Annamma Monish and Selvarani, J Mass Communicat Journalism 2014, 4:8
DOI: 10.4172/2165-7912.1000212

Journal of
Mass Communication & Journalism

Research Article Open Access

Mitigating Insider Threat and Avoiding Unauthorized Knowledge Acquirement Using Acquaintance Based Threat Prediction Graph

Annamma Monisha I* and Grace Selvarani

Department Of Computer Science and Engineering (Pg), Sri Ramakrishna Engineering College, Anna University, Chennai, India

Abstract

An Insider Threat is a malicious threat to an organization it actually comes from people within the organization, such as employees, former employees, contractors or business associates, who have access to the confidential information of the organization. The paper characterizes various types of dependencies as well as constraints on dependencies that may be used by insiders to deduce unauthorized information. It pioneers the constraint and dependency graph (CDG) that characterizes dependencies and constraints. Additionally, CDG shows the paths that insiders can track to acquire unauthorized knowledge. In addition, the paper presents the acquaintance graph (AG) that reveals the knowledgebase of an insider and the amount of information that the insider has about data items. To forecast and prevent insider threat, the paper characterizes and uses the threat prediction graph (TPG). A TPG illustrates the threat prediction value (TPV) of each data item in insiders' AG, where TPV is used to lift up an alert when an insider threat occurs.

Keywords: Constraints; Dependencies; Threat prediction graph; Acquaintance graph; Knowledge base

Introduction

Security issues are getting more and more critical with the continual use of computers and communication systems. Since data are a vital asset for both individuals and organizations, mechanisms that defend data from interception, modification and invention in such systems have become very serious. One of the major concerns in computer security is the insider threat difficulty. Insider threat is defined as the threat that is reasoned by a malicious insider who has authorized right to use privileges and knowledge of the computer systems of an organization and is encouraged to antagonistically control the organization [1]. Insider threat problem is as important as the problem of outsiders' intimidation (hackers) due to the excessive harm that it may pose.

According to the Computer Crime and Security Survey, insider attacks accounted for 33% of the total incidents reported in 2010 (C. S. Institute, 2010).

Many mechanisms have been planned for protecting data from outside attacks. However, those mechanisms do not guard data from authorized users who may mishandle their privileges to breach systems security. Thus, developing mechanisms that protect receptive data from insiders has become a key demand due to the amount of harm that can be caused by those spiteful insiders.

Insufficient research has been performed on insider threat in relational databases, such as the work in [2-4]. This paper confers insider threat problem in relational databases. It defines more than a few types of dependencies and constraints that may be used by insiders to get unlawful information. To symbolize dependencies and constraints, the constraint and dependency graph (CDG) is provided. Then, the paper presents a graph-based approach to predict and put off insider threat by using the threat prediction graph (TPG).

The rest of the paper is organized as follows. Related work presents some earlier work which is discussed under heading 4. Types of dependencies are discussed under heading 5. Heading 6 makes obvious the types of constraints on dependencies. Heading 7 Insider Threat: unauthorized knowledge acquisition introduces the problem of gathering unauthorized information by insiders as well as the proposed solutions. Heading 8 presents the conclusions and future work.

Related Work

Insider threat has turned out to be an important security issue due to the tremendous harm [5,6]. Different researchers introduced diverse definitions for insiders at system level, such as [1,7], whereas others defined the insider according to different classes [8]. However, Yaseen and Panda [8-10] defined the insider at the relational databases level, which is the framework of this paper. They defined the insider as a person who has right to use privileges, is familiar with dependencies and their constraints and is familiar with the system under contemplation.

Researchers used existing methods of detecting external threat, such as using honeypots [11], to sense insider threat. However, these methods are not efficient since insiders and outsiders use diverse paths or advancements to attack systems. In other terms, insiders use their right to use privileges and knowledge about systems to harass sensitive data items using paths that are hard to be detected by security mechanisms, while outsiders use arbitrary paths that lead to perceptive and insensitive data items, where these paths can be detected by well-arranged security mechanisms. Other researchers such as [7] initiated new methods to deal with this problem. They introduced a new acquaintance base approach to notice and prevent insider threat. However, the aforesaid research was at the system level and did not consider relational databases, where insiders have more means such as the knowledge about dependencies and constraints that facilitate start on attacks.

