2009

Database Intrusion Detection Using Role Profiling

Zhiping Wu

Database Intrusion Detection Using Role Profiling

(Thesis format: Monograph)

by

Zhiping Wu

Graduate Program
in
Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

Insider threats cause the majority of computer system security problems and are also among the most challenging research topics in database security. An anomaly-based intrusion detection system (IDS), which can profile inside users' normal behaviors and detect anomalies when a user's behaviors deviate from his/her profiles, is effective to protect computer systems against insider threats since the IDS can profile each insider and then monitor them continuously. Although many IDSes have been developed at the network or host level since 1980s, there are still very few IDSes specifically tailored to database systems. We initially build our anomaly-based database IDS using two different profiling methods: one is to build profiles for each individual user (user profiling) and the other is to mine profiles for roles (role profiling). Detailed comparative evaluations between role profiling and user profiling are conducted, and we also analyze the reasons why role profiling is more effective and efficient than user profiling. Another contribution of this thesis is that we introduce role hierarchy into database IDS and remarkably reduce the false positive rate without increasing the false negative rate.

Keywords: Insider threats, Intrusion detection, RBAC, Database security, Role profiling.

Acknowledgements

First and foremost, I want to thank my family in China not only for their financial support, but more importantly, for their patience and always believing in me. Without them, this thesis would never have been possible.

I am also so grateful to my supervisor, Sylvia L. Osborn, for her suggestions, guidance and always being available when I had problems. Because of her endless support, my journey of studying at Western has become much easier.

Special thank you to Xin Jin. You opened the door, leading me into this research area, and passed through the early stage of the project together with me. Your advice has been very valuable in this research.

Last but not least, I would like to thank my friends in Canada, including but not limited to Jia Peng, Delei Weng, Ling Ding, Enxin Wu, Shuyi Feng, Hanlin Lv, Yue Zhang, Yang Xu and Jiaming You. We came to Canada together and spent a lot of time together here. Especially during the beginning days, we comforted and encouraged each other when meeting difficulties and being sad. Without you guys, my brothers and sisters, the life here would have been harder and more boring.

Contents

Certificate of Examination ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Problem Statement 1

1.2 The Objective of this Research 2

1.3 The Organization of the Thesis 3

2 Terminology 5

3 Literature Review 9

3.1 Generic Intrusion Detection System Model 9

3.2 Intrusion Detection Classification 10

3.2.1 Signature-based IDS versus Anomaly-based IDS 10

3.2.2 Network-based IDS versus Host-based IDS versus Database-based IDS 13

3.3 The Necessity of Database Intrusion Detection 14
3.4 Review of Current Database IDS Research 16
3.4.1 Signature-based Database Intrusion Detection 16
3.4.2 Anomaly-based Database Intrusion Detection 21
3.4.3 Hybrid Database Intrusion Detection 24
3.4.4 Intrusion Prevention System 25
3.5 Chapter Summary 28

4 Preliminaries **29**
4.1 Role-based Access Control 29
4.1.1 Role Hierarchy 31
4.1.2 Abstract Roles 31
4.2 AdventureWorks 32
4.3 Raw Data Format 33
4.4 False Positives and False Negatives 35
4.5 Chapter Summary 36

5 System Design **37**
5.1 Data Generation 37
5.1.1 Training and Testing Data Set 37
5.1.2 Intrusion Data Set 38
5.2 System Architecture 40
5.3 Classifier 42
5.3.1 Decision Tree 42
5.3.2 Naive Bayes Classifier 44
5.4 User Profiling Vs. Role Profiling 47
5.5 Role Profiling with Role Hierarchy 50
5.6 Chapter Summary 51

6 Experimental Evaluation and Related Analysis **52**

6.1 False Negatives/Positives Test Results and Discussion 53

6.1.1 False Negatives/Positives Test Results 53

6.1.2 Analysis of the System Behaviors 55

6.1.3 Training Threshold 57

6.2 Overhead 57

6.3 Training Time 58

6.4 Chapter Summary 60

7 Conclusions **61**

7.1 Summary 61

7.2 Contributions 62

7.3 Future Work 63

7.3.1 Mine Referenced Tables From Collected Data 63

7.3.2 Roles with Overlapped Permissions 64

7.3.3 Monitor the DBAs 64

7.3.4 Real-time Detection 64

7.3.5 Response Mechanism 65

7.3.6 More Ambitious Plans 65

Bibliography **67**

Curriculum Vitae **72**

List of Tables

4.1 The tables (views) numbered 34

5.1 Role-permission assignment and user-role assignment 39

5.2 An example of a collected transaction in raw format 41

5.3 An example of a transaction after being parsed 42

5.4 Two sample days 43

6.1 Performance test (in Sec) 58

List of Figures

3.1 A generic intrusion detection model (From [16]) 10

3.2 Components of the DEMIDS architecture (From [8]) 17

3.3 Architecture of DIDAFIT (From [20, 18]) 19

3.4 Overview of Valeur's System (From [37]) 21

3.5 Overview of the intrusion detection process of Bertino's system (From [6]) 23

3.6 The Authorization Rule System Architecture (From [29]) 26

3.7 Multi-Agent based Database Statistical Anomaly Prediction System (From [30]) 27

4.1 Components of the ANSI RBAC (From [26]) 30

4.2 Components of the Role Graph Model (From [26]) 30

4.3 An example of the abstract roles 32

4.4 Role hierarchy of the Sales Department and the Marketing Department 33

4.5 An example of applications we design 34

5.1 System working process 40

5.2 A sample decision tree (From [21]) 43

5.3 An example of one role's profile 48

5.4 An example of one user's profile 49

6.1 False negative rate test 1 53

6.2 False negative rate test 2 54

6.3 False positive rate test 54
6.4 Training time 59

7.1 System architecture for real-time detection 65

Chapter 1
Introduction

1.1 Problem Statement

With the digitalization of the world, a considerable amount of invaluable data has been stored in databases, and its security problems have also begun to attract more and more attention. Modern database management system (DBMS) can provide multi-layer security; for example, access control, authentication, authorization and encryption are often applied to ensure the security of the data; however, they cannot perform perfectly. A significant drawback of these traditional database security mechanisms is that they could do little to prevent insider threats while the majority of the security problems are actually caused by insiders [28, 4].

