International Journal Of Advanced Research and Innovations Vol.1, Issue .1
ISSN Online: 2319 – 9253
Print: 2319 – 9245

DATABASE INTRUSION DETECTION RESPONSE SYSTEM

T.Swetha[1]   B.KRISHNA PRASAD[2]   P.V.KUMAR[3]

1. (M.TECH, CSE), BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY
2. Associate Professor(CSE), BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY
3. HOD(CSE), BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

ABSTRACT

We present the findings of my survey on intrusion detection system. The intrusion detection system is responsible for issuing a suitable response to request. We describes the database response policies to support our intrusion response system. it very easy for the database administrators to specify appropriate response actions for different circumstances depending upon the nature of the anomalous request. We mainly focuses on two issues that policy matching, and policy administration. We also extend the PostgreSQL DBMS with our policy matching mechanism, and report experimental results.The experimental evaluation shows that our techniques are very efficient. The other issue that we address is that of administration of response policies to prevent malicious modifications to policy objects from legitimate users.

Index Terms—Databases, intrusion detection, response, prevention, policies, threshold signatures.

I. INTRODUCTION

Database activity monitoring has been identified by Gartner research as one of the top five strategies that are crucial for reducing data leaks in Organizations. Such step-up in data vigilance by organizations is partly driven by various US government regulations concerning data management such as SOX, PCI, GLBA, HIPAA, and so forth. Organizations have also come to realize that current attack techniques are more sophisticated, organized, and targeted than the broad-based hacking days of past. Often, it is the sensitive and proprietary data that is the real target of attackers. Also, with greater data integration, aggregation and disclosure, preventing data theft, from both inside and outside organizations, has become a major challenge.

Standard database security mechanisms, such as access control, authentication, and encryption, are not of much help when it comes to preventing data theft from insiders Such threats have thus forced organizations to reevaluate security strategies for their internal databases. Monitoring a 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 database security. Note that the ID systems that are developed must be tailored for a Database Management System (DBMS) since database-related attacks such as SQL injection and data exfiltration are not malicious for the underlying operating system or the network.

A suspended request is simply put on hold, until some specific actions are executed by the user, such as the execution of further authentication steps. A tainted request is marked as a potential suspicious request resulting in further monitoring of the user and possibly in the suspension or dropping of subsequent requests by the same user.

II. Existing System

Organizations have also come to realize that current attack techniques are more sophisticated, organized, and targeted than the broad-based hacking days of past. Often, it is the sensitive and proprietary data that is the real target of attackers. Also, with greater data integration, aggregation and disclosure, preventing data theft, from both inside and outside organizations, has become a major challenge. Standard database security mechanisms, such as access control, authentication, and encryption, are not of much help when it comes to preventing data theft from insiders. Such threats have thus forced organizations to reevaluate security strategies for their internal databases. Monitoring a 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 database security.

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. Profiles can record information of different levels of details; we refer the reader to for additional information and experimental results. The second element of our approach the focus of this paper—is in charge of taking some actions once an anomaly is detected. There are three main types of response actions, that we refer to, respectively, 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. Fine-grained response actions, on the other hand, are neither conservative nor aggressive. Such actions may suspend or taint an anomalous request. A suspended request is simply put on hold, until some specific actions are executed by the user, such as the execution of further authentication steps. A tainted request is marked as a potential suspicious request resulting in further monitoring of the user and possibly in the suspension or dropping of subsequent requests by the same user.

III. PROPOSED SYSTEM

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 Attack. A user-request that does not conform to the normal access profiles is characterized as anomalous. Profiles can record information of different levels of details; we refer the reader to for additional information and experimental results. The second element of our approach— the focus of this paper—is in charge of taking some actions once an anomaly is detected. There are three main types of response actions that we refer to, respectively, 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. Fine-grained response actions, on the other hand, are neither conservative nor aggressive. Such actions may suspend or taint an anomalous request . A suspended request is simply put on hold, until some specific actions are executed by the user, such as the execution of further authentication steps. A tainted request is marked as a potential suspicious request resulting in further monitoring of the user and possibly in the suspension or dropping of subsequent requests by the same user.

Advantage in Proposed System:

- 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.

