Efficient Implementation of Insider and Outsider Activity Response System for RDB

[1]Mr.Mallikharjuna Rao Jonnalagadda, [2]Mrs.G.K.Sandhia

[1]PG Scholar, [2]Assistant Professor
Department of Computer Science, SRM University, Chennai, Tamil nadu, India
[1]jmalli1arjun@gmail.com, [2]ksandhia@gmail.com

Abstract : In every organization data is crucial part. Organization have shown interest in DB monitoring and intrusion detection. Every organization protect data from the outsider attacks by using different security techniques, such as authentication, access permissions. But detecting the insider attacks on the database is difficult. To protect data from insider attack this paper provides a approach on a DBS is called multi hand administration. Policy languages are used to DBA detecting the anomalous request and take the appropriate actions based on the nature of anomaly request. To support intrusion response system these proposed database response policies are adapted to DBMS. The two main issues that are considered in this system is policy matching and policy administration. For policy matching propose two algorithms they effectively search the policy DB for policies that alike an anomalous request. The other problem is administration of response policies to avoid the malicious changes to policy objects from valid users. To avoid policy administration problem a novel Joint Threshold Administration Model (JTAM) is used. JTAM is based on the standard of partition of duty. The main idea following the JTAM is policy object is together administrated by at least k-DBAs, if any modification done to the policy object is invalid except it authorized by minimum k-DBAs.JTAM is based on the Cryptographic threshold signature scheme. HMAC is use to provide Integrity to the data.

***Index Terms*— Intrusion detection, Intrusion response system, HMAC, DataBase, Threshold Signatures**

I. INTRODUCTION

Organizations have recently shown interest in monitoring database system and report any relevant suspicious activity[2]. Database activity monitoring has been identified as one of the top five strategies that are crucial for reducing data leak in organizations[3],[4]. Monitoring database to detect potential intrusions, intrusion detection (ID), is a crucial technique that has to be part of any comprehensive security solution for high-assurance data-base security. An ID mechanism consists of two main elements, specifically tailored to a DBMS: an anomaly detection (AD) system and an anomaly response system. The first element is based on the construction of database-access profiles of roles and users, and on the use of such profiles for the AD task. A user-request that does not conform to the normal access profiles is characterized as anomalous. The second element is in charge of taking actions once an anomaly is detected. There are three main types of response actions, such as conservative actions, fine-grained actions, and aggressive actions. The conservative actions, such as sending an alert, allow the anomalous request to go through, whereas the aggressive actions can effectively block the anomalous request[8][9]. Fine-grained response actions are neither conservative nor aggressive.

II. RESPONSE POLICY LANGUAGE

The detection of an anomaly by the detection engine can be considered as the system event. An Event-Condition-Action (ECA) language is used for specifying response policies. An ECA rule is typically organized as follows:

ON {Event} IF {Condition} THEN {Action}

If the event arises and the condition evaluates to true, the specified action is executed. In this context, an event is the detection of an anomaly by the detection engine. A condition is specified on the attributes of the detected anomaly and on the attributes representing the internal state of the DBMS. An action is the internal response action executed by the engine.

Anomaly Attributes: The anomaly detection mechanism provide its assessment of the anomaly using the anomaly attributes. There are two main categories of those attributes.1.Contextual category includes all attributes describing the context of the anomalous request such as user, roles, source and time. 2. Structural category includes all attributes conveying information about the structure of the anomalous request such as SQL command and accessed database objects.

Policy conditions: A response policy is a conjunction of atomic predicates where each predicate is specified against a single

anomaly attribute. Example of such predicates is given below:

Role	!= DBA
Source IP	IN 127.0.0.1
Objs	IN {dbo.*}

Response Actions:

Once a database request has been flagged off as anomalous, an action is executed by the response system to address the anomaly. The response action to be executed is specified as part of a response policy. Details concerning these attributes are reported in Table 1. The detection engine submits its characterization of the anomaly using the anomaly attributes. Therefore, the anomaly attributes also act as an interface for the response engine, thereby hiding the internals of the detection mechanism[5]. Note that the list of anomaly attributes provided here is not exhaustive. Our implementation of the response system can be configured to include/exclude other user-defined anomaly attributes.