Therefore, extra security mechanisms are necessary; using Intrusion Detection Systems (IDS) can be a promising approach to enhance the security of the data in the databases. During the last decade, much work has been done to develop IDSes, but most of them are for networks or hosts (operating systems), implying that currently people could find few ideal IDSes specifically designed for databases. In this case, we believe it is meaningful to take into account the special characteristics of DBMS and develop a new database intrusion detection system in which every inside user's behaviors can be continuously monitored at the application level.

1.2 The Objective of this Research

The overall objective of this thesis is to provide a prototype of the anomaly-based database IDS that can be used to monitor legitimate database inside users and reduce insider threats. This overall goal can be divided into several subgoals.

First, our system will be coupled with role based access control (RBAC) when RBAC is supported by the DBMS, and this means we will build profiles which characterize the normal behaviors for roles (role profiling) instead of individual users (user profiling). The most apparent advantage of role profiling is that it can substantially reduce the number of necessary profiles, which makes the IDS more efficient and much easier to be deployed within the very large organizations. In addition to fewer profiles, if the IDS using role profiling can portray the users' behaviors as accurately as the one using user profiling and perform as effective as the latter one, too, then it is certainly preferred in practice. Therefore, we conduct detailed comparisons between user profiling and role profiling in order to evaluate the effectiveness of these two profiling approaches. We also analyze the reasons why one profiling approach performs better than the other.

Second, in the research on anomaly-based IDSes, although people can often achieve a relatively low false negative rate, a high false positive rate maintains a challenging problems. One direction of decreasing the false positive rate is to develop a better machine learning classifier which is applied to profiling normal behaviors. However, we feel that it is difficult to largely improve the behavior of a machine learning classifier, so we turn to another direction. We make use of more features of RBAC so as to reduce the false positive rate without increasing the false negative rate.

Finally, we want to highlight again that intrusion detection should be a supplementary security layer of the database system but not a replacement to traditional security mechanisms.

1.3 The Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we give some basic definitions and notations related to our work so that the readers can better understand the rest of the thesis.

Chapter 3 consists of a literature review of intrusion detection systems. We introduce a generic intrusion detection model, followed by the classification of IDSes – misuse-based versus anomaly-based or network-based versus host-based. We also describe the fundamental differences between misuse-based intrusion detection and anomaly-based intrusion detection and between network-based intrusion detection and host-based intrusion detection. The rest of this chapter focuses on database intrusion detection. Several database IDSes are described and their advantages and drawbacks are analyzed.

In Chapter 4, necessary preliminaries for understanding the rest of our work are introduced. We present the fundamental components of RBAC including role hierarchy at first. The database named AdventureWorks based on which our system is built is then described; also, we build appropriate role hierarchy for this database. After that, we show the raw data format we collect. We conclude this chapter by explaining the concepts of false positives and false negatives which will be used to evaluate our system during our test mode.

Our IDS is proposed in Chapter 5 in detail. At the beginning of this chapter, we introduce how we generate training and testing data including the approach to generating intrusions. The system architecture and the machining learning classifier we use are then presented. Following that, we arrive at the description about how to parse the raw data and how to use the classifier to build our system, using user profiling (build profiles for each individual user) and simple role profiling (build profiles for roles), respectively. Our novel advanced profiling method taking into account the role hierarchy is described at the end of this chapter.

Chapter 6 illustrates the results of the experimental tests for the evaluations of the three profiling approaches – user profiling, simple role profiling and advanced role profiling. False positives, false negatives, overhead and training time are carefully and fairly measured for each profiling method. Based on the evaluation results, we further analyze the reasons why user profiling results in the much inferior behaviors compared with the other two alternatives and why our advanced role profiling approach can overtake the simple role profiling approach.

Finally, we present the conclusions of our research and our possible future research directions in Chapter 7.

Chapter 2
Terminology

This chapter explains some related terms used in this thesis, including several generic concepts relating to intrusion detection and some specific concepts of database intrusion detection. The terms belonging to the former category are well-accepted definitions, but as database intrusion detection is too new and very limited work has been done on this area, terms for database intrusion detection have not been standardized among security professionals.

- *Intrusion detection*

  Intrusion detection acts to detect actions that attempt to compromise the confidentiality, integrity or availability of a resource [3].

- *Intrusion prevention*

  Intrusion prevention is to act in real time to block or prevent malicious or unwanted behaviors of a network and/or system by monitoring its/their activities [35].

- *Database auditing*

  Database auditing involves observing the activities related to a database in order to be aware of the actions of database users, usually for security purposes. It mainly concerns authentication and authorization issues, and the auditor needs to develop an audit strategy, audit suspicious database activity and audit normal database activity [5].

- *Insider abuse/threat*

  An insider abuse/threat is a malicious or inadvertent activity by the legitimate internal users that threatens system security. In the rest of this thesis, insider threats are included in intrusions (in other papers, intrusions may refer to only break-in attacks).

- *Signature*

  In misuse detection, a signature refers to "the specification of features, conditions, arrangements and interrelationship that signify a break-in or other misuse, or their attempts" [16].

- *Misuse detection*

  Misuse detection defines abnormal network or/and system behaviors as signatures in advance and any activities that match the signatures are recognized abnormal. In intrusion detection, it can work against both attacks from outside or inside users.

- *Profile*

  In the context of anomaly detection, profiles are models that can represent the normal behaviors of a network or/and system.

- *Anomaly detection*

  Anomaly detection portrays normal network or/and system behaviors within profiles in advance typically using machine learning technique. It then compares actual behaviors with the profiles and any deviation from the profiles is considered to be an anomaly.

- *SQL injection*

  SQL injection is a category of attacks which exploit the vulnerabilities of DBMS by inserting some malicious code into strings that are then passed to be parsed and formulated into SQL statements. We give an example of this attack below.

Suppose we have a database storing the academic records of students. The student number and the password are required by the system when a student requests access to his/her own records. Moreover, Java is used to implemented the logging application. In this case, when a student inputs his/her student number (012003017217) and password (850719), Java manufactures
"select * from student_table where studentnumber = '" + 012003017217 + "' and password = '" + 850719 + "';"
to generate the SQL statement that is submitted to the database server. The generated SQL statement is:
select * from student_table where studentnumber = '012003017217' and password = '850719';
However, if a string ' or '1' = '1 is entered as the password, and 012003017217 is the student number, then the following SQL statement is generated:
select * from student_table where studentnumber = '012003017217' and password = '' or '1' = '1';
As 1 = 1 is always true, the attacker can illegally access the academic records of the student whose student number is 012003017217 without really knowing the password.

