International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 01 | March-2015 www.irjet.net p-ISSN: 2395-0072

Anomaly Response System for Relational Database System Using Joint Administration Model

Mr. K. Awale[1], Prof. U. A. Nuli[2]

[1] *Student, ME (CSE)-II, D.Y.Patil College of Engg and Technology, Kolhapur, Maharashtra, India*

[2]*Professor, DKTE Society's Textile and Engineering Institute, Ichalkaranji, Maharashtra, India*

Abstract: *Data stored in the organization is one of its most important assets. The organizational data has to be protected from various threats that are posed from both insiders and outsiders. Not every user within the organization has the rights to view or modify this data. This restrictions on the access of data is to be implemented by the Database Management System (DBMS). But the current DBM) with their current security mechanisms may be of little help in providing security from internal threats. In this paper we propose a role based approach for anomaly detection system for security of the data from insiders based on the role assigned to them. Apart from detecting the anomalous data request the systems also needs to have a strong response policy. We propose three levels of response policies for responding to the anomalous requests. Additionally we propose a Joint Administration Model for providing security at the administrative level also.*

Keywords: Database, Anomaly, Anomaly Detection, role, response policy.

I. INTRODUCTION

Data stored in the organization is one of its most important assets. Some of these data are worth millions, and organizations take great care controlling access to these data, with respect to both internal users, within the organization, and external users, outside the organization. Data security has a central role in the larger context of information systems security.

The development of new Database Management Systems (DBMS) requires a revision of architectures and techniques adopted by traditional DBMS. An important component of this new generation security-aware DBMS is an Anomaly Detection (AD) mechanism. Even though DBMS provide access control mechanisms, these mechanisms alone are not enough to guarantee data security. They need to be complemented by suitable AD mechanisms. AD mechanisms helps in addressing the problem of insider threats, an increasingly important problem in today's organizations for which not many solutions have been devised.

Intrusions in an information system are the activities that violate the security policy of the system, and intrusion detection is the process used to identify intrusions. It is based on the beliefs that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions will be detectable. Intrusion detection complements these protective mechanisms to improve the database security. Moreover, even if the preventive security mechanisms can protect information systems successfully, it is still desirable to know what intrusions have happened or are happening, so that we can understand the security threats and risks and thus be better prepared for future attacks.

A RDBMS in which data is stored in tables and the relationships among the data are also stored in tables. The data can be accessed or reassembled in many different ways without having to change the table forms. For example, student, banking database etc. An intruder in the database system might be viewed as a malicious agent that tries to violate the security policy.

Recent years have witnessed a significant increase in the number of vulnerabilities in database system. Vulnerabilities in database system can be exploited by attackers to obtain unauthorized access to data stored database systems or to illegally execute malicious commands on host computers. Obtaining sensitive data is the real intension of the intruders. And another major challenging in the organization is to prevent the theft of data from the outsider as well as insider of the organization. Therefore, Intrusion Detection (ID) is the most critical technique for observing the database to detect the potential intrusion and take appropriate action on that anomalous request.

II. RELATED WORK

Many approaches for dealing with intrusion in OS and Network have been proposed and worked on. But these methods may not be appropriate for a Database System.

An anomaly detection system for relational databases is proposed by Spalka et al. [7]. This work focuses on detecting anomalies in a particular database state that is represented by the data in the relations. Their first technique uses basic statistical functions to compare reference values for relation attributes being monitored for anomaly detection. The second technique introduces the concept of ¢ relations that record the history of changes of data values of monitored attributes between two runs of the anomaly detection system. This work focuses on the semantic aspects of the SQL queries by detecting anomalous database states as represented by the data in the relations, while we focus detecting anomalous access patterns in a DBMS.

An Intrusion Detection mechanism has been proposed in [3]. The approach is based on mining the SQL queries stored in database audit log files. The result of the mining process is used to form profiles that can model normal database access behavior and identify intruders.

The problem of issuing an appropriate response to a detected database anomaly has been discussed in [5]. The work presented here is an extension of the work done. Also the proposed JTAM frame work would be used here to incorporate security from internal anomalous requests.

III. PROPOSED SCHEME

The underlying reasons for using intrusion detection systems are relatively straightforward: protect data and maintain database integrity.

ID mechanism consists of two main elements, an Anomaly Detection (AD) system and Anomaly Response system.