TABLE 1
Anomaly Attributes

ATTRIBUTE	DESCRIPTION
CONTEXTUAL	
USER	The user associated with the request.
ROLE	The role associated with the request.
CLIENT APP	The client application is associated with the request.
Source IP	The IP address is associated with request.
Data Time	Data/Time of the anomalous request.
STRUCTURAL	
Database	The database referred to in the request.
Schema	The database referred to in the request.
Obj Type	The schema referred to in the request such as table, view, stored procedure.
Obj(s)	The object name(s) referred in the request.
SQLCmd	The SQL Command associated with the request.
Obj attr(s)	The attributes of the object(s) referred in the request.

III. RESPONSE POLICIES

A response policy condition is a conjunction of predicates where each predicate is specified against a single anomaly attribute. Note that to minimize the overhead of the policy matching procedure (cf. Section 4), we do not support disjunctions between predicates of different attributes such as SQL Cmd = "Select" OR "IP Address" = "127.0.0.1." However, disjunctions between predicates of the same attribute are still supported. For example, if an administrator wants to create a policy with the condition SQL Cmd = "Select" OR SQLCmd = "Insert"; such condition can be supported by our framework by specifying a single predicate as SQL Cmd IN {"Select", "Insert"}. More exam-ples of such predicates are given below:

TABLE 2
Response Policy Examples

Policy 1 ON ANOMALY DETECTION IF Role !=DBA and Obj Type=table and Objs IN dbo.* and SQLCmd IN{Select} THEN DISCONNECT
Policy 2 ON ANOMALY DETECTION IF Role !=DBA and source IP IN 127.0.0.1 and Date Time BETWEEN 0800-1700 THEN LOPP

Table 2 describes two response policy examples. The threat scenario addressed by Policy 1 is as follows: In many cases, the database users and applications have read access to the system catalogs tables by default. Such access is sometimes misused during a SQL Injection attack to gather sensitive information about the DBMS structure. An anomaly detection engine will be able to catch such requests, since they will not match the normal profile of the user. Suppose if the DBMS is to be protected from anomalous reads to the system catalogs("dbo" schema) from unprivileged database users. Policy one aggressively prevents against such attacks by disconnecting the user. Policy two prevents the false alarms originating from the privileged users during usual business hours. The policy is formulated to take no action on any anomaly that originates from the internal network of an organization from the privileged users during normal business hours.

IV. LIFECYCLE OF A RESPONSE POLICY OBJECT

The steps in the lifecycle of a policy object are policy creation, activation, suspension, alteration and deletion.

- *Policy Activation* : Once the policy has been created, it must be authorized for activation by at least k À 1 administrators after which the DBMS changes the state of the policy to ACTIVATED. The policy activation command has the following format: Authorize Response Policy ١⁄2 Policy IDŠ Create.
- *Policy Suspension* :To alter/drop a policy or to make it nonoperational, the policy state must be changed to SUSPENDED. To change the policy state to SUSPENDED, an administrator issues the Suspend Response Policy [Policy ID] command.
- *Policy Alteration* :An administrator can alter a policy in the SUSPENDED state by executing the Alter Response Policy [Policy ID] [Policy Data] command.
- *Policy Drop* :A response policy is dropped by executing the Drop response Policy [Policy ID] command. The sequence of steps performed to drop a policy is similar to the steps performed or policy suspension.

V. POLICY ADMINISTRATION

The main issue in the administration of response policies is how to protect a policy from malicious modifications made by a DBA that has legitimate access rights to the policy object. To address this issue, an administration model referred to as the JTAM is proposed. It protects a response policy against malicious modifications by maintaining a digital signature on the policy definition[13]. The signature is then validated either periodically or upon policy usage to verify the integrity of the policy definition. One of the key assumptions in JTAM is that it is not possible to assume the DBMS to be in possession of a secret key for verifying the integrity of policies. If the DBMS had possessed such key, it could simply create a HMAC Hashed Message Authentication Code) of each policy using its secret key, and later use the same key to verify the integrity of the policy. However, management of such secret key is an issue since it cannot be assumed the key to be hidden from a malicious DBA. The fundamental premise of the approach is that it is not feasible to trust a single DBA (with the secret key) to create or manage the response policies, but the threat is mitigated if the trust (the secret key) is distributed among multiple DBAs. This is also the fundamental problem in threshold cryptography, that is, the problem of secure sharing of a secret. JTAM is used on a threshold cryptographic signature scheme[14].