1. Policy Language

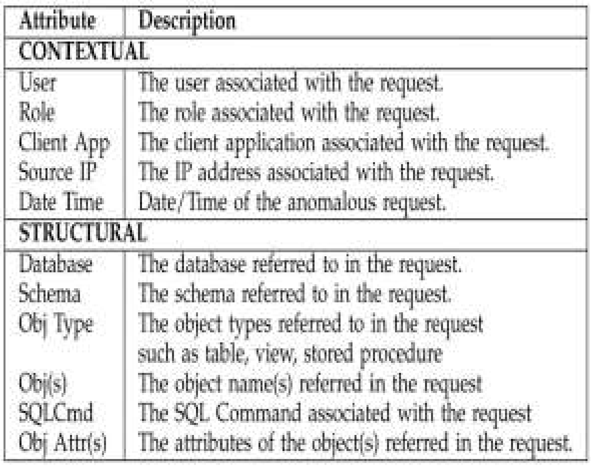
The detection of an anomaly by the detection engine can be considered as a system event. The attributes of the anomaly, such as user, role, SQL command, then correspond to the environment surrounding such an event. Intuitively, a policy can be specified taking into account the anomaly attributes to guide the response engine in taking a suitable action. Keeping this in mind, we propose an Event-Condition-Action (ECA) language for specifying response policies. Later in this section, we extend the ECA language to support novel response semantics. ECA rules have been widely investigated in the field of active databases [10]. An ECA rule is typically organized as follows:

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

As it is well known, its semantics is as follows: if the event arises and the condition evaluates to true, the specified action is executed. In our context, an event is the detection of an anomaly by the detection engine. A condition is specified on the attributes of the detected anomaly. An action is the response action executed by the engine. In what follows, we use the term ECA policy instead of the common terms ECA rules and triggers to emphasize the

2. Anomaly Attributes

The anomaly detection mechanism provides its assessment of the anomaly using the anomaly attributes. We have identified two main categories for such attributes. The first category, referred to as contextual category, includes all attributes describing the context of the anomalous request such as user, role, source, and time. The second category, referred to as structural category, includes all attributes conveying information about the structure of the anomalous request such as SQL command, and accessed database objects. 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. 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.



Attribute	Description
CONTEXTUAL	
User	The user associated with the request.
Role	The role associated with the request.
Client App	The client application associated with the request.
Source IP	The IP address associated with the request.
Date Time	Date/Time of the anomalous request.
STRUCTURAL	
Database	The database referred to in the request.
Schema	The schema referred to in the request.
Obj Type	The object types 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.

Table 3.1 anomaly attributes

3. Interactive ECA Response Policies

An ECA policy is sufficient to trigger simple response measures such as disconnecting users, dropping an anomalous request, sending an alert, and so forth. In some cases, however, we need to engage in interactions with users. As ECA policies are unable to support such sequence of actions, we extend them with a confirmation action construct. A confirmation action is the second course of action after the initial response action. Its purpose is to interact with the user to resolve the effects of the initial action. If the confirmation action is successful, the resolution action is executed, otherwise the failure action is executed. Thus, a response policy in our framework can be symbolically represented as follows:

ON {Event} IF {Condition}

THEN {Initial Action} CONFIRM {Confirmation Action}

ON SUCCESS {Resolution Action} ON FAILURE {Failure Action}

IV. Architecture:

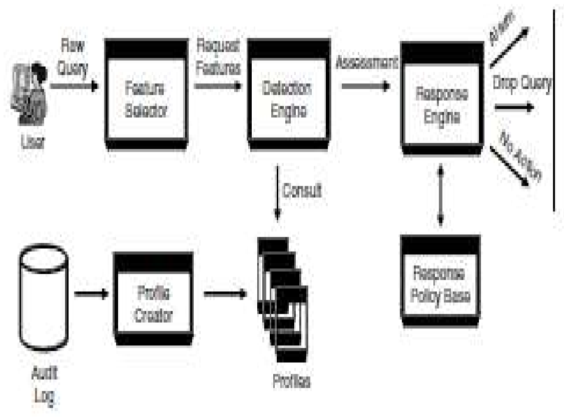


Fig: 3.3 System Architecture

V. APPROACHES

This project provides the maintenance of the data and to detect the anomaly actions. To those reasons we are developing this project with the following modules.

- User administration & Authentication
- Policy Administration
- Policy Creation
- Anomaly detection
- Anomaly Response System
- System Log & Access Information

1. User Administration & Authentication

The DBA is centrally responsible for DB and user maintenance. By default there is only one DBA. The DBA then creates other DBA's and users. These users are maintained with appropriate privileges or permissions. Only these users can login and access the database objects.