- *Query flood*
  Query flood is a type of attacks that can be conducted towards databases. By issuing a very large number of requests, a subject or a set of subjects can flood the database, therefore causing the database server to be unable to answer the requests from honest subjects within the reasonable response time [7].

- *Role-based access control (RBAC)*
  RBAC is an approach to restricting system access to authorized users. In RBAC, permissions are not granted to users directly; instead, they are assigned to roles. The users can get the permissions associated with the role or roles

he/she is assigned to [10, 34, 24].

- *False positive and false negative*

  False positive and false negative are two concepts used to describe possible statistical errors [2]. Abstractly, the former one means "rejecting null when null is true" while, in contrast, the latter one means "retaining null when null is false" [23].

  In the context of intrusion detection, a negative instance means a legitimate transaction and a false negative refers to an intrusion considered legitimate; a positive instance means an illegitimate transaction (intrusion) and a false positive refers to a legitimate transaction recognized as an intrusion.

Chapter 3

Literature Review

3.1 Generic Intrusion Detection System Model

Intrusion detection systems(IDSes) are software and/or hardware designed to detect unwanted attempts at accessing, manipulating, and/or disabling of computer systems [3]. They have been proved effective and efficient to detect actions that attempt to compromise the confidentiality, integrity or availability of a digital resource, namely intrusions [12], since Dorothy Denning [9] first proposed the concept of intrusion detection.

The basic idea of intrusion detection is that the models of normal usage of the system can be built based on security rules, and an intrusion will always involve some abnormal usage of the system. Therefore, the IDS can compare the usage of the system against normal usage and any significant deviations from normal usage will be flagged as abnormal usage. Figure 3.1 [16] depicts a generic intrusion detection model which is independent from any particular systems, application environments, system vulnerabilities and intrusion types. This model, introduced as a framework for a generic intrusion detection system, is an abstract model for further development in this area.

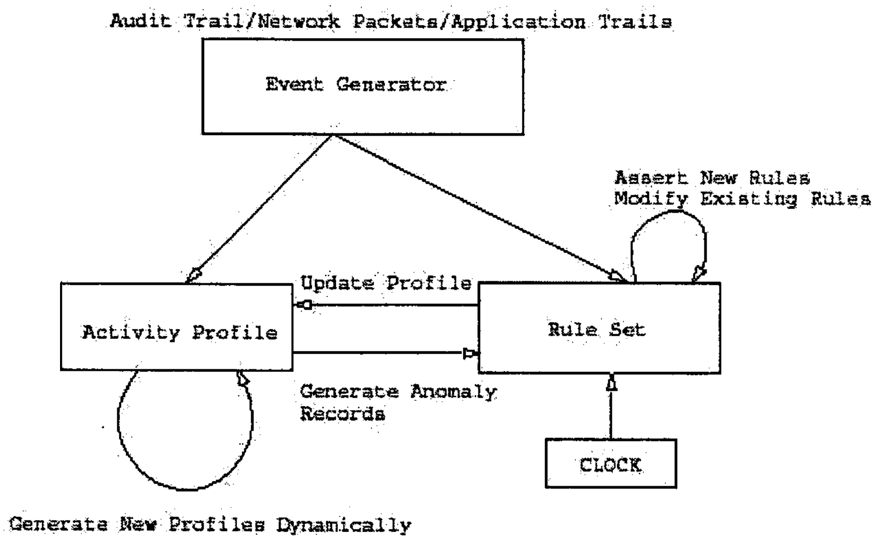


Figure 3.1: A generic intrusion detection model (From [16])

3.2 Intrusion Detection Classification

3.2.1 Signature-based IDS versus Anomaly-based IDS

There are a lot of IDS models which have been developed and all of them are either signature-based or anomaly-based. Generally speaking, signature-based IDSes can detect previously known attacks very well but do little when encountering original intrusions; while an anomaly-based IDS is able to detect new/unseen attacks but may have a high false positive rate.

Signature-based IDS

Some intrusions follow well-defined patterns of attacks that exploit vulnerabilities of computer system and applications, so a straightforward idea of detecting such intrusions is to portray the patterns (signatures) of attacks in advance and store them in a database of signatures against which the users' behaviors are matched. In this way, intrusions can be detected as long as the corresponding signatures exist. This idea actually forms the fundamental methodology of a signature-based intrusion detection system, also known as a misuse-based intrusion detection system. The explicit patterns, or signatures, that are previously characterized can be typical strings of attacks or a sequence of suspicious actions; for example, a network packet may be a pattern if it has the same source and destination IP addresses, which indicates a Land attack is under way [1].

As we can see, the detection ability of a signature-based IDS mainly depends on the database of previously written signatures, since the IDS must know explicitly about an attack before being able to detect it, and in this case, the database of signatures must be updated constantly to include as many signatures as possible, in order to maintain its effectiveness.

Anomaly-based IDS

An anomaly-based IDS refers to anomalous behaviors that deviate from normal behaviors, and such anomalous behaviors are considered potential intrusions. A simple example is that the user Bob in a bank usually uses his office computer between 9:00AM and 5:00PM, so if someone attempts to log into the computer using Bob's account at mid-night, we can say this behavior is anomalous, and may be an intrusion.

The above description shows that for an anomaly-based IDS, it is significant to build models for normal behaviors in advance which are usually named profiles. Machine learning algorithms are often applied to build the profiles necessary and the statistical measures of the system features; for example, the time, source/destination IP addresses, CPU usage, etc., can be taken into consideration. This process that creates normal profiles is called the training mode, and in order to portray normal behaviors, the IDS needs a lot of intrusion-free training data to build the profiles.

In detection mode, the anomaly-based IDS works by comparing any new behavior with normal profiles using some distance measures, and whenever the distance crosses the preset threshold, the new behavior is recognized as an anomaly and the IDS will alarm the administrators.

As the normal behaviors of a user or other subjects may change as time goes on, it is necessary to update the profiles so that they can represent the most recent normal behaviors.

Comparisons of Signature-based IDS and Anomaly-based IDS

Signature-based IDS

The central premise for a signature-based IDS to work effectively is that it can extract the signatures of previous intrusions. For intrusions whose signatures have been stored, the IDS can detect them accurately, which means it has low false positives.