HMAC

In cryptography, a keyed-hash message authentication code (HMAC) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. As with any MAC, it may be used to simultaneously verify both the *data integrity* and the *authentication* of a message. Any cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA1 accordingly. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key.An iterative hash function breaks up a message into blocks of a fixed size and iterates over them with a compression function. For example, MD5 and SHA-1 operate on 512-bit blocks. The size of the output of HMAC is the same as that of the underlying hash function (128 or 160 bits in the case of MD5 or SHA-1, respectively), although it can be truncated if desired

VI. JTAM SETUP

The main issue is essentially that of insider threats, that is, how to protect a response policy object from malicious modifications made by a database user that has legitimate access rights to the policy object. To address this issue, an administration model that is based on the well-known security principle of separation of duties (SoD) is proposed. It is a principle whereby multiple users are required in order to complete a given task. As a security principle, the primary objective of SoD is prevention of fraud (insider threats), and user generated errors. Such objective is traditionally achieved by dividing the task and its associated privileges among multiple users. However, the approach of using privilege dissemination is not applicable in this case as DBAs possess all possible privileges in the DBMS. This approach instead applies the technique of threshold cryptography signatures to achieve SoD[14]. A DBA authorizes a policy operation, such a create or drop, by submitting a signature share on the

policy. At least k signature shares are required to form a valid final signature on a policy, where k is a threshold parameter defined for each policy at the time of policy creation[13]. The final signature is then validated either periodically or upon policy usage to detect any malicious modifications to the policies. The key idea in this approach is that a policy operation is invalid unless it has been authorized by at least k DBAs. The model is thus referred to as the Joint Threshold Administration Model (JTAM) for managing response policy objects. The three main advantages of JTAM are as follows: First, it requires no changes to the existing access control mechanisms of a DBMS for achieving SoD. Second, the final signature on a policy is non repudiable, thus making the DBAs accountable for authorizing a policy operation[13]. Third, and probably the most important, JTAM allows an organization to utilize existing man-power resources to address the problem of insider threats since it is no longer required to employ additional users as policy administrators.The main contributions of this paper can be summarized as follows:

- A framework for specifying intrusion response policies in the context of a DBMS.
- A novel administration model called JTAM for administration of response policies.
- Algorithms to efficiently search the policy database for policies that match an anomalous request.

VII. POLICY MATCHING

There are two variations of policy matching algorithm.

A. Base Policy Matching

The policy matching algorithm is invoked when the response engine receives an anomaly detection assessment. For every attribute A in the anomaly assessment, the algorithm evaluates the predicates defined on A. After evaluating a predicate, the algorithm visits all the policy nodes connected to the evaluated predicate node. A policy is matched when its predicate-match-count becomes equal to the number of predicates in the policy condition. On the other hand, if the predicate evaluates to false, the algorithm marks the connected policy nodes as invalidated. For every invalidated policy, the algorithm decrements the policy-match-count9 of the connected predicates; the rationale is that a predicate need not be evaluated if its policy-match-count reaches zero.

B. Ordered Policy Matching

The search procedure in the base policy matching algorithm does not go through the predicates according to a fixed order. A heuristic by which the predicates are evaluated in descending order of their policy-count is introduced; the policy-count of a predicate being the number of policies that the predicate belongs to. Such heuristic is referred to as the Ordered Policy Matching algorithm. The rationale behind the ordered policy matching algorithm is that choosing the correct order of predicates is important as it may lead to an early termination of the policy search procedure either by invalidating all the policies or by exhausting all the predicates. Note that the sorting of the predicates in decreasing order of their policy-count is a pre-computation step which is not performed during the runtime of the policy matching procedure

VIII. MODULES

1. Authentication module
2. Data process
3. Security generation
4. Data verification

Authentication module:

This module is to register the new users and previously registered users can enter into our project. The Register user only can enter into Proposed Process in our Project. The Other user can view Existing Of our Project

- Data Process: In this module user can able to identify the data that means the over all content of the database analyzed and viewed by the authorized user.
- Security generation: The notion of database response policies for specifying appropriate response actions performed in this module the Joint Threshold Administration Model (JTAM) used to secure a data content in relational database.
- Data verification: Data verification performed using already stored database values. once again Joint Threshold Administration Model performed and data once verified with already presented data.