Anomaly: Anomaly is characterized when a user request doesn't conform to the normal access profile. Information of various levels of details is recorded in the profiles.

The AD is based on the construction of database access profiles of roles [8] and users and that are used for the AD task. A user request that does not match to the normal access profiles is detected as anomalous. The second element is taking proper action, once the anomaly is detected. There are three main response actions: low level actions like sending an alert, whereas the high security actions can effectively block the anomalous request, middle level response actions may suspend or taint an anomalous user from further using the services of the database system.

Architecture of proposed system:

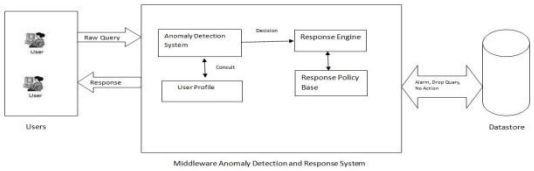

Figure 1: Proposed Architecture

The proposed system will be designed and implemented in the following manner,

3.1 Anomaly Detection (AD) system

3.2 Anomaly Response system.

3.3 Joint Administration model

3.1: Anomaly Detection (AD) system

The detection of an anomaly by the detection engine can be considered as a system event. The attributes of the anomaly, such as user, role, SQL command, then correspond to the environment surrounding such an event. A policy can be specified taking into account the anomaly attributes to guide the response engine in taking a suitable action. The anomaly detection mechanism provides its assessment of the anomaly using the anomaly attributes.

We propose a role based approach for detection of anomaly activity. In this we use the profiles which describe the typical behavior of the role that the user is working with the database. Authorizations are specified with respect to roles and not with respect to individual users. One role is assigned to each user and privileges are assigned to roles. Our AD system uses this profile for each role to determine role anomalies, that is, individuals that while holding a specific role deviate from the normal behavior of that role.

3.2 Anomaly Response system

The database request that has been detected as anomalous has to be given proper response. The response to anomalous request is to be given by the Anomaly Response system.

There are three proposed categories of response policies namely:

Conservative: low security

Fine-Grained: medium security

Aggressive: high security

The Low security (i.e. Conservative) level users are allowed any number of anomalous queries without

discontinuing the session. In case of Medium security (i.e. Fine-Grained) level implies that the users with this role will be logged out after fixed number of anomalous queries in a fixed amount of time. In case of High security (i.e. Aggressive) level after a fixed number of anomalous queries in fixed time quantum the user will be logged off and blocked from using the system.

3.3 Joint Administration model

The single administrator model used by the DBMS today risks the system to be endangered by the lone administrator. This risk is probably neutralized by using the joint model. The proposed system has n number of administrators. If a new policy is to be created then it has to be approved by l out of the n administrators where l is atleast ((n/2)+1).

IV. Work done so far

4.1 Anomaly Detection (AD) system

In order to identify the anomalies we first have to identify them. The following list enlists the type of anomalies that can be monitored in this system.

List of Anomalies:

Select anomaly: When a user is given the rights using the grant option on a specific relation the user gets the right to view the whole relation and thus the internal user may get the details which he/she should not actually be able to see. Although the Mandatory Access Control mechanism of the database security can be applied in this case but most of the DBMS packages in the market today do not support this mechanism leaving the data to be venerable to be viewed by internal users.

Time based: If we provide an operation for controlling the maximum number of tuples in a query still the risk of the user firing the queries based on specific selection to get the required number of tuples that they want. To restrain this sort of behavior provision should be made to monitor the maximum number of queries the user can fire in a specified time period.

Insert anomaly: The user that tend to harm the database generally use the strategy of filling up the relation with unwanted data or duplicated data so as to make the database useless thereafter. The grant command of the database that provides for insert privileges does not control this behavior.

Delete anomaly: The most common anomaly with database from insiders is that of deletion of the relational data that they have delete privileges to. The data once deleted from the relation is difficult to recollect and may cause serious loss of information. This can be controlled by monitoring the number of tuples affected by the delete query of the privileged user and then allowing it to work only if dose not violet the predefined policy.

The use of roles makes our approach usable even for databases with a large user population. Managing a few roles is much more efficient than managing many individual users.

Having identified the possible anomalies one needs to identify the attributes using which the anomalies can be identified by the system. Below is the list of possible anomaly attributes.