However, it is highly unrealistic in practice to get the signatures of all intrusions since people keep creating new types of intrusions. As we can see, signature-based IDS can actually do little when unknown attacks happen, which may cause high false negatives, and this is the biggest limitation of this type of IDS. Another problem is that an intrusion can have many variants, and sometimes a new variant can be created by simply adjusting the input (s). Therefore, what an attacker needs to do to avoid being detected by the IDS is just to change the input because the signature of the new variant may not be in the signature database, which also can cause false negatives.

In summary, a signature-based IDS usually has low false positive rate but has high false negative rate.

Anomaly-based IDS

The main assumption of anomaly-based intrusion detection is that attacks are anomalous behaviors that deviate from normal behaviors. Such type of IDS compares new behaviors with profiles that can represent normal behaviors. The most significant advantage of anomaly-based intrusion detection is that it can deal with unknown intrusions.

One problem is that the profiles cannot always include all normal behaviors, which makes some legitimate behaviors be considered anomalous and cause high false positive rate. Additionally, the administrator needs to decide a threshold for the distance measures used in the detection mode, and as an anomaly-based IDS is often subject to high false positives, the administrator may prefer a higher threshold. This, however, may make some real attacks unrecognized. Moreover, an attacker may slowly modify or spread his/her behaviors over time to cause the distance between the behaviors and profiles to be under the threshold so that they cannot be detected. They can even change the profiles bit by bit to what he/she wants if the profiles are being updated constantly (although it is really time consuming).

In summary, anomaly-based intrusion detection can detect previously unknown attacks but often has high false positives.

3.2.2 Network-based IDS versus Host-based IDS versus Database-based IDS

The research on intrusion detection has been on going for more than 20 years, most of which is either at the network level (network based IDS) or at the host level (host-based IDS); it was not until very recently that some researchers began to focus on database intrusion detection, taking into account special features of DBMS such as SQL statements.

Either network-based intrusion detection or host-based intrusion detection can be signature-based or anomaly-based. Therefore, there are four combinations in total:

- Signature-based network intrusion detection
- Anomaly-based network intrusion detection
- Signature-based host intrusion detection
- Anomaly-based host intrusion detection

Network-based Intrusion Detection

Network-based intrusion detection is the most active area in intrusion detection research. Typically, it monitors the network traffic and builds models for normal traffic (anomaly-based) or extracts signatures of intrusions like Denial of Service (signature-based). Currently, people are paying more attention to anomaly-based network intrusion detection because there are often many new types of attacks on networks that only anomaly-based intrusion detection can deal with. The authors of [17] give a detailed comparative study of several anomaly-based network intrusion detection schemes, including both supervised ones and unsupervised ones.

Host-based Intrusion Detection

Host-based intrusion detection, in contrast, usually works at the operating system level, monitoring each user's behaviors (usually the log files of each user). Lee et al. [13] introduce an anomaly-based host IDS for UNIX systems using a Self-Organizing Map (SOM) to test if the behavior of a user is anomalous.

Database-based Intrusion Detection

Database-based intrusion detection is a relatively new branch in intrusion detection research and not much research has been done. The next section will focus on database intrusion detection.

3.3 The Necessity of Database Intrusion Detection

Databases, needless to say, play important roles in information systems, but their security problems have caused a huge amount of loss. According to the FBI Computer Crime and Security Survey in 2005 [11], the 700 surveyed entities reported a loss of $30,933,000 due to the theft of valuable information, most of which is stored in databases.

Admittedly, we already have many security mechanisms used in DBMS like authorization, authentication and integrity control which can solve a lot of database security problems; we can also use encryption, firewalls and anti-virus products to maintain security. But intrusions and insider abuses still exist. A key weakness of traditional security mechanisms is that as long as a user logs into the DBMS successfully, he/she will no longer be monitored. Therefore, a database IDS that can continuously monitor every user's behavior will be a meaningful complementary security mechanism for database systems.

In addition, although there are many network-based and host-based IDSes developed, these IDSes cannot detect malicious or misuse behaviors at the database level well because most of them do not work at the application layer and the audit mechanisms at the host (actually operating system) level or at the network level cannot reflect a database user's behavior accurately. So as long as a 'bad guy' is a user with a legal account on the system and he/she does not cause network traffic anomalies, any illegitimate actions to try to gain database privileges will be invisible to the current IDS.

Even though some network-based IDSes or host-based IDSes indeed work at the application layer, which means they can usually have more accurate detection functionalities for a specific application, they are still not able to ideally detect intrusions targeted at databases. For example, SQL-injection attacks are out of control whatever network-based IDS or host-based IDS people use because none of them take into account the special features a DBMS has.

Moreover, both network-based intrusion detection and host-based intrusion detection mainly focus on detecting intrusion from external users; however, people find the key threats for databases are the privilege abuses of internal legitimate users [28, 4].

Therefore, it is undeniable that research on database intrusion detection is imperative and meaningful, and with deployment of the IDS at the database level where valuable data is stored, it will be much more difficult for attackers to carry out attacks such as SQL-injections targeted at a database so that the security of a database can be largely improved. Also, although database intrusion detection by taking into account the distinctive characteristics of DBMS can surely be a promising complement to database security, it is not aiming to replace traditional database security mechanisms.

3.4 Review of Current Database IDS Research

Very limited work on database intrusion detection has been done and all of the database IDSes that have been developed can fall into three categories:

- Signature-based database intrusion detection
- Anomaly-based database intrusion detection
- Hybrid database intrusion detection

3.4.1 Signature-based Database Intrusion Detection

DEMIDS

DEMIDS, introduced in [8], is a misuse-based IDS coupled with some anomaly-based intrusion detection characteristics. Through the notion of distance measure that can measure the closeness of a set of attributes, the domain knowledge of the database such as the data structure and semantics specified in the database is considered. Based on the assumption that typically a user will only access some particular attributes and data in a schema or database, Chung et al. then introduce the concepts of work scope and frequent itemsets (which can be considered as signatures) to capture the attributes that are often referenced together by a user. DEMIDS then uses its novel machine learning algorithm to discover all frequent itemsets in audit sessions and