***Corresponding author:** Annamma Monisha, Department of computer science and engineering (PG), Sri Ramakrishna engineering college, Anna University, Chennai, India, Tel: 0422 246 1588; E-mail: monishaimmans@gmail.com

Received February 06, 2014; **Accepted** July 31, 2014; **Published** August 07, 2014

Citation: Annamma Monisha I, Selvarani G (2014) Mitigating Insider Threat and Avoiding Unauthorized Knowledge Acquirement Using Acquaintance Based Threat Prediction Graph. J Mass Communicat Journalism 4: 212 doi:10.4172/2165-7912.1000212

Jabbour and Menasce [12,13] proposed a self-protection mechanism, which is the Insider Threat Security Architecture (ITSA), that is totally incorporated into the computing system. However, their self-protection mechanism has scalability problems that may mortify the performance of hosting systems and influence other services. Moreover, their approach needs to be tested and estimated to ensure its effectiveness and applicability in diverse environments.

Dependencies in relational databases play a most important role in insider threat problem. Dependencies as well as the inference problem have been conversed extensively in [14]. Farkas, and Jajodia [14] and Farkas et al. [14] discussed how users can get receptive data using non sensitive data to which they have no right to use. Yaseen and Panda [8-10] take in hand the insider threat in relational databases and developed the Neural Dependency and Inference Graph NDIG, which shows the dependencies among data items and the amount of information that can be inferred about data items using dependencies. However, they did not offer mechanisms to spot and prevent insider threat. In the next section, we will demonstrate new group of dependencies that help more in understanding how insiders begin attacks, and determine which data items are targeted.

In addition to dependencies, the knowledgebase of insiders could be a stern source for insider threat. Furthermore, the lifetimes of data items in knowledgebase can be used to predict insider threat. Althebyan and Panda [7] presented the knowledge graph of an insider at the system level, but they did not mull over relational databases, which has different properties to be represented in knowledge graphs. Farkas et al. [14] introduced the effect of updates on the knowledgebase of users. Their work explained that looking through the history of accesses of users without checking the lifetimes of the data items could limit the availability of data items. Nevertheless, they considered that updating a data item makes it terminate, which is not always correct as we will show in this paper. Yaseen and Panda [8-10] discussed the outcome of knowledgebase of insiders and the lifetimes of data items on insider threat in relational databases.

Types of Dependencies

Two data items, A and B, have a dependency relationship if at slight the value of one of them depends on the value of the other. A dependency between A and B is symbolized by the notation A→B, which means that B depends on A.

This paper organizes dependencies into quite a lot of types, which are *strong*, *weak*, *direct*, *indirect*, *one-way* and *cyclic* dependencies. A dependency (A→B) is called a *strong dependency* when each change in A makes a transformation in B, where a *weak dependency* means that a vary in A may or may not make a variation in B. For example, the dependency [Rank → Salary], which illustrates the dependency between the Rank of an academic staff and his/her Salary, is a strong dependency, while the dependency [Score →Grade], which illustrates the dependency between the Score of a student and his/her Grade, is a weak dependency. A *direct dependency* means that a modification in A makes a straight change in B, where a *transitive dependency* means that a change in A makes a change in intermediate data items which then makes a change in B. A *one-way dependency* means that B depends on A, but A does not depend on B, whereas a *cyclic dependency* means that B and A depend on each other.

These dependencies may be applied by insiders to infer unauthorized information. For example, presume that an insider has a read right on a data item B, which depends strongly on another data item A. Then, a transformation in B makes the insider surmise that a change has been happened in A. In addition, if the insider identifies the constraints on the dependency (constraints on dependencies are discussed in the next section), she/he can presume the new value of A. Homogeneously, if the value of B is not transformed, the insider knows that the value of A has not been varied. On the other hand, in case of weak dependencies, if the value of B is not transformed, this does not mean that the value of A is still the same.