Number of queries submitted in unit time
IP address assignment
Number of concurrent connections
Creation of table rights
Modification of table (Alter table query) rights
Bulk delete operation
Truncate table
Drop table query
Bulk insert
Update the table record (Update query).

The anomaly attributes now are given the values according to the policy decisions for usage of the database.

Once the policy has been specified it can be assigned to the role and thus to the users that are to use the database application. A defined policy can be attached to various users. But every user must be assigned at least one policy object. Figure 2 demonstrates the panel for defining the values of the policy attributes.

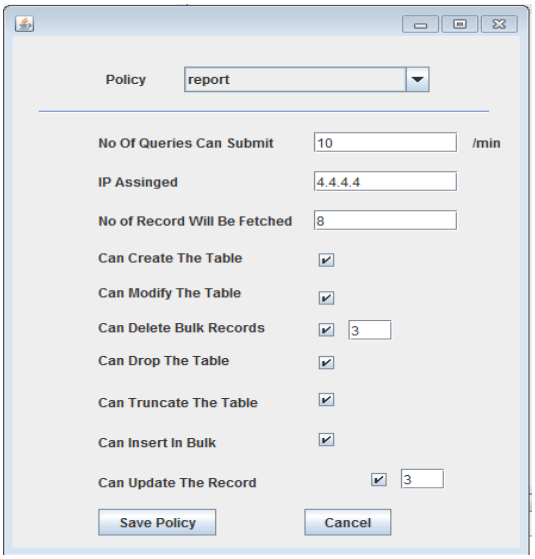

Figure 2: Panel for defining values of policy attributes.

The policy creation panel has a field to assign the maximum number of tuples that can be selected in a single query by the user with the specified role.

The third attribute given in the window provides the option for setting the maximum number of records that can be fetched in a single query. Also the 6th and 10th attributes of the panel are maximum number of tuples that can be deleted and updated in a single SQL query.

The values of the attributes are stored in the policy data store.The user interface for the client is shown in figure 3.

The client interface provided shows the policy values set or reset according to the role with the flag set or reset. The query message box allows the user to put up query while the output box allows for viewing of the query result.

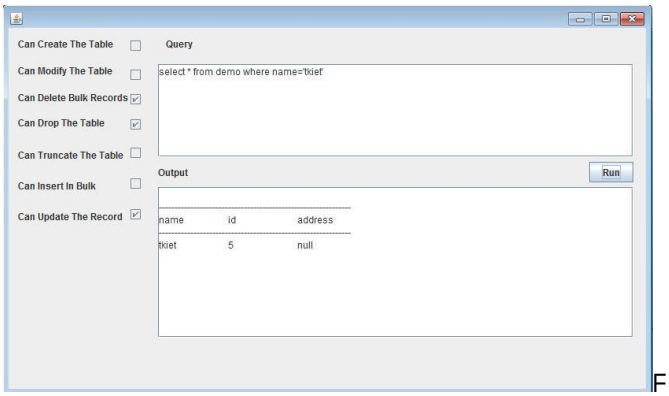

Figure 3: Client Interface

The working of the system starts by taking the input query and comparing the same with the values set for the role of the user that is issuing the query. If the rules of the role are followed than the result is displayed in the output window or the message with respective error is displayed.

The implementation of different policy attributes are discussed next.

Number of tuples to be selected:
When the user with the specified role logs in to the system the policy attribute for the user is imported to the client machine so as to reduce the cost of transporting each anomalous query to the server. Once a query is submitted for execution it is first scanned for its possibly anomalous execution at the client side.
If the query is select query it is operation which is valid for all users so the query is sent for execution at the server side for execution. But as has been studied if the user gets the right to view all the data it can be misused, so to constrain this anomalous misuse of the data the query is again evaluated at the server end against the policy object.
If the query being executed results in less than the specified number of tuples for the given user then the query result is shown to the user. If the user query results in more than the specified number of tuples than the query result is not shown to the user displaying a error message instead as demonstrated in figure 4.