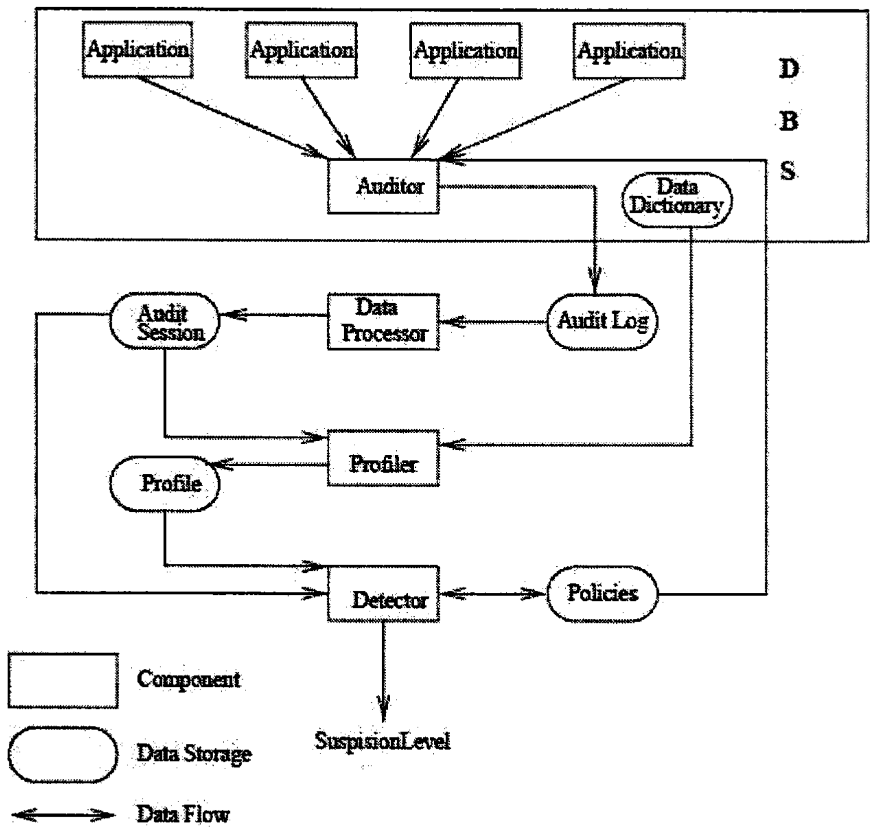


Figure 3.2: Components of the DEMIDS architecture (From [8])

builds profiles for users. In the detection module, it collects the new audit data in the form of newly issued SQL statements by a user, and then compares them against the derived profiles of this user. If the attributes contained in the new SQL statement are not equal to any frequent itemset or its subsets in stored in the profiles of that user, the user is recognized an attacker because he/she is trying to go out of his/her work scope. Figure 3.2 [8] depicts the system architecture of DEMIDS. We can see that it consists of four components: Auditor, Data Processor, Profiler and Detector. In order to exploit some functionalities of the database such as auditing and query processing, the DEMIDS is tightly coupled to the existing DBMS.

The Auditor is responsible for collecting interesting audit data of users by auditing their queries by making use of the auditing functionality of the DBMS. These monitored features are then recorded in audit logs. The Data Processor is used to preprocess the raw data in the audit logs, for example, it handles missing values and converts the raw data into proper data structures and types for the Profiler. Another crucial responsibility of the Data Processor is to group the raw audit data into different audit sessions, for instance, based on different UserID. The Profiler generates a profile for each audit session during the training process while the Detector computes a score for each new audit record to determine if user activities are abnormal during the detection stage.

Comments:

The paper introducing DEMIDS also presents the theoretical proof that it can perform effectively using itemsets; moreover, the Data Processor can also group audit data based on roles so that that we need much fewer profiles.

But this IDS assumes domain knowledge encoded in a certain database schema while building user profiles, and this can potentially weaken the generic applicability of this system. In addition, it utilizes the auditing and query processing functionalities provided by the DBMS, which may make the IDS unable to respond in time when a misuse happens.

DIDAFIT

DIDAFIT is a signature-based IDS proposed in [20, 18]. This IDS assumes all users are application users and it characterizes legitimate transactions (SQL statements issued) by abstracting their fingerprints instead of fingerprinting illegal SQL statements. It collects legitimate SQL statements to form a database of legitimate fingerprints, against which a newly issued SQL statement is matched, and if the system cannot find a corresponding fingerprint, the SQL statement would be considered illegitimate. In order to include as many legitimate transactions in the database of fingerprints while ensuring that the database is still not too large, algorithms are developed to

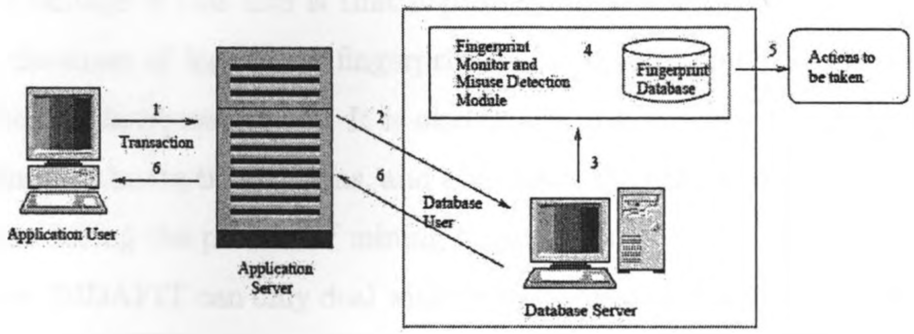


Figure 3.3: Architecture of DIDAFIT (From [20, 18])

summarize the fingerprints and deduce missing fingerprints. Figure 3.3 [20, 18] portrays the architecture of DIDAFIT. Assuming the database of legitimate fingerprints has already been set up, the system works as follows:

1. The application user issues a legitimate or illegitimate service request to the application server.

2. The application server then formulates the necessary SQL statements and issues them to the database server through the corresponding database user.

3. The database user then logs into the database. Meanwhile, the database session is traced and all SQL statement received from the application server are channeled to the misuse detection module.

4. In the misuse detection module, the SQL statement are matched against the legitimate fingerprints and examined if there are corresponding ones.

5. If a SQL statement cannot find a corresponding legitimate fingerprint, an anomaly or intrusion is detected and it will be channeled to the reaction module. Possible responses that can be taken include notifying the DBA, sounding an alarm and so on.

Comments:

The key advantage of this IDS is that it produces very low false positives especially when the database of legitimate fingerprints is complete; moreover, it causes little overhead to the database system. It is also interesting that it can automatically deduce missing legitimate transactions, and even allow the existence of some illegitimate transactions during the process of mining fingerprints.