Relational databases have several levels of granularities. These levels are sorted into the low level (attribute level), the intermediate level (record level) and the high level (table level). In this paper, these kinds of dependencies are discussed at various levels of granularities. All types of dependencies, apart from the cyclic dependency, are found at the attribute level. Such examples are simple to find. All types of dependencies are established at the table level since a table inherits the dependencies present at its attribute level that is, a dependency between two tables is essentially a dependency between attributes that belong to those tables. Note that two tables may have more than one type of dependency.

Similarly, records inherit dependencies from their attributes. This means that a variety of types of dependencies exist at record level also. Two records having a dependency relationship may survive in the same table or in two different tables. When the records fit in to the same table, there are two possibilities. Initially, an attribute depends on itself, which is called a *self-dependent* attribute. Secondly, the attributes of the dependency are different. This case survives when a self-dependent attribute is also dependent on another attribute.

Constraints and Dependencies

A dependency relationship may engage a constraint. Specifically, a change on a dependent data item (right side) happens only when a specified constraint is satisfied (on the left side). This paper categorizes constraints into two types: *changing the value of an attribute*, and *deleting/inserting records*. Characterizing dependencies and constraints between data items makes possible understanding the relationships between them and the overall structure of relational database systems. Furthermore, it facilitates discovering the vulnerabilities that a relational data base may have. This paper pioneers the CDG, which represents the dependencies and constraints in sort to investigate the flow of information between different data items at several granularity.

Petri Nets are used to build up the CDG. To show how Petri Nets can be employed to represent dependencies and constraints between data items in relational databases, think about the two tables $T1(a1, a2, a3)$ and $T2$ $(a4, a5, a6, a7)$ that have the following dependencies and constraints: $\{(c1 \leq a1 \rightarrow a2=c3), (c1>a1 \rightarrow a2=c4), (a4=2*a3 + 3), (a6=6*a2 + 2*a5)\}$. Figure 1 illustrates the CDG of this relational data base using Petri Nets. Every attribute is represented by a circle, and a dependency is symbolized by an edge (arrow) from the source attribute (left side) to the destined (dependent) attribute (right side). Constraints are placed on top of bars. For example, to vary the value of the attribute $a2$ in table $T1$ to $c4$, the value of the attribute $a1$ in table $T1$ should be set to a value less than $c1$. The final constraint is represented by writing $(T1 \cdot a1<c1)$ above the transition (bar). Note that each attribute is headed by the table to which it belongs, which expands the possible use of CDG to all granularity levels.

The CDG characterizes the constraints of the first type. To illustrate both types of constraints at the table level (and implicitly the record level), the *Dependency Matrix* is used. Table 1 symbolizes an example

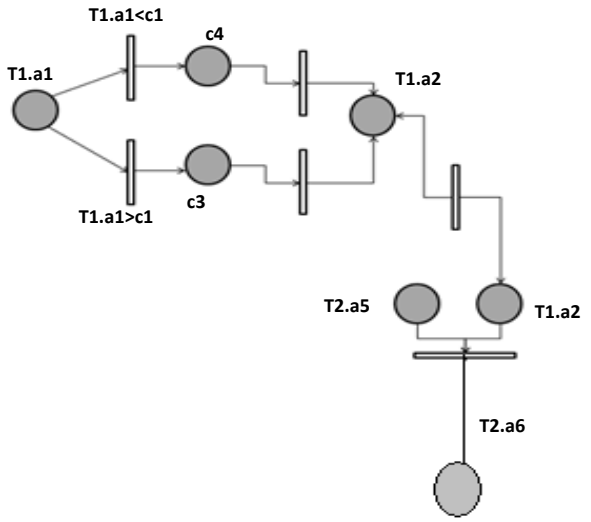

Figure 1: Constraint and dependency graph.