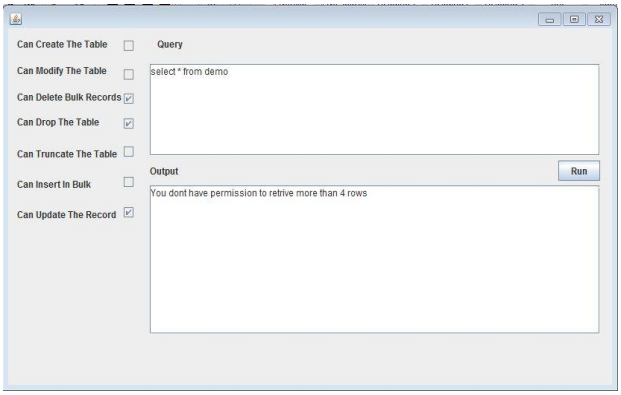

Figure 4: Result of Select Query if number of records exceeds satisfy policy.

Delete and Update operation:
Like the select operation the delete and update operations are also likely to produce unaccountable changes to the database if allowed to affect the database without monitoring the effects of the query after execution.
The policy attributes first identifies if the user is allowed to delete/update any tuples from the dataset by having the

admin to use the check box for the same. If the admin wants the policy such that the user can delete/update the data from the database the text box beside the check box allows the admin to specify the maximum number of tuples that can be deleted/updated from the dataset by the user in a single query.
Flag attribute values:
All the other attributes demonstrated are of Boolean type wherein the values are either set or reset shown the allowance of the operation for the user of the specified role.

Anomaly Response system
Having identified the anomalous queries the response policies play an important role in the overall system as it is the work of response system to appropriately take actions against the users that fire the anomalous queries time and again.
The administrator while creating the role will have to assign the security policy to the role which will then be applied to all the users with the specified role. Although the admin is itself a user but this role has been kept apart from the security policy as the admin controls the overall system and is also responsible for the proper working of the system. All the other user will be abide to the security policy assigned to the role.
There are three categories of response policies namely:
Conservative: low security
Fine-Grained: medium security
Aggressive: high security

Conservative: low security
The users with conservative security applied to the role will be able to fire queries that are monitored by the anomaly system with only restriction of the anomaly policy defined for the role. The numbers of anomalous queries that are fired by the users in this role are not monitored.
Figure 5 demonstrates the scenario of conservative response policy. As seen in the figure the logged user does not have the authority to modify more than 5 rows which happens to be the result of the entered query. So the query is not executed and the relevant message is displayed to the user. No other action is taken in case of any number of such anomalous queries being fired by the user with this response policy.

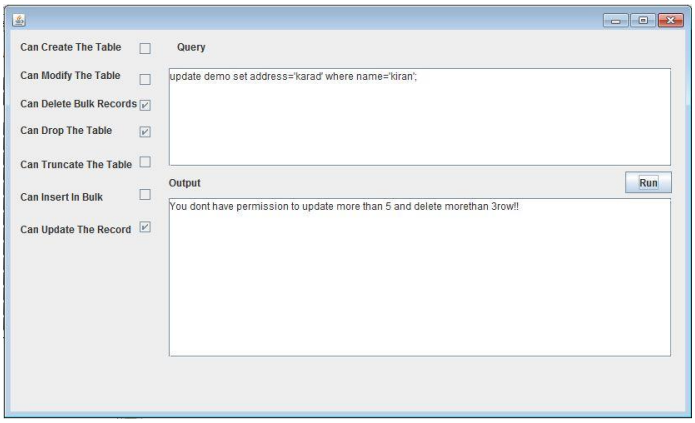

Fig. 5: Conservative Response Policy

Fine-Grained: medium security
With the Fine-Grained security the user will still see the same interface and work space as in case of the Conservative response policy. Here though the additional security constraints have been specified that the user can not execute more than specified number in specified amount of time.
If the users with the role that has medium security applied fires anomalous query less than the upper bound on anomalous number of queries they receive the message like in case of conservative approach. But if the user fires more than the threshold number of anomalous query the use will be logged off and the status at the server would be changed to blocked status. This is demonstrated in figure 6.

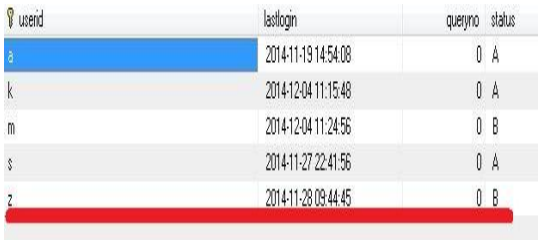

Fig. 6: Status change in Fine Grained policy
The user Z in figure 4 is the user that fired more than 5 anomalous queries which caused the status to change to B. The user status will be restored to A after predefined amount of time after which the user that was logged off will now be able to use the database service as earlier.
4.2.3 Aggressive: high security
The users with role assigned with Aggressive security level will be allowed only a limited number of anomalous queries in a time bound. If the user fires more than the specified number of anomalous queries the user will be logged out and blocked until the admin reinstates the user

to be able to use the database services. The implementation details are explained below.