2. 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, we propose an administration model referred to as the JTAM. The threat scenario that we assume is that a DBA has all the privileges in the DBMS, and thus it is able to execute arbitrary SQL insert, update, and delete commands to make malicious modifications to the policies. Such actions are possible even if the policies are stored in the system catalogs.3 JTAM protects a response policy against malicious modifications by maintaining a digital signature on the policy definition. The fundamental premise of our approach is that we do not 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. Each DBA who floats a policy has the policy in cipher text. Only when the appropriate keys as associated with the other DBA's are provided they unlock the cipher text and then place their opinion or status on the policies. Only when a majority of DBA's approve the policy the policy can be associated with the created users. The policy also associates with a response action that has to be performed when an anomaly is detected.

3. JTAM Setup

Before the response policies can be used, some security parameters are registered with the DBMS as part of a onetime registration phase. The details of the registration phase are as follows: The parameter l is set equal to the number of DBAs registered with the DBMS. Such requirement allows any DBA to generate a valid signature share on a policy object, thereby making our approach very flexible. Shoup's scheme also requires a trusted dealer to generate the security parameters. This is because it relies on a special property of the RSA modulus, namely, that it must be the product of two safe primes. We assume the DBMS to be the trusted component that generates the security parameters.

For all values of k, such that 2 _ k _ l _ 1, the DBMS generates the following parameters:

Here the technique used is ascii key protection. For security purpose the ascii key system is used. For every character in the password the ascii values of each value are encrpted with the constant value 118.

epwd = epwd + (Chr(Asc(Mid(pwd1.Text, i, 1)) + 118))

4. Life Cycle of a Response Policy Object

The steps in the lifecycle of a policy object are policy creation, activation, suspension, alteration, and deletion. The lifecycle is shown in Fig. 1 using a policy state transition diagram. The initial state of a policy object after policy creation is CREATED. After the policy has been authorized by k -1 administrators, the policy state is changed to ACTIVATED. A policy in an ACTIVATED state is operational, that is, it is considered by the policy matching procedure in its search for matching policies. If a policy needs to be altered, dropped or made nonoperational, it must be moved to the SUSPENDED state. The transition from the ACTIVATED state to the SUSPENDED state must also be authorized by k-1 administrators, before which the policy is in the SUSPEND IN-PROGRESS state. Note that a policy in the SUSPEND IN-PROGRESS state is also considered to be operational. From the SUSPENDED state, a policy can be either moved back to the CREATED state or it can be moved to the DROPPED state. A single administrator can move a policy to the CREATED state from the SUSPENDED state, while a policy drop operation must be authorized by k-1 administrators (before which the policy is in the DROP IN-PROGRESS state).

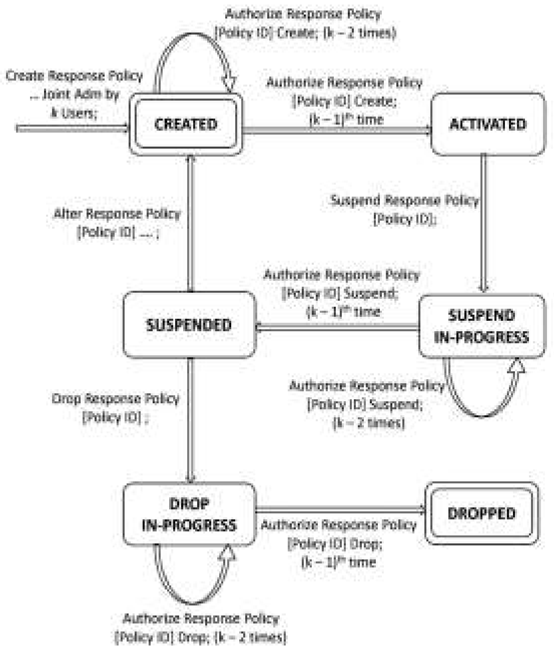

Authorize Response Policy [Policy ID] Create; (k – 2 times)
Create Response Policy ... Joint Adm by k Users;
CREATED
Authorize Response Policy [Policy ID] Create; (k – 1)th time
ACTIVATED
Alter Response Policy [Policy ID] ... ;
Suspend Response Policy [Policy ID];
Authorize Response Policy [Policy ID] Suspend; (k – 1)th time
SUSPENDED
SUSPEND IN-PROGRESS
Drop Response Policy [Policy ID] ;
Authorize Response Policy [Policy ID] Suspend; (k – 2 times)
DROP IN-PROGRESS
Authorize Response Policy [Policy ID] Drop; (k – 1)th time
DROPPED
Authorize Response Policy [Policy ID] Drop; (k – 2 times)

Fig 4.1: policy state transition diagram**.**

5. Policy Activation

Once the policy has been created, it must be authorized for activation by at least k - 1 administrators after which the DBMS changes the state of the policy to ACTIVATED. An activated policy can be assigned to a user. A policy identifies the objects and privileges the assigned user has on the object.