	T1	T2	T3	T4	T5
T1	-	[(C1,2),(C8,2)]	(C2,2)	0	0
T2	0		0	(C3,2)	0
T3	0		-	0	(C4,2)
T4	0		0	-	0
T5	0		(C6,2)	0	-

Table 1: Dependency matrix.

of a Dependency Matrix that shows dependencies between different tables as well as the constraints on such dependencies. The first row and first column Characterize tables. Every cell contains a set of pairs (*C, T*), where *C* denotes a constraint, and *T* indicates the type of the dependency. The value *2* means a strong dependency, while the value *1* means a weak dependency.

Insider threat: Unauthorized Knowledge Acquisition

This section reveals how insiders can get unauthorized information and widen their knowledge bases using dependencies and constraints. Additionally, it discusses approaches for predicting and preventing such threats.

Insiders knowledge bases (KBs)

Aacquaintance base (AB) decides which data items a insider has read. Actually, it is a profile of insider accesses to data items. Knowledgebase is raised based on the several levels of granularities of relational databases.

To calculate how much information the insider has about specific attributes, the NDIG [3] is used. The Neural Dependency and Inference Graph (*D,N,W,E*) (NDIG) gives you an idea about dependencies among relational database data items and the amount of information that can be inferred regarding them using dependencies, where *D, N,W* and *E* indicate data items, neurons, weights on edges and edges, correspondingly. Figure 2 gives you an idea about an example of the NDIG of the database in Figure 1, where data items are symbolized by rectangles and neurons are represented by circles. Arrows symbolize dependencies, where the destination data item depends on the source data item. Weights on edges give details about the amount of information that can be inferred about destination data items using the source data items. For instance, the attribute *a*2 alone can be used to infer 60% of information about *a*6. On the other hand, *a*2 and *a*5 can be used jointly to infer 100% of information about *a*6. These values are calculated by calculating the uncertainty of the value of a destination data item with and without getting the value of source data item(s). For instance, the uncertainty of the letter grade of a student in a course is 5 since the possible values are *A, B, C, D* and *F*. Nevertheless, without loss of generalization, with getting the Score of the student, the improbability of the letter score is 0 since, in this case, the letter grade is calculated exactly. Next, the amount of information that can be obtained about the letter grade given the Score is computed as follows: *(5−0)/5* = 100%.

Acquaintance algorithm

Algorithm 1 demonstrates the algorithm for building the acquaintance graph, which symbolizes knowledgebase (KB) at several levels of granularity. It makes use of the NIDG as well as CDG of the relational database beneath consideration as well as the Dependency Matrix. The algorithm appends the insider as a root of the knowledge graph. The second level of the graph encloses the tables to which the insider has read right (directly or by inference) (Figure 3). For every table in the second level, the algorithm decides to which attributes the insider has read access. The NIDG is used to tag edges by the amount of information the insider can have about apiece data item (attribute or

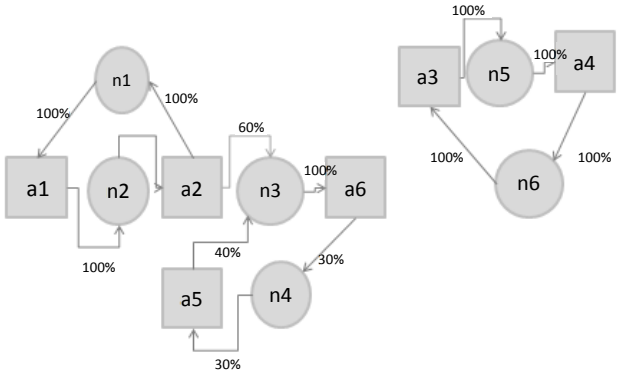

Figure 2: The NDIG of the database.