However, DIDAFIT can only deal with limited variants of standard queries. Moreover, as it assumes all users can only interact with the database server through the application server and there are no direct user-database interactions, it actually could not be used at the database administrator (DBA) level, which indicates some privileged users such as the DBA are still out of control.

Intrusion Detection In Real-time Database Systems via Time Signatures

Victor C.S. Lee et al. present an IDS designed for real-time database systems [19]. The real-time properties of data, mainly the time semantics of data objects, are exploited to detect intrusions. Update rates are taken into consideration as time signatures at the sensor level. Any request of update issued out of the expected time will trigger alarms. The core of this method is actually to embed security rules into data objects. These rules specify constraints that define the correct state of a data object and the relationship of objects as well as the action to be taken over the events such as rejecting the anomalies.

Comments:

This IDS produces low false positives like other signature-based IDSes, and although it is designed for real-time database systems, its application can be extended to some non-real-time database systems in which time signature can be portrayed.

However, if the time correlations or the expectation of update transactions are not obvious in a database system, this IDS cannot perform effectively. Additionally, it only monitors sensor transactions, or specifically only update transactions, so it is

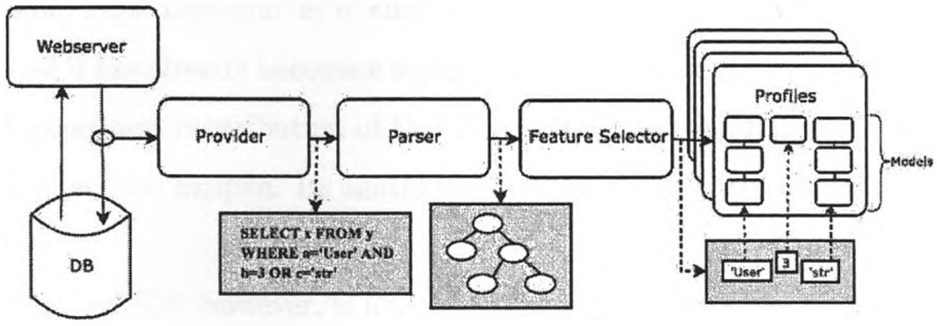


Figure 3.4: Overview of Valeur's System (From [37])

able to do little when other types of intrusions occur.

3.4.2 Anomaly-based Database Intrusion Detection

Learning-based Approach to the Detection of SQL Attacks

Fredrik Valeur et al. developed an anomaly-based IDS that constructs profiles reflecting normal database transactions performed by web-based applications using a number of different statistical models [37]. String Models can portray normal string length, string character distribution, etc, while the Data Type Independent model can portray common integer values.

An overview of the system architecture is shown in Figure 3.4. The IDS taps into the communication channel between web-based applications and the back-end database server so that all queries issued by the applications can be intercepted and sent to the IDS for examination. The IDS parses the SQL statements and selects which features of the statements should be modeled. In the training phase, features selected from the first training set are fed to build profiles and those from the second training set are used to generate a threshold. An anomaly score for every query will be generated to check if it is anomalous in the detection phase.

Comments:

As we know, SQL injection is a kind of attack specifically towards the database systems, and it has already become a significant threat to database-based applications. The most important contribution of this approach is that the IDS performs very well when SQL injections happen. Its another advantage is that it can detect some mimicry attacks, too.

The proposed IDS, however, is limited to detecting three classes of SQL-based attacks which are SQL injection, cross-site scripting and data-centric attacks, so other types of attacks like query flood cannot be detected effectively. Additionally, the overhead caused by the IDS is not ignorable. Another significant limitation of this approach is that it requires some modification to the library responsible for communication between applications and database servers, resulting in that it can only be used in an open source database such as MySQL, which indicates that it is infeasible for this IDS to be extended to most commercial DBMS such as DB2, Oracle and SQL Server.

Database Intrusion Detection coupled with Role-based Access Control

Elisa Bertino et al. introduce Role-based Access Control (RBAC) into the database intrusion detection research in [6]. The major point of this work is that it assumes eight application roles and builds profiles for roles instead of for individual users, so that the number of profiles necessary can be dramatically reduced. The system's architecture has four main components: the user that issues queries, the conventional DBMS mechanism that processes queries, the database log files that record queries and the intrusion detection mechanism that detects intrusions. The latter three components constitute the new extended DBMS integrated with an independent IDS working at the database level. Figure 3.5 illustrates the working flow of the system. The log files are updated whenever a new query is issued. In the training phase, the system mines the existing log files and builds profiles for roles while in the detection mode, every newly issued SQL statement is examined to determine if it is anomalous

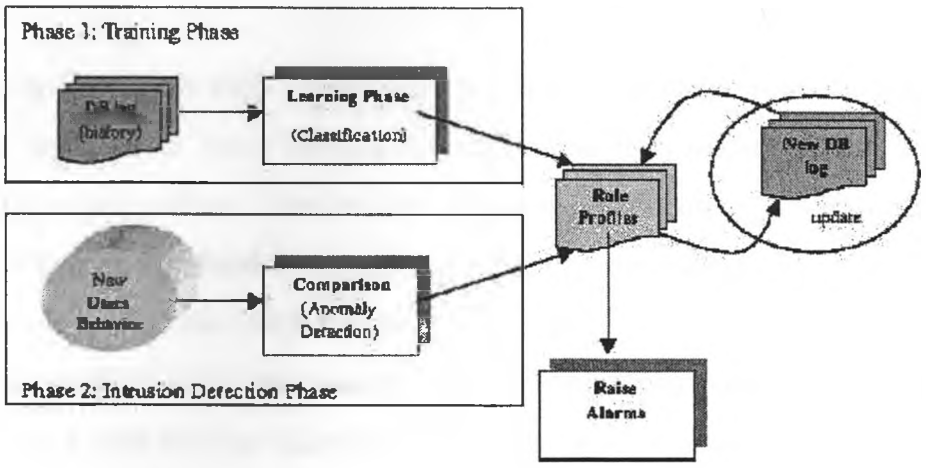


Figure 3.5: Overview of the intrusion detection process of Bertino's system (From [6])

based on the profiles. An alarm will be raised if an anomaly is detected. A Naive Bayes Classifier is used in the training and detection phases. The profiles can be updated periodically so that they can represent the most recent activities of roles.

To give more details, using a Naive Bayes Classifier, each role is considered as a class and profiles for each class are built, which is called supervised training. Then in the detection phase, a Naive Bayes classifier also predicts a role for every new query, and if the predicted role is different from the original role associated with the query checked, an intrusion is detected. It is also worth mentioning that using a Naive Bayes classifier, most of the computational tasks can be done during the training phase [21], so the detection latency can be minimized.

