Detection of Insiders Misuse in Database Systems

Nahla Shatnawi, Qutaibah Althebyan, and Wail Mardini

Abstract— **Almost all systems all over the world suffer from outsider and insider attacks. Outsider attacks are those that come from outside the system, however, insider attacks are those that are launched from insiders of the system. In this paper we concentrate on insider attacks detection on the application level; database is our focus. Insider attacks differ from outsider attacks in many ways; most importantly, insiders have more knowledge about the underlying systems. Because of their knowledge and their privileges of the system resources; their risk can be greater and more severe. In fact, insiders can find vulnerabilities in the system easily. Several techniques have been proposed that tackled the insider threat problem, but most of them concentrate on insider threat detection in computer system level. We describe a method for insider threat detection in database systems that handle entrants on the role of insiders for such attacks. Our simulation results show resistance against such attacks. Also, our results show good performance in terms of reducing false alarms to the minimum.**

Index Terms—**About four key words or phrases in alphabetical order, separated by commas, for example, visual-servoing, tracking, biomimetic, redundancy, degrees-of-freedom**

I. Introduction

Insider attacks are a well-known problem acknowledged as a threat as early as 1980s [1], but few works have been conducted to deal with this problem effectively. Intruders are divided into two types' insiders and outsiders. But most Intrusion Detection Systems (IDSs) are designed to deal with outsiders. IDSs aim to help in the detection of important types of computer security violations. Although outsider attacks are greater than the number of insider attacks, the insider attacks are more severe [2]; these kinds Many definitions exist in the literature for the insider. For example, the authors of [3] define the insider as a person who has privileges to access the underlying system. Another definition of an insider would be "an individual who has the knowledge of the organizations information system structure to which he/she has authorized access" [4]. In [5] the authors stated that an insider can be an employee that uses his/her privileges to do activities based on his/her knowledge.of attacks are very serious and dangerous due to their nature which includes personnel who have privileges and authorizations to access organizations' resources. Such attacks also involve personnel who have, in addition to their privileges, knowledge of the information system resources and might know some vulnerabilities of the system. This makes the problem more severe and serious. The insider has general knowledge of everything in the organization as a whole. For example, knowledge of other insiders in the organization, knowledge of different kinds of dependencies style menu. The style will adjust your fonts and line spacing. In our work use the last definition, where the insider is one of the organization/company employees and this explain how he gains privileges and knowledge. The insider knowledge of the underlying system entails him/her: access all documentation on the underlying system, collect intelligence and perform discovery without suspicion, know detailed information about objects of the system and have good intelligence on the entire system [7]. Because of the nature of insiders who pose great risks on assets of the system, those risks or threats should be discovered and revealed either before they take place or as soon as they take place.

Several models have been recently proposed that tackled the insider threat problem, but very few models concentrate on the application level such as databases. Databases is very important where it contains mission-critical and sensitive data; data that have been coordinated and maintained over usually long period of time, which make their loss or damage more costly. Most organizations and e-businesses use Database Management System (DBMS) to manage and deliver missions' critical and sensitive data. Database is a collection of named data items (objects), or a collection of related data [7]. Databases used to save the data that have been collected and maintained over usually long period of time were loss of such data will cost more than any other components [8].

As Kamra et al noted in [9], intrusion detection mechanisms for DBMS are important for the following three reasons; First, malicious actions on DBMS not necessarily malicious for the operating system or the network; and the IDSs for them are not effective for database attacks [8]. Second, data care done by various government regulations concerning with data management such as SOX, GLBA, HIPAA and so on, has been prepared by many organizations. Third, the most important reason, where the insider threat problem is recognized as a major threat, its solution needs a mechanism to detect access anomalies by insiders.

II. Related Work

In this section we discuss some related works about insider

Nahla Shatnawi is a graduate student of the Computer Science Department in the Jordan University of Science and Technology.

Qutaibah Althebyan is an Assistant Professor in the Software Engineering Department, Jordan University of Science and Technology, Irbid, Jordan P O Box 3030, 2210 (email: qaalthebyan@just.edu.jo).

