$ Hourly_Rates (for Rates attribute)

Table 3. Doc 1.

Topics	*R(f)*: Doc1
Name	0.43
Cost	0.12
Salary	0.0

Table 4. Doc 2.

Topics	*R(f)*: Doc2
Name	0.71
ID	0.71
Salary	0.3

Table 5. Doc 3.

Topics	*R(f)*: Doc3
Rank	0.1
Rates	0.48
Category	0.0

The above new knowledge units are added as new nodes of a knowledge graph (KG) [7] for the corresponding insider. For example, if we consider only one knowledge unit from the above tables namely (K_1, K_5, K_7) and represent them in a knowledge graph KG of the corresponding insider then this KG will look as depicted in fig. 1. In fig. 1, O_1 corresponds to document 1, O_2 corresponds to document 2, and O_3 corresponds to document 3 in our example. All other knowledge units will also be added to the insider's KG. Readers who are interested in more details about knowledge graphs and how to build these graphs, may refer to [7] for more details.

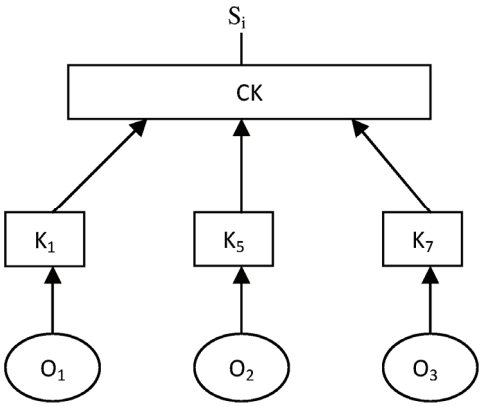

Fig. 1. An Example of a Knowledge Graph.

3 Dependency Graphs and Ontology

In this section we use concepts of ontology [9, 10, 11] discussed in the previous section and dependency graphs (DGs) [7] to extract knowledge units from descendent documents given that a set of knowledge units in the ancestor document has been specified. To provide familiarity with the concept of dependency graph, we give a brief description of dependency graphs next.

3.1 Dependency Graphs (DGs)

A *Dependency graph* (DG) "is a global hierarchal graph that shows all dependencies among various objects in the system" [7].

Usually objects, especially documents, are created depending on other objects in the same system. Such a dependency relationship is an important relationship through which insiders may be able to locate important objects in the system. 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 [12] for modeling an object-oriented program.

Fig. 2 represents an example of a DG of objects. The DG is a dynamic graph that is updated for every newly created document in the system. A new edge is added between the new document and the document (or documents) for which the new document relies on. Also, the DG will be updated periodically to reflect operations affecting the documents. A DG is a one directional graph in which the direction of the edge indicates the dependency direction which can be verified from fig. 2. Throughout this paper, we follow the top down approach for representing dependencies among different objects in the DG. So, if we consider the dependency relation between any two objects as a function F, then we can verify the following relation $O_1 = F(O_2, O_3)$ from fig. 2 to be true.

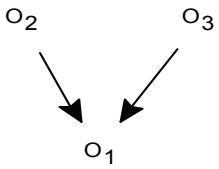

Fig. 2. An example of a Dependency Graph of objects.

It is important to clarify the meaning of several dependency relations (represented by directed edges) that might exist between nodes of a DG. For example, in the above example, the relation $O_1 = F(O_2, O_3)$ indicates that object O_1 is a function of two objects O_2 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_2 and O_3 has been used to derive information in O_1. This information contributes to some or all knowledge units that O_1 contains.

3.2 Dependency Graphs and Ontology

A dependency relationship between any two objects (documents in this discussion) as discussed earlier is an important relation through which knowledge units are specified. An example of such important relation between two objects, which extends our previous example in section 2.3 with some modifications, is the following:

Object D_1 contains salaries of employees computed by considering their hourly rates and the number of hours they worked. That is: $D_1 \rightarrow$ salary(hourly_rate, hours_worked).
Information in object D_1 can be represented by the following table:

Table 6. Information in object D_1.

Employee ID	Salary	Hourly_rate	Hours_worked

Also consider that object D_2 contains hourly rates of all employees, that is: $D_2 \rightarrow$ hourly_rate, and object D_3 contains number of hours each employee worked, that is: $D_3 \rightarrow$ hours_worked.

The above three objects D_1, D_2, and D_3 have the following dependency relation: $D_1 = F(D_2, D_3)$. Fig. 2 illustrates this relation where (O_1, O_2, O_3) are represented by (D_1, D_2, D_3) in our current example.