IX. SYSTEM ARCHITECTURE

Figure 1 is the architecture diagram for this paper for multi hand administration and intrusion avoidance. For each user, Detection engine compares the user query with the profile information maintained for the respective user. If an anomaly is detected the response engine will provide a response to the user based on the type of data the user is trying to access.In Figure 1 Shows how the this project work. Initially user enter the Query request checks the user is authorised user or not.If user is not a authorised Access is denied.if user is Authorised user based on query check the policy by using policy administration based on

JTAM.

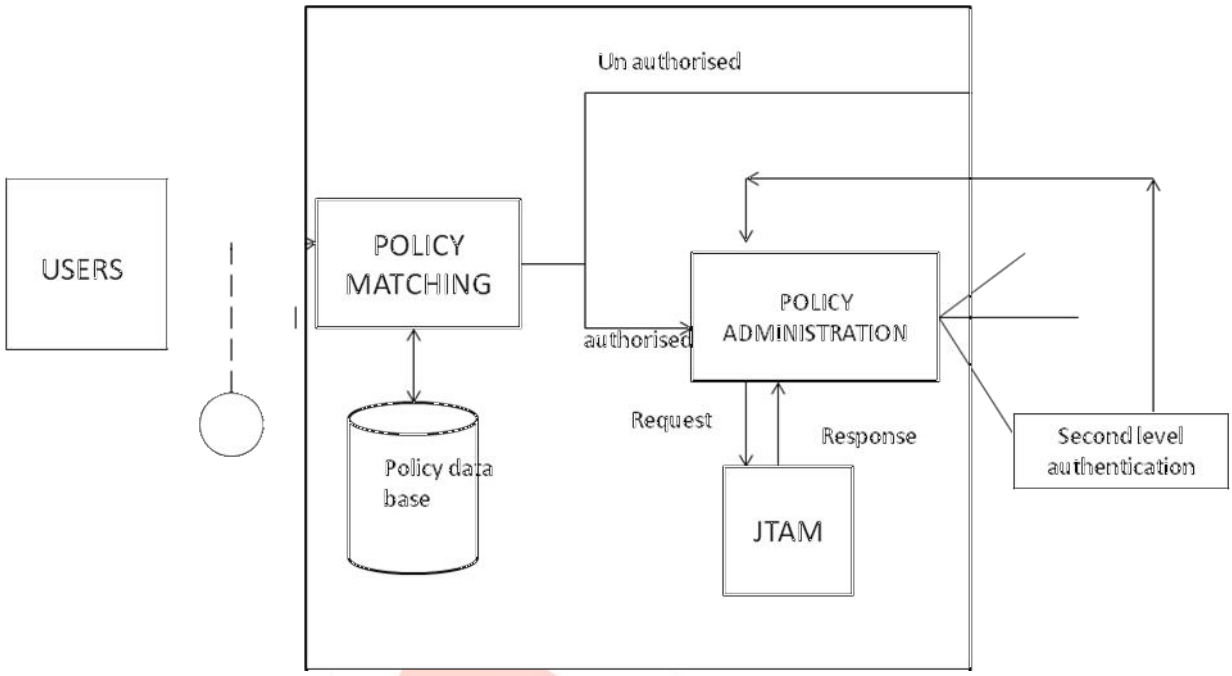

Figure 1: *Architecture Diagram*

Experimental Evaluation

The goal of the experimental evaluation is to measure the overhead incurred by the base policy matching, and the ordered policy matching algorithms. Also report experimental results on the overhead of the signature verification scheme in JTAM[13]. In what follows, first describe the experimental setup, and then report the evaluation results.Here use the following six anomaly attributes for our experimental evaluation: *User*, *Client App*, *Source IP*, *Database*, *Objs*, and *SQLCmd*. The predicate generation code randomly assigns set-valued data to these anomaly attributes to create the policy predicates. The policy generation code randomly assigns such predicates to policy conditions to create the policies. Figure 2 shows the policy matching overhead for the two algorithms as a function of the number of predicates.