4.3 Joint Administration Model

As proposed earlier the system works by having n number of administrator rather than a single admin as in the traditional systems. To apply a policy the rule of l out of n has been implemented. Where in for a policy to be implied to any role it has to be first approved by atleast l (considered as ((n/2) +1) in implementation) including the administrator that has created the policy.

The administrator can use the create policy option to create the policy. Once the policy has been created and saved the creator of the policy himself also has to approve the policy. The approve policy panel is shown in the figure 6.

The table in the system called jtm is used to store the log of userids that have voted for the given policy and the status that they have chosen by that user. Once the policy status is clear the entry in the jtm table for that policy is removed.

On using the option for approval of policy the administrator will see the panel shown in figure 6. The new policy drop down box shows only those policies that have not yet been processed by the current logged in admin. The lost focus event of the drop down box is used here to present the values of the policy that were used for defining the policy. At the end of panel page the need

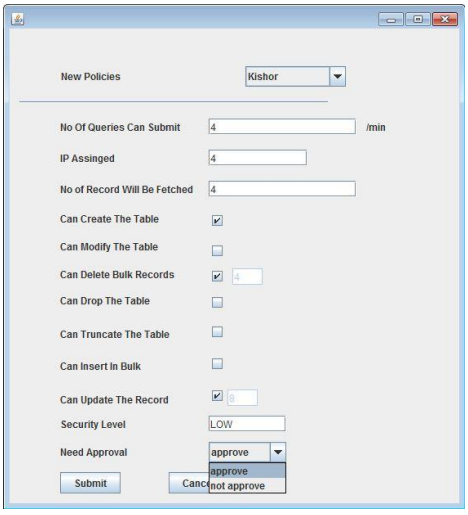

Figure 6: Approve Policy Panel

approval drop down box is shown with the option for approving or disapproving the chosen policy. If the current admin approves or disapprove the policy the status is accordingly saved in the jtm table.

The count of administrator in the policytouser table is used to check if the necessary condition for approval or disapproval of the policy is reached. According to the satisfied condition the status of the policy is kept 0 (inactive) or 1 (active).

V. CONCLUSION:

This paper has proposed a new mechanism to detect and respond to malicious data access. As database systems play a vital role in organizational information architectures, procedures must be in place to ensure that these resources are not being used maliciously. We have presented the concepts and underlying architecture and shown how they can be applied.

The response component is responsible for issuing a suitable response to an anomalous user request. We proposed the notion of database response policies for specifying appropriate response actions.

The paper also proposes a Joint Administration Model inorder to prevent the policy administration from being monopolized by a single administrator as in the traditional database systems.

VI. REFERENCES

[1] Ashish Kamra, Elisa Bertino "Design and Implementation of an Intrusion Response System for Relational Databases" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 6, JUNE 2011.

[2] Tran Khanh Dang, Thieu Hoa Le, Duy Tin Truong "An Extensible Framework for Database Security Assessment and Visualization" Proceedings of iiWAS2007

[3] Ashish Kamra · Evimaria Terzi · Elisa Bertino "Detecting anomalous access patterns in relational databases" The VLDB Journal (2008) 17:1063–1077

[4] Raji V, Ashokkumar P "Protecting Database from Malicious Modifications Using JTAM" Journal of Computer Applications

[5] Ashish Kamra, Elisa Bertino, Rimma Nehme "Responding to Anomalous Database Requests" SDM 2008, LNCS 5159, pages 50–66, Springer-Verlag Berlin Heidelberg 2008

[6] Michael Kirkpatrick, Elisa Bertino "An Architecture for Contextual Insider Threat Detection"

[7] A. Spalka and J. Lehnhardt. "A comprehensive approach to anomaly detection in relational databases" In DBSec, pages 207-221, 2005

[8] Mohammed Masoud Javidi, Marjan Kuchaki Rafsanjani, Sattar Hashemi and Mina Sohrabi1. " An overview of anomaly based database intrusion detection systems" Indian Journal of Science and Technology