Assume that in the above example knowledge units of object D_1 have been specified using our previous discussion for extracting knowledge units of an object discussed in section 2.3.Among the extracted knowledge units are U_i, U_j, and U_l which are specified as follows:

$U_i \rightarrow$ salary $U_j \rightarrow$ hourly_rates $U_l \rightarrow$ hours_worked

where U_i, U_j, and U_l represent different knowledge units for different employees under consideration in object D_1. That is, U_i (as an example) represents a set of

knowledge units for salaries of different employees under consideration. Knowledge units U_j, and U_l can be interpreted in a similar way.

As stated earlier there is a dependency relation between D_1 and D_2. Another dependency relation from D_1 to D_3 also exists. Different sets of knowledge units of D_1 (U_i, U_j, U_l) have been specified. The above two dependency relations and the sets of knowledge units can be used to specify part of the knowledge units of descendent objects D_2 and D_3. Hence, in the previous example a new set of knowledge units U_m among others in D_2 can be specified and they have the form $U_m \rightarrow$ hourly_rates. Also, a new knowledge unit U_r in D_3 can be specified as $U_r \rightarrow$ worked_hours. The above example is a brief illustration of the importance of dependency graphs in extracting knowledge units from different related objects.

Propagation of Knowledge Units using DGs. In the following, we discuss in more details using DGs and ontology to extract knowledge units from an object given that this object depends on another object. Fig. 3 shows two objects D_i and D_j. Knowledge units in D_i have been specified based on a specific predefined ontology as K_1, K_2, K_3, and K_{12}. Both objects D_i and D_j have a dependency relation represented by $D_j = F(D_i)$. This dependency relation can be captured from the DG of the underlying system and is illustrated in fig. 3. Since D_j is dependent on D_i there should be at least one knowledge unit that is common in both documents. Using this dependency relation between the two objects and a common ontology as illustrated in fig. 3 a knowledge unit say K_1 is a common knowledge unit (attribute) in both objects. Since K_1 is specified as knowledge unit in D_i, it will also be specified as a new knowledge unit in D_j.

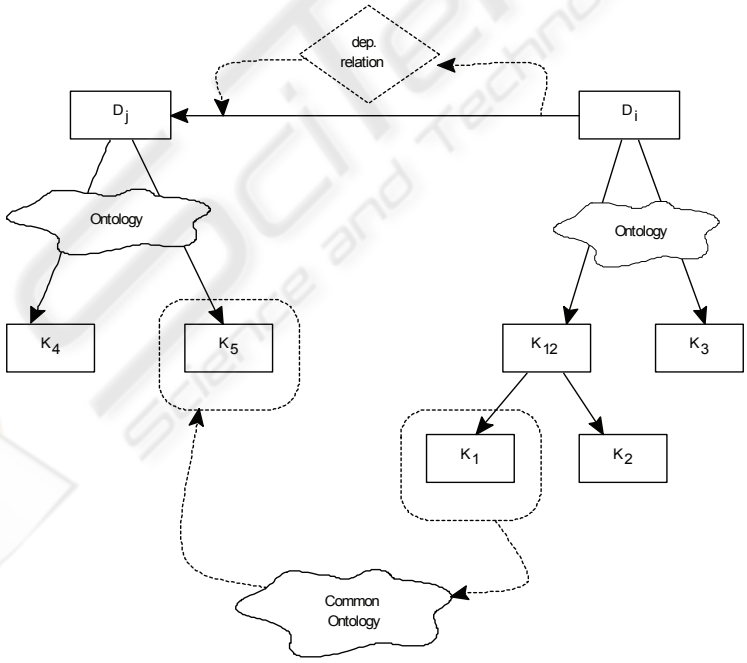

Fig. 3. Overview of knowledge units' extraction using DGs and Ontology.

This fact is shown in fig. 3 and the common knowledge unit K1 in Di has been specified as a new knowledge unit in Dj as K5. More precisely, the relevance of each knowledge unit (in object Di) to object Dj will be computed using our previous procedure in section 2.3. If any knowledge unit of Di is found to be relevant to Dj then it will be specified as a new knowledge unit in Dj. Since the two objects use different ontologies, a different name (K5) for the new knowledge unit has been specified.