Before the response policies can be used, some security parameters are registered with the DBMS as part of a onetime registration phase. The details of the registration phase are as follows: The parameter l is set equal to the number of DBAs registered with the DBMS. Such requirement allows any DBA to generate a valid signature share on a policy object, thereby making our approach very flexible.

6. Anomaly Detection:

This 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 fundamental problem in such administration model is that of conflict-of-interest. 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.

- Anomaly:

A user-request that does not conform to the normal access profiles is characterized as anomaly.

- Anomaly Attributes:

7. Anomaly Response System:

This element is in charge of taking some actions once an anomaly is detected. There are three main types of response actions, that we refer to, respectively, 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. Fine-grained response actions, on the other hand, are neither conservative nor aggressive. Such actions may suspend or taint an anomalous request. A suspended request is simply put on hold, until some specific actions are executed by the user, such as the execution of further authentication steps.

A tainted request is marked as a potential suspicious request resulting in further monitoring of the user and possibly in the suspension or dropping of subsequent requests by the same user.

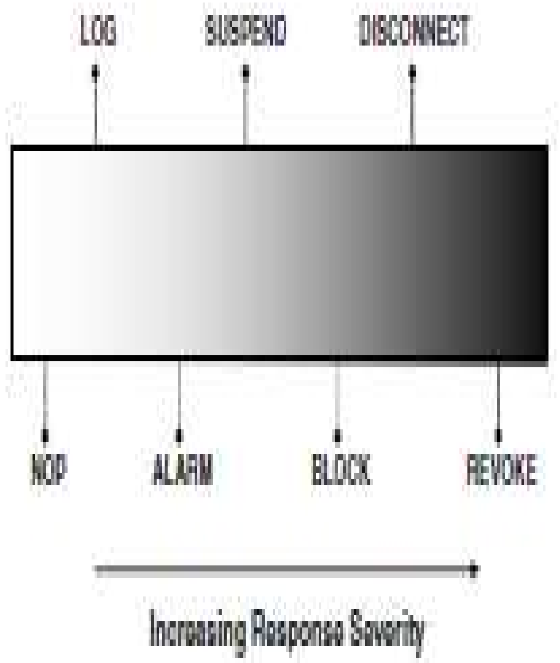
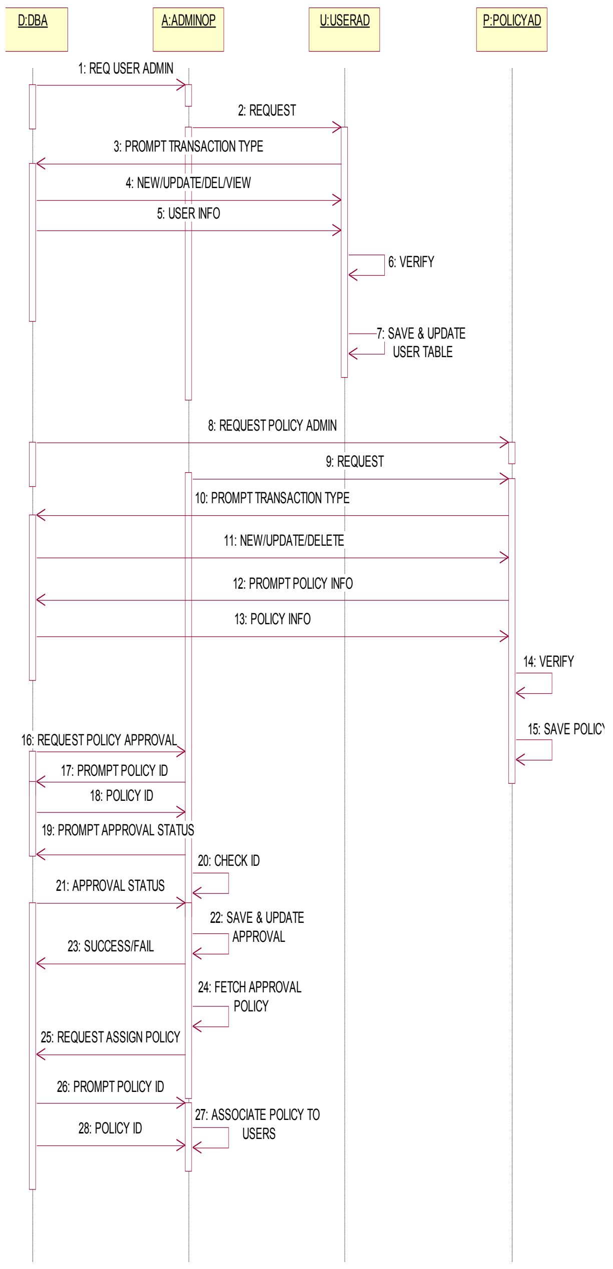


Fig: 4.2 Immediate Response Actions

8. System Log & Access Information:

This module provides the DBA with anomalies detected and reports various activities or attempted activities made by the users. The DBA uses this to generate various reports based on which an action or the revoke of privileges are made.