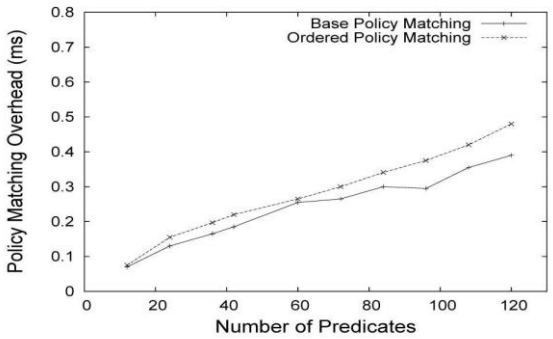

Figure 2*: Number of predicates versus policy matching overhead.*

X. CONCLUSION

In this paper, described the response component of our intrusion detection system for a DBMS. 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. We presented an interactive Event-Condition-Action type response policy language that makes it very easy for the database security administrator to specify appropriate response actions for different circumstances depending upon the nature of the anomalous request. The two main issues that we addressed in the context of such response policies are *policy matching*, and *policy administration.* For the policy matching procedure, we described algorithms to efficiently search the policy database for policies matching an anomalous request assessment. The experimental evaluation of our policy matching algorithms showed that our techniques are efficient. The other issue that we addressed is the administration of response policies to prevent malicious modifications to policy objects from legitimate users.

REFERENCES

[1] Ashish Karma and Elisa Bertino, "Design and Implementation of an Intrusion Response System for Relational Databases

(June. 2011)," http://www.networkcomputing.com/showArticle.
[2] A. Conry-Murray, "The Threat from within. Network Computing (Aug. 2005)," http://www.networkcomputing.com/showArticle. jhtml?articleID=166400792, July 2009.
[3] R. Mogull, "Top Five Steps to Prevent Data Loss and Information Leaks. Gartner Research (July 2006)," http://www.gartner.com, 2010.
[4] M. Nicolett and J. Wheatman, "Dam Technology Provides Monitoring and Analytics with Less Overhead. Gartner Research (Nov. 2007)," http://www.gartner.com, 2010.
[5] R.B. Natan, Implementing Database Security and Auditing. Digital Press, 2005.
[6] D. Brackney, T. Goan, A. Ott, and L. Martin, "The Cyber Enemy within ... Countering the Threat from Malicious Insiders," Proc. Ann. Computer Security Applications Conf. (ACSAC). pp. 346-347, 2004.
[7] A. Kamra, E. Terzi, and E. Bertino, "Detecting Anomalous Access Patterns in Relational Databases," J. Very Large DataBases (VLDB), vol. 17, no. 5, pp. 1063-1077, 2008.
[8] A. Kamra, E. Bertino, and R.V. Nehme, "Responding to Anomalous Database Requests," Secure Data Management, pp. 50- 66, Springer, 2008.
[9] A. Kamra and E. Bertino, "Design and Implementation of SAACS: A State-Aware Access Control System," Proc. Ann. Computer Security Applications Conf. (ACSAC), 2009.
[10] "Postgresql 8.3. The Postgresql Global Development Group," http://www.postgresql.org/, July 2008.
[11] J. Widom and S. Ceri, Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, 1995.
[12] "Oracle Database Concepts 11g Release 1 (11.1)," http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/datadict.htm, July 2009.
[13] V. Shoup, "Practical Threshold Signatures," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), pp. 207-220, 2000.
[14] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk, "Robust and Efficient Sharing of RSA Functions," J. Cryptology, vol. 20, no. 3, pp. 393-400, 2007.
[15] Kincaid and W. Cheney, Numerical Analysis: Mathematics of Scientific Computing. Brooks Cole, 2001.
[16] "Openpgp Message Format. rfc 4800," http://www.ietf.org/rfc/ rfc4880.txt, July 2009.
[17] K. Koc, "High-Speed RSA Implementation," Technical Report tr-201, Version 2.0, RSA Laboratories, 1994

Author Profile

Mr.MallikharjunaRao.J received the B.Tech in Computer Science and Engineering from Gudlavalleru Engineering college, Andhra Pradesh, India in 2012. He is currently pursuing the M.Tech. in C.S.E. from SRM University. His current research interests include DataBase security and Image Processing.

Mrs.G.K.Sandhia received the B.E.in Information Technology from Anna University, Tamil Nadu and M.E.in Computer Science and Engineering from Anna University, Tamil Nadu, India. She is currently working an Assistant Professor in the Department of CSE at SRM University. Her research interest includes Cyber security and Network Security.

.