4 Managing Insiders' Accesses

We presented an ontological approach to extract knowledge units from an object that belongs to a specific domain of a given insider. In our approach we calculated relevance values for several facts (called topics). These topics, if found relevant to the domain of access, are considered new knowledge units for the given user if he/she has not accessed them before. These knowledge units are saved for future use and represented by knowledge graphs (KGs). Hence, each knowledge unit is associated with a specific relevance value. That is: $K_1 \rightarrow R_1(f)$, $K_2 \rightarrow R_2(f)$, …, $K_n \rightarrow R_n(f)$.
Since we assume that several domains exist, then some objects (especially documents) might belong to different domains. These objects might be requested by insiders from different domains. When such an object is requested by an insider for the first time, knowledge units relevant to that domain are extracted. Subsequent requests will use existing knowledge units. An object as a whole might contain knowledge units that are relevant to multiple domains. Knowledge units that are not relevant to the requesting insider domain should not be disclosed to such insiders unless they are not sensitive. So, the system must detect such situations and be able to stop them. In our model, every domain has a specific threshold value. Knowledge units of an object having relevance factor more than the threshold value are considered sensitive and should not be provided to users who belong to different domains.

These several threshold values are used in sensitivity check which says:
Sensitivity_Check: For a specific domain of access D and a knowledge unit K_i: if relelvance(K_i) > domain_threshold(D), then K_i is sensitive. Otherwise, it is not sensitive.

In our model, employees of an organization can access objects in their domain. Objects that do not belong to an insider's domain should not be accessed by such person except for certain cases that will be discussed shortly. If insiders are allowed to freely access documents outside of their domain then this represents a violation that might be devastating to the system. An approach that detects similar situations and stops insiders from accessing information that they should not access is developed and provided next.
Our model defines the following access policy:
When an insider requests access, access is only granted if the object:

1. Is in the same domain of the requesting insider, or
2. It does not belong to the requesting insider's domain, but has knowledge units that are relevant to the domain of access of that requester, or

3. It does not belong to the requesting insider domain and does not contain knowledge units that are relevant to his/her domain, but contains knowledge units that are not sensitive.

To clarify things suppose that there exists an object O that belongs to domain A. Object O contains knowledge units K_1, K_2, K_3, and K_4 with relevance values as: 0.6, 0.4, 0.8, and 0.2 respectively. These knowledge units are relevant to different domains, namely; K_1 and K_2 are relevant to domain B while K_3 and K_4 are relevant to domain C. Domain A threshold value equals to 0.7, while domain B threshold value equals to 0.65, and domain C threshold value equals to 0.75. Suppose that two insiders $S_1 \in$ dom(B) and $S_2 \in$ dom(C) try to access object O. Since object O belongs to neither domain B nor domain C, then their requests might be denied unless it contains knowledge units relevant to their domains or they satisfy the sensitivity check. The following cases can be distinguished:

- Insider S_1 belongs to domain B. Both K_1 and K_2 are relevant to domain B. Neither K_3 nor K_4 are relevant to domain B (they are relevant to domain C)
- Insider S_2 belongs to domain C. Both K_3 and K_4 are relevant to domain C. Neither K_1 nor K_2 are relevant to domain C (they are relevant to domain B).

Since knowledge units K_1 and K_2 are relevant to domain B (which insider S_1 belongs to) then both K_1 and K_2 are accessible by S_1. However, since K_3 and K_4 are not relevant to his/her domain then they might/might not be accessible. After applying the sensitivity check: relevance(K_3) = 0.8 which is greater than domain C threshold value = 0.75. That is, K_3 is considered sensitive and hence, K_3 will not be accessible. Also, relevance(K_4) = 0.2 which is less than 0.75. Hence, K_4 is accessible because it does not contain enough sensitive information related to domain C that must not be revealed to insider S_1. Thus, our model compares relevance of a knowledge unit with the threshold value of the domain it is relevant to. The reason is to make sure that this knowledge unit does not contain enough sensitive information that should be kept hidden from insiders of other domains. That is, the knowledge an insider can gain from this knowledge unit is less than the sensitivity threshold of that domain. The same procedure is applied to requests of S_2. It is found that knowledge units K_3 and K_4 are accessible by S_2 because they are relevant to his/her domain. However, for K_1 and K_2 the sensitivity check is applied. Relevance(K_1) = 0.6 which is less than domain B threshold value. Also relevance(K_2) is less than domain B threshold value. Therefore, both K_1 and K_2 are accessible by S_2 besides K_3 and K_4. That is, S_2 can access the entire object O.