Wail Mardidni is an Assistant Professor in the Computer Science Department, Jordan University of Science and Technology, Irbid, Jordan, P O Box 3030, 22110 (email: wail@just.edu.jo).

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

attacks. The insider threat problem may exist on different levels: network level, system level, and finally application and data level, where our work is concentrated. We mention methods and techniques that are proposed to detect, predict and prevent, and assess such threats. Based on [8] there are three methodologies for intrusion detection systems that have been used, which are: **Misuse Detection**- based on *signatures* that demonstrate the characteristics of well-known system vulnerabilities and attacks. It works well with known misuse patterns but fails with new ones, **Anomaly Detection-** based on the behavior of the matter, e.g., user, application, or component of a system. It's the most popular way of detection. It is better than the misuse detection methodology because it has a better opportunity to detect previously unknown attacks and Insider Misuse- sources ranging from discontent employee (database administrators, application developers, application users), who may maliciously damage the data integrity to outsider that gain access to the data [10]. The problem of insider misuse is more dangerous in database systems because it works and manages critical data.

A detection-oriented approach classifies insider misuses based on the level of the system at which they might be detected. The basis for this is that different types of misuses manifest themselves at varying levels of the system (e.g. some may be apparent at the network level, whereas others are most visible at higher levels, such as the operating system or application levels) [11]. Network-level misuses do monitoring activity at the network traffic level. System-Level misuses do monitoring at the system level necessitates that monitoring activity be conducted upon individual host systems. Our concentration will be on the application and data-level misuses.

We choose to work on the database systems from application and data level; due to the database importance. The techniques that are proposed for network and system level are not sufficient for application and data level. Existing intrusion taxonomies mainly describe characteristics of various attacks, and not developed specifically for monitoring insider misuse. Anderson in [11] was the first person to perform classification for insiders who may misuse the IT systems into, masqueraders, clandestine users, and misfeasors. However, this classification only characterizes the type of users and not the actual misuse or how they may be detected. Another technique concentrates in detecting anomalous access patterns in relational databases. This approach is based on mining SQL queries from database log files. The mining process results used to form profiles that model the normal behavior of database access to determine intruders. Kamra et al [13] consider two scenarios, in the first scenario; they assume that database has a Role Based Access Control (RBAC) model. Their ID system can identify role intruders, which behave differently than expected. The advantage of combining ID with the RBAC databases is that it can do protection against insider threats. In the second scenario, they assume that no roles are associated with users of the database. So, they can look to users' behavior directly.

III. Problem Statement

Databases of an organization might face similar problems of insider and outsider attacks to companies or organizations. Employees (insiders) of the organization might try to mislead the system by trying to implement actions that look fine to the system; however, they can be very dangerous. In this paper we built an insider threat model that takes care of such threats on the application level; threat is considered as an activity that violates the security policy of a system [8]. In our model we seek preventing insider threats as well as detecting possible threats for individual tasks. As an indication of the effectiveness of our proposed model, we want to minimize the false alarms percentage which will be shown in our results and analysis chapter.

A. Model

In our model, we assume that the number of tasks an insider can execute is limited. We assume that the number of transactions the insider can execute is limited. This means that the order of executing a task can be predefined and hence, the order of executing the associated transactions that are needed to perform the associated task can also be predefined. In our model, we try to control the insiders by watching their sequence of actions. An insider before implementing his/her action has to declare his/her intention by specifying the work that he/she needs to do. Based on that, a list of objects/ transactions that he/she needs to follow to finish his/her task and the data items (objects) that he/she can use will be prepared by the system. After specifying the set of objects and transactions by the system based on the user intended work, a dependency graph can be set up; this shows dependency relationships among objects and/or transactions. This process can be achieved because we assume that the number of tasks and the number of transactions are limited. Based on these dependencies, the sequence of objects to be accessed within a transaction and the sequence of the transactions themselves to be executed can be determined. For example: to perform task TS1, we need to perform the following transaction in the following order: T1 → T2 → T3 (and so on). Moreover, within each transaction, we can specify the set of objects and operations to be performed in a specific order: T1: R1 → W2 → W3. Each transaction's information will be kept in the log file. Any deviation from the above mentioned order of transactions and/or objects in the log file will be a possible threat. In case this situation happens, our model should detect this situation and catch it. The following contains important definitions that will help in understanding our model and its description. We will start by defining task:

Definition: a task is some work done by a user and can be specified in many ways [7].