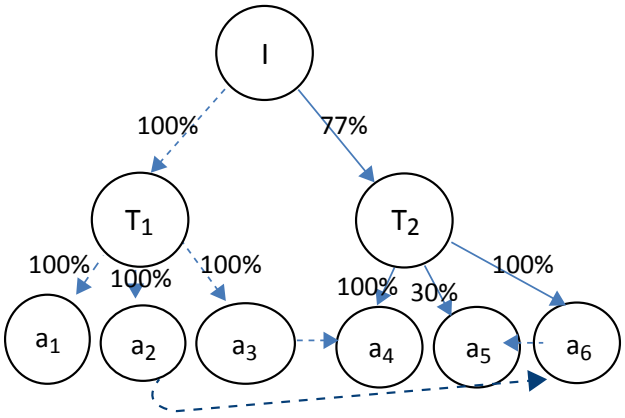

Figure 3: Acquaintance graph of insider.

table). NDIG or CDG is used to show dependencies between knowledge units (attributes), which are corresponded by an edge (arrow) from the source attribute to the destination (dependent) attribute. Additionally, the CDG is used to show what values of attributes are accumulated in the KB of the corresponding insider as mentioned earlier, which is used in insider threat predicting and preventing later in Insider threat prediction and prevention. Note that the amount of information the insider has about a table is the standard of all information she/he has about all attributes belonging to the Table 1.

Algorithm 1: Acquaintance base (AB) Algorithm

Input: An insider I, Dependency Matrix, CDG, NDIG, Set of tables that insider has direct read access.

Output: The Acquaintance graph of the insider I.

1. Initialize the AG=(V,E), where V=I, E={}, and insider I.

2. **For** each table T*k*in D//*add directly accessed tables*

3. V=V UT*k*//*add the node Tk to AG*

4. E=E Ue(I, T*k*)//*add the edge e(I, Tk) to the AG*

5. **For** each t ∈attributes(T*k*) and the insider has a readaccess to it//*add directly accessed attributes*

6. V=VUt //*add the attribute t to AG*

7. E=E Ue(T*k*, t)//*add edge e(Tk, t) to the AG*

8. **Endfor**

9. **Endfor**

10.**For**each T*k*∈D do//*consider dependencies*

11. **For**each Safe cluster R to which T*k*belongs

12. ∀X∈R ∧∉X/∧D→X/∉AG(I)//*exclude X from AG(I)*

13. **Endfor**

14. **For**each Hot cluster H to which T*k*belongs

15. **For**∀T*m*∈*H*∧T*m*≠T*k*

16. V=V UT*m*//*Add the node Tm to AG*

17. E=E Ue(I, T*m*)//*add edge e(I, Tm) to the AG*

18. **For** each t*m* ∈attributes(T*m*) ∧t*k*→ t*m*,where t*k*∈attributes(T*k*)//*add the directly inferred attribute(s) to the AG*

19. V=V Ut*m*//*add the attribute tm to AG*

20. E=E Ue(T*m*, t*m*)//*add e(Tm, tm) to AG*

21. E=E Ue(t*k*, t*m*)//*add e(tk , tm) to AG*

22. **Endfor**

23. **Endfor**

24. **Endfor**

25. **For** each other table T*s* that has dependency (one-way)withT*k*//*add tables from other clusters*

26. V=V UT*s* //*Add the node Ts to AG*

27. E=E Ue(I, T*s*) //*add edge e(I, Ts) to the AG*

28. **For** each ts∈attributes(T*s*) t*k*→ ts(direct dependency),where t*k*∈attributes(T*k*)//*add the directly inferred attribute(s) to the AG*

29. V=V Uts//*add the attribute ts to AG*

30. E=E Ue(T*s*, ts)//*add e(Ts, ts) to AG*

31. E=E Ue(t*k*, ts)//*add e(tk, ts) to AG*

32. **Endfor**

33. **Endfor**

34. **For** each table T*j*that depends transitively on T*k*

35. V=V UT*j*//*add the node Tj to AG*

36. E=E Ue(I, T*j*)//*add edge e(I, Tj) to the AG*

37. **For** each t *j* ∈ attributes(T *j*)∧t*k*–→ t *j* (transitive dependency), where t*k*∈attributes(T*k*)//*add the transitively inferred attribute(s) to the AG*