According to different security requirements, the IDS can work at three granularities by generating varying triplets used in the training phase from log files: coarse triplet, medium triplet and fine triplet. The first one can only reflect the number of relations and attributes accessed, the second one can show which tables are accessed and the number of attributes accessed in every table, and the last one can illustrate precisely which tables and attributes are accessed.

Comments:

As this work is coupled with RBAC and profiles are built for roles rather than for individual users, much fewer profiles are needed so it is easier for this IDS to be used in large organizations. Also because of the short detection latency, the IDS can respond in time when intrusions happen. Moreover, users assigned to privileged roles could be monitored as well with a simple extension.

But one problem is it only assumes eight application roles, which is far from realistic, and it also fails to take into the situation that a user's behavior deviates from the role assigned but not enough to be recognized as another one, which may cause false negatives. Certainly another obvious problem is that this system could not be used in a DBMS that does not support RBAC, which narrows the application of this IDS. Kamra et al. extend this work using unsupervised machine learning techniques [15]. K-centers or k-means algorithms are applied to train the data so the users are grouped and profiles are constructed for every group. However, this approach fails to avoid high false positives/negatives.

3.4.3 Hybrid Database Intrusion Detection

Database IDS Detecting Query Flood

Elisa Bertino et al. exploit information theory to detect a specific type of attack towards databases - query flood [7]. It can work either as misuse intrusion detection by analyzing and mining the log files and focusing on the frequency of commands of a specific type, or anomaly intrusion detection based on modeling access profiles and using them to detect unusual actions. In their IDS, the objects monitored are not users but the database or tables in the database.

Comments:

This approach is able to detect query flood effectively and it can support varying security granularities flexibly since it not only can analyze and model for an entire database, but also is able to focus on a particular table; however, the most obvious drawback is that it can do nothing to detect other types of attacks. Another problem is that as it monitors the whole database or tables instead of users, it is hard to react without influencing other legitimate users when attacks are detected because it does not know where the attacks come from and therefore is not able to stop the connection between attack sources and the database directly. A possible method is to temporarily deny all access requests to the database or to some particular tables, which, however, also causes denial of service.

3.4.4 Intrusion Prevention System

Strictly speaking, an intrusion prevention system (IPS), which can predict and prevent intrusions, is different from an IDS, but it can be also considered as an extension to an IDS.

Ramasubramanian et al. present a distributed real-time IPS called Intelligent Multi-agent Based Database Hybrid Intrusion Prevention System [30, 33]. This approach applies both misuse prevention and anomaly prevention. The misuse prevention module is rule based and the rules include those authorized permissions and privileges [29]. The anomaly prevention module uses ensemble Quickprop neural networks to forecast potential intrusions (see [32, 31] for details of ensemble Quickprop neural networks).

The architecture of the misuse prevention module, namely the Dynamic Access Control System, is shown in Figure 3.6. It consists of five components: XML Database Server, Query language Component, Database Manager, Rule Events Manager and Query Processor. The architecture of the anomaly module is presented in Figure 3.7. The Host Agent is embedded into every host system and monitors each host's behaviors, while the Information Agent acts as the data processing unit and data repository for the Host Agent. More specifically, the Information Agent collects and stores user profiles for all users from various Host Agents.

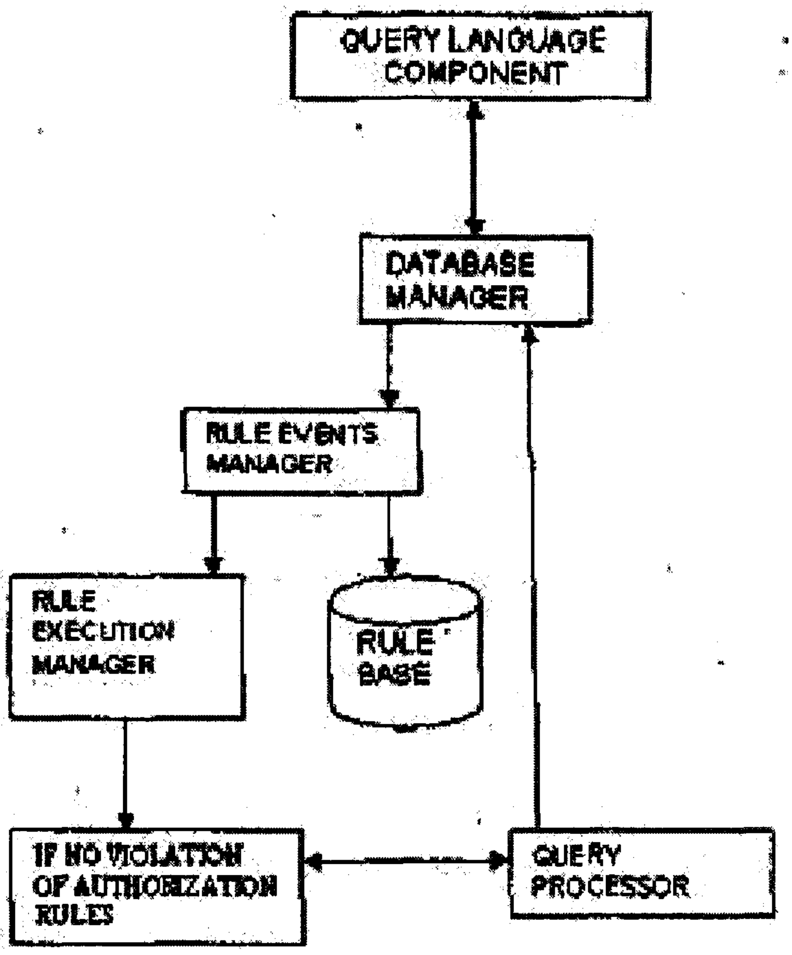


Figure 3.6: The Authorization Rule System Architecture (From [29])