From the above example, it can be concluded that insider S_2 can access the whole object O. However, insider S_1 gets partial access to the object O. In fact, he will get access to knowledge units K_1, K_2, and K_4. Since K_3 has sensitive information that he should not access, a filtration process will be initiated which filters the object O out by removing knowledge unit K_3 from that object. The remaining content of object O is then presented to insider S_1.

Based on the above example, there are three cases to consider: objects that contain knowledge units that are only relevant to the domain of access of the requesting individual, objects with knowledge units some of which are relevant to the domain of access and some others are not relevant, and objects with knowledge units all of which are not relevant to the domain of access of the requesting individual.

For an outstanding request to access an object O by a specific insider S_i the following algorithm is used:

```
Algorithm: Manage_Insider_Request{
    If O ∈ domain D of Si then
        Grant Access to object O
    Else
        For every knowledge unit Ki ∈ O
                If Ki is relevant to domain D then
                        Ki is accessible
                Else    //apply the sensitivity check
                        If    relevant(Ki)    <    domain_threshold    then
                        //threshold  for related domain of Ki
                                Ki is accessible
                        Else
                                Ki is not accessible
        If Ki is not accessible then
                Filtration(object   O)//   a   function   that   removes//
                sensitive Ki's from the object O.
    Return object O to Si after Filtration()
}
```

The following actions are performed for the above categorization of objects: after applying the above algorithm to the first type of objects, requests to these objects are allowed. For second type of objects: knowledge units found to be non sensitive are accessible by such insiders while knowledge units found to be sensitive are not accessible. Hence, the filtration operation deletes these sensitive knowledge units from the requested object. So, the insider receives an object with only partial content of the original object. For the third type of objects: knowledge units found to be non sensitive are accessible by such insiders while knowledge units that are sensitive are not accessible. Hence, the filtration operation deletes these sensitive knowledge units from the requested object. The filtration operation deletes content of knowledge units that should not be revealed to the underlying insider. The above procedure ensures that only relevant information to the insiders' assigned tasks are revealed besides information that is not sensitive to his/her domain. This limits malicious activities that might be initiated by insiders with malicious intensions.

5 Conclusions

This paper studies the problem of insider threats. In our paper we presented an ontological approach to extract knowledge units from a given object. After extracting knowledge units we presented a procedure which used these knowledge units to control insiders' knowledge. We have the assumption that increased knowledge of insiders with malicious intension gives them more opportunities to fulfill their motives. We then presented an insider threat mitigation model which ensures that insiders, with knowledge of their organization, access only objects that are related to their domain and assigned tasks. According to our model, knowledge units of a given object are either sensitive or non sensitive. Our model ensures that insiders who request objects outside their domain do not get access to objects with sensitive information. In fact, our model utilized a filtration process which filters out those sensitive knowledge units from an object and presents the filtered object without the sensitive information to the requesting insider.

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. 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.
2. R. Chinchani, A. Iyer, H. Ngo, S. Upadhayaya. Towards a theory of insider threat assessment. Proceedings of the 2005 International Conference of the Dependable Systems and Networks (DSN 2005). Yokohama, Japan, June 28 – July 01, 2005.
3. N. Nguyen, P. Reiher, G. Kuenning. Detecting Insider Threats by Monitoring System Call Activity. In Proceedings of the 2003 IEEE Workshop on Information Assurance. United States Military Academy, West Point, NY.
4. E. Schultz. A framework for understanding and predicting insider attacks. Computers & Security, Vol. 21, p. 526-531, 2002.
5. I. Ray and N. Poolsappasit. Using Attack Trees to Identify Malicious Attacks from Authorized Insiders. In the Proceedings of the Tenth European Symposium on Research in Computer Security, Milan, Italy, September 2005.
6. 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.
7. 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.
8. S. Chakrabarty, M. V. D. Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. Computer Networks, 31(11-16), pp 1623-1640, 1999.
9. S. Symonenko, et al. Semantic Analysis for Monitoring Insider Threats. IEEE Intelligence and Security Informatics (ISI), 2004.
10. M. K. Henry, S. Arijit, M. Fox and M. Dalkilic. A Measurement Ontology Generalizable for Emerging Domain Applications. Journal of Database Management (JDM) 18:1, Jan-Mar 2007, pp. 20-42.
11. M. K. Henry, S. Arijit. Extracting Knowledge from XML Document Repository: A Semantic Web-Based Approach. Journal of Information Technology and Management, Feb 2007.
12. L. Larsen, M. Harrold. Slicing Object-Oriented Software. In Proceedings of the 18th International Conference in Software Engineering, March 1996.