38. V=V Ut *j*//*add the attribute t j to AG*

39. E=E Ue(T *j*, t *j*)//*add edge e(T j, t j) to the AG*

40. E=E U*e*(t*k*, t *j*)//*add e(tk, t j) to AG*

41. **Endfor**

42. **Endfor**

43. **Endfor**

44. **For** each edge e(T,t)UAG//*T is a table and t is an attribute*

45. Weight (e(T,t))=the amount of information the insider has about t//*weights of attributes using NDIG*

46. **Endfor**

47. **For** each edge e(I,T)∈AG//*weight of tables*

48. Weight (e(I,T))=

$$\sum_{i=1}^{n} Weight\,(e(T,t_i))/n$$

where n is the number of attributes in T.

49. **Endfor**

Insider threat prediction and prevention

Building the knowledge graph of an insider assists in predicting and preventing insider threat (revelation of unauthorized information). To attain this goal, the threat prediction graph (TPG) is used, which is built based on the knowledge graph. Previous to defining the TPG properly, let us introduce the threat prediction value (TPV). A TPV is a value stock up in each attribute that belongs to the TPG of the insider and used to foresee inside threat. A TPG is calculated as follows:

$$TPV\ (k) = f\ (k)/T\ (k) \quad (1)$$

Where k is an attribute, $f\ (k)$ is the amount of information the insider has about k, and $T\ (k)$ is the threshold value of k (the amount of information that the insider is permitted to get about k) according to the insider under deliberation. The TPG uses TPV to sense and prevent insider threat.

Figure 4 shows an instance of a TPG. The NDIG, the AG and the set of threshold values according to the underlying insider are used to

construct the TPG. A threshold value of an attribute according to an insider represents the percent amount of information that the insider is allowed to get about the data item 100% indicates that the insider can get full information about the data item, and values less than that indicate that the insider can get partial information about the data item.

As discussed earlier, the amount of information that an insider gets about a data item is retrieved using theNDIG.

Algorithm 2: TPG Algorithm

Input: An insider I, the set of threshold values according to the insider, NDIG, the acquaintance graph AG of the insider.

Output: The Threat Prediction Graph TPG of the insider I.

1. Initialize the set of pairs T={(KU,TKU)}, where TKUis the threshold value about a knowledge unit KU according to the insider I, an empty set S={}

2. Recall the AG of the insider and the NDIG, initialize theTPG as TPG=AG, but without labels

3. **For** each KU ∈V(TPG)//*KU is a knowledge unit and Vis the set of vertices*

4. TPV(KU)=f(KU)/T(KU)//*compute the TPV of KU*

5. **Endfor**

6. **For** each requested knowledge unit RKU by the insider

7. **If** TPV(RKU) >1//*threat predicted*

8. Deny this request

9. **Else**//*add RKU temporarily for further inspection*

10. V=VU{T*k* }//*add table Tk,where RKU ∈Tk andTk/∉TPG*

11. E=E U{e(I, T*k*)}//*add an edge if e(I, Tk)/∉E*

12. V=V U{RKU}//*add a node for RKU*

13. E=E U{T*k*, RKU}//*add an edge to the TPG*

14. TPV(RKU)=f(RKU)/T(RKU)

15. **For** each knowledge unit KU*x*that has a dependency with the RKU//*add inferred attributes*

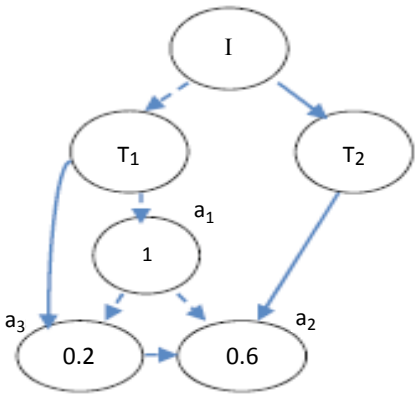

Figure 4: Threat prediction graph.