Definition: in [14] Yi Hu et al introduce user's tasks as "a group of transactions that are always submitted to the DBMS together to achieve a certain goal". For example, in order to perform withdrawal in a banking application, many transactions may be sent to the database consecutively to fulfill the task. An application program may contain several

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

transactions separated by the transaction boundaries (i.e. Begin and End of transaction). And it's used to interact with database [7].

Definition: a transaction is a logical unit of database processing that includes one or more access operations (i.e. read -retrieval, write - insert or update, delete) [14].
The definition of equivalent tasks based on the next definition, where two tasks are equivalent if the transactions that these tasks used are equivalent, e.g. $task_n$ and $task_m$ are equivalent if $task_n$ transactions are equivalent to $task_m$ transactions.

Definition: semantically equivalent transactions are "transactions that have the same effect on the database as a sequence of transactions. If (T_1, T_2… T_n) is a sequence of transactions, then the semantically equivalent transaction of this sequence is denoted by $T_{1,2,\ldots,n}$".

Definition: dependency graphs help in showing dependencies among transactions and/or objects. In fact, this technique can be used and implemented because objects of the database and hence transactions on database are dependent of each others [8].

Definition: read operation for object x can be defined as read_item(X) which reads a database item named X into a program variable [1].

Definition: write operation for object x can be defined as write_item(X) writes the value of program variable X into the database item named X [1].

B. Model Description

We discuss in this section the details and operations of our model. Our model will work in two phases, the first phase is to limit the number of malicious activities that the user can do; this is done by put limitations on the user work and on the way of doing this work. In this step we assume that the user work is limited i.e. the user can do limited set of tasks. Each task can work using limited number of transactions in specific order. The second phase detects possible threats.

First, in the proposed model the user needs to identify his/her intended work that he/she wants to do under specific application; the task must be from a limited set of tasks that he/she is authenticated to do. Then, the system will call the database; to return ordered transactions and objects from a predefined set for the called task to user, and transactions and objects for the equivalent task if exist. The user uses these transactions and objects (of the task or of the equivalent) to do his/her work; in normal ways or as a threat. The system will draw dependency graphs for the user work that is stored in the log file, for the task, and for the equivalent task (if exist). A comparison of user work graph with the task graph, and with the equivalent task graph (if exist) is performed to find possible threats. Any transaction performed by any user will be stored in the log file. Any deviation in the order of transactions and/or objects displayed by the dependency graph (for task/equivalent task) comparing with the log file transactions and/or objects graph (of user work) might be a possible threat. In case this situation happens, our model should detect this situation and catch it.

IV. Simulation and Results

This section demonstrates the main assumptions used in the simulation explained by examples. We discuss how the simulator works based on these assumptions. The following are our assumptions: We consider twenty users, each user has limited number of tasks to do.
Example: U1 TS1 TS2 TS3 TS5 TS32 TS33 TS34 ……
U2 TS3 TS5 TS6 TS7 TS8 TS9 TS10
U20 TS4 TS5 TS8 TS14 TS15 TS32 TS33 ……..
Where U: user, TS: task, Number: define the user or the task.

We consider that there are five applications; some users can work under more than one application.

Example: APP1 U1 U2 U3 U4 U17
APP2 U1 U2 U3 U11 U16 U19 U20
……
APP5 U6 U10 U12 U13
Where APP: application.

We consider that there is 100 individual tasks run under the five applications; some tasks can run under more than one application.

Example: APP1 TS1 TS2 TS3 TS6 TS7 ……
APP3 TS1 TS2 TS16 TS17 TS18 …….