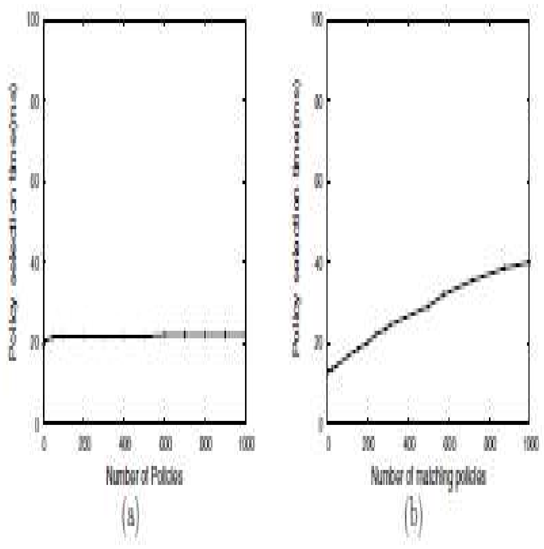
The fallowing sequence diagram shows the overall functionality of all users



VI. Experimental Results:

We begin with describing the experimental set-up. The experiments are performed on a Pentium dual-core processor machine with 2 GB RAM running openSUSE 10.3. The programming language used is PostgreSQL's procedural language i.e. pl/pgsql. For the DBMS installation, we create 10 databases.

We vary the number of policies keeping the anomaly size (i.e number of anomaly attribute values submitted by the detection engine) constant at 10. Note that the anomaly attribute values include the "Objs" attribute values. The size of the PCL is kept at 1000 implying that the system can support a maximum of 1000 policies. The results are shown in figure 6(a). The policy selection time is very low at approximately 20 ms. Moreover, it remains almost constant irrespective of the number of policies in the database. The reason is that the queries to PG PCL and PG SOURCE PCL tables are very efficient (due to indexing), while the bulk of the time of the policy selection algorithm is instead spent in obtaining the policy ids from the final PCL. This is because we currently use PostgreSQL's builtin bit-string datatype for storing the PCLs. We believe that the efficiency of the policy selection algorithm using PCLs can be further improved by a more efficient encoding of the PCLs.



VII. CONCLUSION

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.

This would enhance database monitoring and misuse of privileges assigned to users. Since the application provides support to K database administrator's security is enhanced. The possibility of an unwanted permission being assigned or excess permission being assigned can now be minimized or nullified.

REFERRENCES

[1] A. Conry-Murray, "The Threat from within. NetworkComputing(Aug2005) "http://www.networkcomputing.com/showArticle.jhtml?articleID=166400792, July 2009.

[2] R. Mogull, "Top Five Steps to Prevent Data Loss and Information Leaks. Gartner Research)," http://www.gartner.com, 2010.

[3] M. Nicolett and J. Wheatman, "Dam Technology Provides Monitoring and Analytics withLess Overhead. Gartner Research(Nov.2007)," Http://www.gartner.com, 2010.

[4] Raji V Ashokkumar P "Protecting Database from Malicious Modifications Using JTAM" February 2012.

[5] Ashish Kamra, Elisa Bertino, and Rimma Nehme " Responding to AnomalousDatabase Requests" Purdue University 2008

[6] 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.

[7] Kamra and E. Bertino, "Design and Implementation of SAACS:A State-Aware Access Control System," Proc. Ann. Computer Security Applications Conf.(ACSAC), 2009.

[8] The Postgresql Global DevelopmentGroup, "http://www.postgresql.org/, July 2008.

[9] J. Widom and S. Ceri, Active Database Systems: Triggers and Rules for Advanced DatabaseProcessing.

[10] Morgan Kaufmann "Oracle Database Concepts 11g Release http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/datadict.htm, July 2009.

[11] Shoup, "Practical Threshold Signatures," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), pp. 207-220, 2000.

[12] 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.

[13] Kincaid and W. Cheney, Numerical Analysis: Mathematics of Scientific Computing. Brooks Cole, 2001.