16. **If** TPV(KU*x*)>1//*threat predicted*

17. Deny RKU and remove it from TPG

18. **Else**//*no threat so far, still needs further inspection*

19. **If** KU*x*/∉V//*not in the TPG*

20. **If** KU*x* and RKU are not in the same table//*add inferred attributes*

21. V=V U{T*x*}//*add table Tx ,where KUx∈Tx andTx/∉TPG*

22. E=E U{e(I, T*x*)}//*add an edge if e(I, Tx)/∉E*

23. V=VU{KU*x*}//*add a node for KUx*

24. E=E U{T*x*, KU*x*}//*add an edge to the TPG*

25. TPV(KU*x*)=f(KU*x*)/T(KU*x*)

26. **Else**

27. V = V U{KU*x*}// *Add a node for KUx*

28. E=E U{e(RKU, KU*x*)}//*add an edge*

29. TPV(KU*x*)= f(KU*x*)/T(KU*x*)

30. **Endif**

31. **Else If** KU*x* $\in$ V//*already in the TPG*

32. Add KU*x*to the set S

33. E=E $\cup$ {e(RKU, KU*x*)}//*add an edge to the TPG*

34. Update the TPV of KU*x*//*recalculate its TPV*

35. **Endif**

36. **Endif**

37. **Endfor**

38. **For**each KU in S

39. **If** TPV(KU) >1//*threat predicted*

then there are two choices://*threat prevention*

40. **First**: Grant access to RKU but revoke accesses toa knowledge unit(s) that has the following three properties:

(**a**). It already exists in the knowledgebase of the insider.

(**b**). Can be used in conjunction with RKU to compromise unauthorized information about KU.

(**c**). The lifetime of the knowledge unit(s) is expired.

41. **If** the first solution does not hold, use the second solution as follows:

42. **Second**: Do not grant access to the RKU and copythe TPG initialized in steps 2-6 and restore it here.

43. **Endif**

44. **Endif**

45. **Endfor**

46. **Endif**

47. **Endfor**

Algorithm 2 shows the algorithm for detection and prevention of insider threat using TPG. In this algorithm, a knowledge unit KU represents an attribute. A knowledge unit is considered a threat if its TPV is greater than one.

Results and Discussion

Insider threat has become an imperative security issue to the organizations confidential information. Different researchers introduced different characterizations for insiders at system level, whereas others defined the insider according to different classes. Nevertheless, this system defines the insider at the relational databases level. They termed the insider as a person who has access privileges, is familiar with dependencies and their constraints and is recognizable with the system under consideration. Some researchers used existing methods of detecting outside threat, such as using honeypots, to detect insider threat. However, these methods are not efficient since insiders and outsiders use different corridors or approaches to attack systems.

Other researchers introduced new methods to deal with this dilemma. They introduced a new knowledge base loom to detect and prevent insider threat. Nevertheless, most of the research was at the system level and did not reflect on relational databases, where insiders have more capabilities such as the knowledge about dependencies and constraints that make possible launching attacks.

Jabbour and Menasce [12,13] suggested a self-protection mechanism, which is the Insider Threat Security Architecture (ITSA) [7] that is totally integrated into the computing system. Nevertheless, their self-protection mechanism has scalability problems that may put down the performance of hosting systems.

Dependencies in relational databases play a major role in insider threat problem. The proposed system addresses the insider threat in relational databases and developed the Neural Dependency and Inference Graph (NDIG), which demonstrates the dependencies amongst data items and the amount of information that can be deduced about data items using dependencies. The system also shows the Knowledge base graph produced based on the Knowledge base algorithm. The Knowledge base graph demonstrates the amount of information that the insider has regarding data items.

To forecast and avoid insider threat, the system defines and uses the threat prediction graph (TPG). A TPG demonstrates the threat prediction value (TPV) of each data item in insiders AG, where TPV is used to lift up an alert when an insider threat occurs, where other methods do not use this approach to prevent the threat.