We divide the tasks into normal tasks and malicious tasks. If a user do malicious task, then this must be discovered because it is a threat. The table below contains the normal and malicious tasks.

TABLE 1: NORMAL TASKS AND MALICIOUS TASKS

No. of transactions per task	Normal tasks	Malicious tasks
1	TS6-TS10	TS1-TS5
2	TS11-TS15	TS16-TS20
3	TS21-TS24,TS30	TS25-TS29
4	TS31,TS32,TS38-TS40	TS33-TS37
5	TS41,TS47-TS50	TS42-TS46
6	TS56-TS60	TS51-TS55
7	TS61-TS63,TS69,TS70	TS64-TS68
8	TS71-TS74,TS80	TS75-TS79
9	TS81,TS82,TS88-TS90	TS83-TS87
10	TS91-TS95	TS96-TS100

We consider that there is a varying number of transactions per task with predefined order; tasks can take one or more transactions as in the following example.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Example: TS1 T1
TS12 T20 T1
…….
TS100 T19 T10 T6 T7 T2 T16 T5 T18 T4 T3 T20

We consider twenty transactions that can be used by different tasks in specific and predefined order. The program starts when a user needs to do some work from an application (user and its intended work generated randomly). The same user can do the same work more than one time. Different users can do same work.

Example 1: U1 TS1,
Task: TS1 T1 R1 W5 W2 R9 W12 R4 R6
Equivalent Task: TS4 T3 R4 R1 W5 W2 W12 R11
Where T1, T3 are the transactions 1 and 3 respectively. R: mean read operation, W: mean write operation, the number identify the transaction/object.

From Example 1, User 1 wants to do Task 1; Task 1 has Task 4 as equivalent task. The user can do his/her intended work by doing transactions of Task 1 or of Task 4. After that, the user may do the work in one of the next three ways:

- The user may do a task, or do the equivalent task to finish his/her intended work (task) in its normal way; i.e. no threat will happen if the user does it as is.
- The user may do part of task transactions and (or) part of the equivalent task transactions or transactions of both in different order. Threat will happen.
- The user may do the task transactions or the equivalent task transactions in different order comparing with the predefined. Threat will happen.

From Example 1 the user may do the work in one of the following ways:

- U1 TS1 T1 R1 W5 W2 R9 W12 R4 R6 ,
- U1 TS4 T3 R4 R1 W5 W2 W12 R11 ,
- U1 TS1 T1 R1 W5 W2 R9 W12 R4 R6 T3 R4 R1 W5 W2 W12 R11 , or
- U1 TS1 T3 R4 R1 W5 W2 W12 R11 T1 R1 W5 W2 R9 W12 R4 R6

The user work will be stored in the log file; a timestamp will be added to determine the time that the user does the work in. See Example 2.

Example 2: U1 TS1 starts
T1 R1 W5 W2 R9 W12 R4 R6 T3 R4 R6 R1 W5 W2 W12 R11
Where User 1 does Task 1

The user work will be displayed as a graph after the simulator read it from the log file. The task and the equivalent will be displayed into graphs too as in Example 3. We use graphs to display the user work, task, and equivalent task and to make comparison between user work and task/ equivalent task more efficient.

Example 3: based on Example 2, User 1 does Task 1 by doing Transaction 1 then Transaction 3. The simulator reads this user and its work from the log file and draws a graph for it as in the following figure.

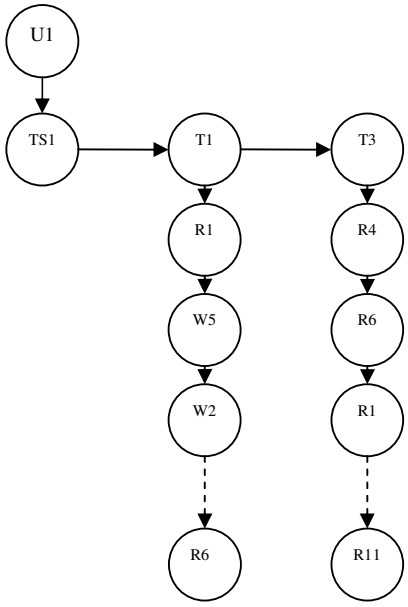

Figure 1: User and its intended work.