The graph in Figure 5 shows the relation between different insiders and their access level. The graph also shows the declining right to use level of the insider 1 due to the threat he possess.

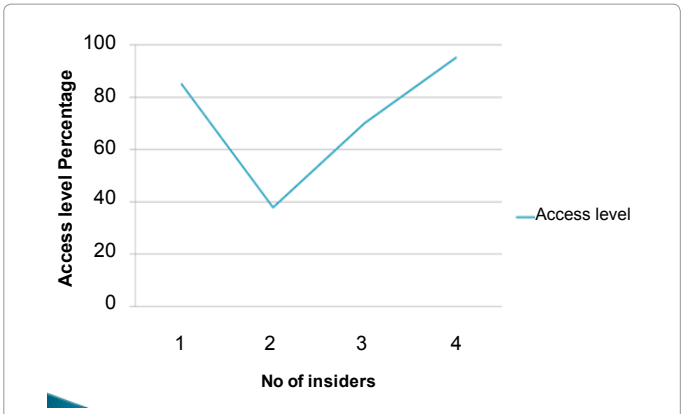

Figure 5: Relation between different insiders and their access level.

Conclusion and Future Work

The proposed system has investigated the problem of insider threat in relational database systems. It has considered different levels of granularities of data items and identified various types of dependencies. Moreover, it has shown how insiders who have knowledge about dependencies may infer information about unauthorized data items. Additionally, the paper has explained how constraints on dependencies play an important role in knowledge acquisition. It has introduced the constraint and dependency graph (CDG) and the Dependency Matrix these data structures show diverse types of dependencies and constraints in relational database systems. An algorithm for constructing insider's knowledgebase has been provided. The algorithm helps in building the knowledgebase of an insider and determines the data items which the insider can access in unauthorized way. Moreover, the paper has defined the threat prediction graph (TPG), which is used to predict and prevent insider threat. An algorithm for predicting and preventing insider threat has been stated. As a future work, the recommended system has been planned to expand the proposed approaches to general access control systems.

References

1. Brackney R, Anderson R (2004) Understanding the insider threat. RAND Corporation.
2. Brodsky A, Farkas C, Jajodia S (2000) Secure databases: constraints, inference channels and monitoring. IEEE Trans. Knowl. Data Eng 12: 900-919.
3. Bishop M, Gates C (2008) Defining the insider threat. Cyber security and information intelligence research.
4. Chagarlamudi M, Panda B, Hu Y (2009) Insider threat in database systems: preventing malicious users activities in databases. Information Technology.
5. Murata T (1989) Petri nets: Properties, analysis and applications. IEEE.
6. Yip R, Levitt K (1998) Data level inference detection in database systems. Computer Security Foundations Workshop.
7. Althebyan Q Panda B (2007) A knowledge base model for insider threat prediction. IEEE Workshop on Information Assurance and Security.
8. Yaseen Q, Panda B (2009) Knowledge acquisition and insider threat prediction in relational database system. Computational Science and Eng, IEEE.
9. Yaseen Q, Panda B (2010) Prediciting and preventing insider threat in relational databases. Information Security Theory and Practice 6033: 368-383.
10. Yaseen Q, Panda B (2010) Organizing access privileges: maximizing the availability and mitigating the threat of insiders knowledge-base. Network and System Security, IEEE.
11. Spitzner L (2003) Honeypots: catching the insider threat. Computer Security Applications Conference, Washington.
12. Jabbour G, Menascé DA (2009) The insider threat security architecture: a framework for an integrated, inseparable, and uninterrupted self-protection mechanism. IEEE, Computational Science and Engineering.
13. Jabbour G, Menascé DA (2009) Stopping the insider threat: the case for implementing autonomic defense mechanisms in computing systems. Information Security and Privacy.
14. Farkas C, Toland T, Eastman C (2001) The inference problem and updates in relational databases. Database and Application Security 87: 181-194.