A. *Results and Analysis*

In our proposed model we concentrate on minimizing the false alarms percentage; false positive and false negative. Table 2 shows the simulator results when we run it on the set that contains tasks that have one transaction, on 20, 40, 60, 80, and 100 tasks. As in the table our model gives bad results.

TABLE 2: RESULTS ON SET OF ONE TRANSACTION PER TASK

No. of users tasks	No. of threats detected	No. of threats	False negative	Percentage of false negative
20	1	11	10	0.91
40	3	19	16	0.84
60	8	43	35	0.81
80	9	48	39	0.81
100	14	70	56	0.8

Table 3 shows the simulator results when we run it on the sets that contain five transactions per task, on 20, 40, 60, 80, and 100 tasks. The results in these tables show that our model gives better results when we increase the number of transactions per task; the number of comparisons increase.

TABLE 3: RESULTS ON SET OF FIVE TRANSACTIONS PER TASK

No. of users tasks	No. of threats detected	No. of threats	False negative	Percentage of false negative
20	12	15	3	0.2
40	28	33	5	0.15
60	47	54	7	0.13
80	60	69	9	0.13
100	78	88	10	0.11

The following table shows the false negative percentage values when we run the simulator on different sets of tasks,

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

different number of tasks with different number of transactions per task. As the table shows, as we increase the number of tasks; the number of false alarms decreases and hence, the performance of our model behaves better.

TABLE 4: FALSE NEGATIVE PERCENTAGE ON DIFFERENT TASKS

No. of users tasks	Percentage of false negative (false negative/No. of threats)				
	1 trans/task	2 trans/task	3 trans/task	4 trans/task	5 trans/task
20	0.91	0.47	0.33	0.25	0.2
40	0.84	0.4	0.26	0.23	0.15
60	0.81	0.37	0.245	0.22	0.13
80	0.81	0.365	0.24	0.21	0.13
100	0.8	0.36	0.235	0.2	0.11

The next figure shows the previous table results. Obviously, our model detects more insider threats when the number of transactions per task increase; more comparison times. The more the insider threats detected the less number of false negative alarms, the smaller value for false negative percentage.

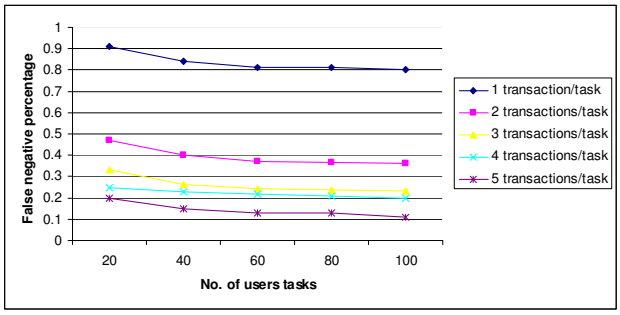

Figure 2: False negative alarms for different sets

The next figure shows that our model give good resistant against false negative alarms; small values for false negative percentage. Although the false negative percentage values when we run the simulator on 100 tasks greater than the value when we run the simulator on sets that contain 6, 7…, 10 transactions per task. We also find that our model decreases the false negative percentage value. Our model removes the false positive alarms; no false positive alarms for all previous simulator runs, the following figure shows the false positive values when we run the simulator on set of all sets.

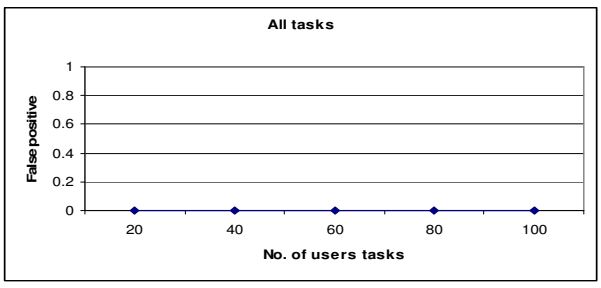

Figure 3: False positive alarms for different sets

Figure 3 shows that the false positive alarms value based on timestamp threats; threats done when not authenticated user try to do a task more than once in a small period. The false positive value is one when we run the simulator on 20, 40, and 60 tasks of different users, but this value become 0 for 80 and 100 tasks. Our model removes the false positive alarms as we increase the number of tasks that we run the simulator on.

V. CONCLUSIONS AND FUTURE WORK

In this paper we evaluated the insider threat problem. The insider threat attacks are very serious and dangerous due to their nature. The insider threat problem can be done on different levels: network level, system level, and application and data level; most importantly the database systems. Our focus is on the detection of insider threats at the application level; database systems. We proposed a new model that tried to prevent and detect insider threats in application level. Our model works in two phases, the first phase is to limit the number of malicious activities that the user can do; this is done by limiting the user work. The second phase detects possible threats. In our work we use graphs to compare the user work with the task and equivalent task (if exist). Our model shows good resistant against false alarms; false positive and false negative alarms. Our simulator results show that our model removes the false positive alarms, decrease the false negative alarms and the false negative percentage values when we increase the number of transactions per task; increase the number of comparisons.
Enhancement and improvement can be done for our proposed model. Some of intended works to do these improvements are: detect attacks that happens when a user do two tasks from different applications at the same time using multithreading; this may be more maliciously. So, a detection model must be developed to handle such attacks. Also, classification of threats based on changes that the user do on the application level and on the data level can be achieved. This will help in understanding of threats, made prevention and detection more reliable and minimize the false alarms.

REFERENCES

[1] Ramkumar Chinchani, Anusha Iyer, Hung Q. Ngo, Shambhu Upadhyaya. Towards a theory of insider threat assessment. IEEE CS digital library 2005; 108-117.
[2] Hyeran Mun, Kyusuk Han, Chan Yeob Yeun, Kwangjo Kim. Yet another intrusion detection system against insider attacks. The 2008 Symposium on Cryptography and Information Security 2008; 22-25.
[3] Nam Nguyen, Peter Reiher, Geoffrey H. Kuenning. Detecting insider threats by monitoring system call activity. IEEE CS digital library 2003.
[4] Qutaibah Althebyan. Design and analysis of knowledge-base centric insider threat models [dissertation]. University of Arkansas; July 2008.
[5] Mark Maybury, Penny Chase, Brant Cheikes, Dick Brackney, Sara Matzner et al. Analysis and detection of malicious insiders. International Conference on Intelligence Analysis; McLean, VA. 2005.
[6] Silberscatz, Korth, Sudarshan. Database system concepts. 4th ed. New York: McGraw-Hill; 2001.
[7] Silberscatz, Korth, Sudarshan. Database system concepts. 4th ed. New York: McGraw-Hill; 2001.
[8] Michael Gertz, Sushil Jajodia. Handbook of database security. Berlin: Springer- Verlag; 2007.
[9] Department of Defense. DoD insider threat mitigation: report of the insider threat integrated process team. Washington DC, USA: Technical report; 2000.
[10] A.H.Phyo, S.M.Furnell. A Detection-oriented classification of insider IT misuse. Network Research Group 2004.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[11] C. Chung, M. Gertz, K. Levitt. DEMIDS: A misuse detection system for database systems. 3ed Annual IFIP TC-11 WG 11.5 Working Conference on Integrity and Internal Control in Information Systems, Kluwer Academic; November 1999; California, USA.
[12] Ashish Kamra , Evimaria Terzi, Elisa Bertino. Detecting anomalous access patterns in relational databases. 2005.
[13] Yi Hu, Brajendra Panda. Design and analysis of techniques for detection of malicious activities in database systems. Journal of Network and Systems Management 2005; 13(3).
[14] Hui Wang, Shufen Liu, Xinjia Zhang. A prediction model of insider threat based on multi-agent. 1st International Symposium on Pervasive Computing and Applications 2006.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)