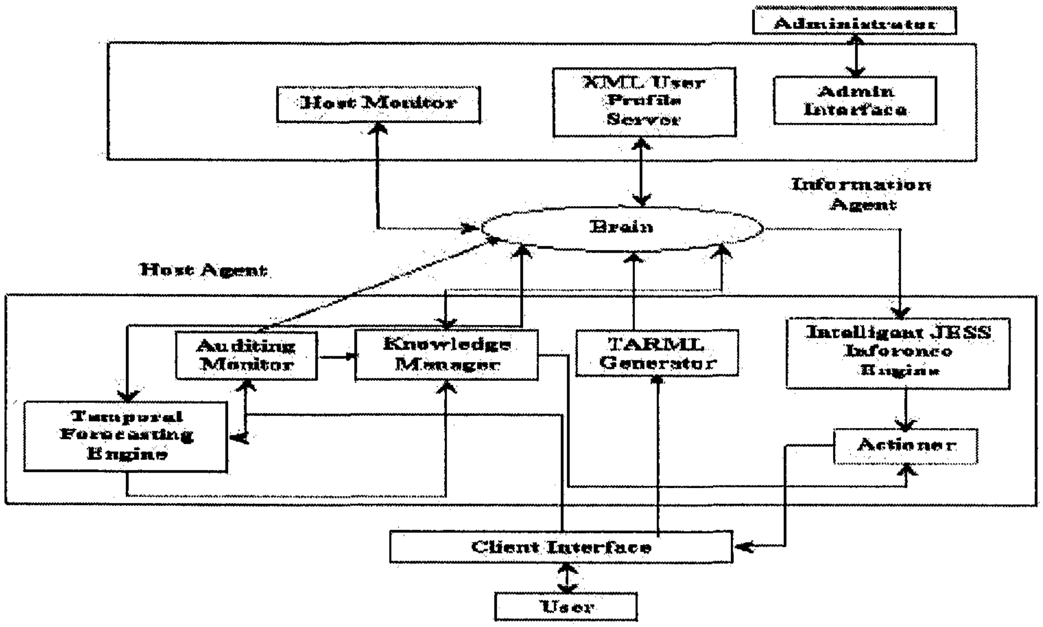


Figure 3.7: Multi-Agent based Database Statistical Anomaly Prediction System (From [30])

Comments:

The most noteworthy point of this approach is its use of a neural network to observe users' previous behaviors and predict their future actions so that intrusions can be prevented before they occur.

But when detecting intrusions after they happen still often suffers from high false positives and false negatives, it is easy to imagine that predicting intrusions may probably be more inaccurate, which is indeed the major problem of this IPS. Particularly, the prediction for users whose behaviors are not very regular is quite inaccurate. Moreover, using neural networks usually requires a very large set of training data which sometimes is not easy to obtain.

3.5 Chapter Summary

This chapter has surveyed the intrusion detection research, particularly paying attention to database intrusion detection.

After introducing some fundamental concepts of intrusion detection, we discussed the database intrusion detection and then devoted a large part of this chapter to reviewing several database intrusion detection approaches presented in the recent literature. Similar to the network and host intrusion detection research, people have also introduced anomaly-based detection and signature-based detection into database intrusion detection; moreover, some researchers apply hybrid approaches to detect intrusions. For every IDS presented, we introduced its main assumptions, models of gaining signatures or building profiles, models of detection, and its system architecture. In addition, the advantages and drawbacks of each system were analyzed.

Chapter 4 Preliminaries

4.1 Role-based Access Control

Role-based access control (RBAC) has been discussed a lot since the mid 1990s [10, 24, 34]. The primary goal of RBAC is to provide a more neutral and flexible access control alternative to traditional discretionary access control (DAC) and mandatory access control (MAC) when people need to manage complex systems with large numbers of users and data items. Osborn et al. propose how to configure RBAC to enforce DAC and MAC [27] so that RBAC can replace DAC and MAC in many practical cases. Currently, there are two mainstream categories of RBAC models – the ANSI standard model [14] and the Role Graph model [24, 25]. The components of these models are presented in Figure 4.1 and Figure 4.2 [26], respectively.

Although important differences exist between the ANSI standard model and the Role Graph Model, this section focuses on the fundamental ideas of RBAC instead of comparing the two models. RBAC has three main components named **users**, **roles** and **permissions (privileges)**. In order to do certain operations within the system, a user has to gain appropriate permissions in advance. However, permissions in RBAC are not granted to the user directly; instead, they are assigned to a role or roles. The user can get the permissions associated with the role or roles he/she is assigned to.

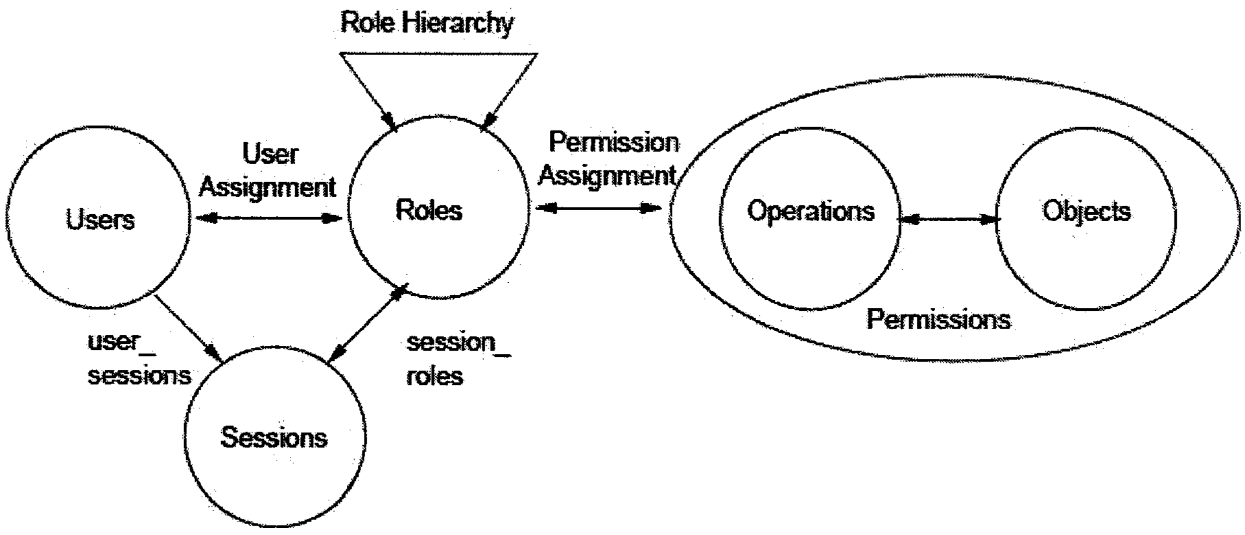


Figure 4.1: Components of the ANSI RBAC (From [26])

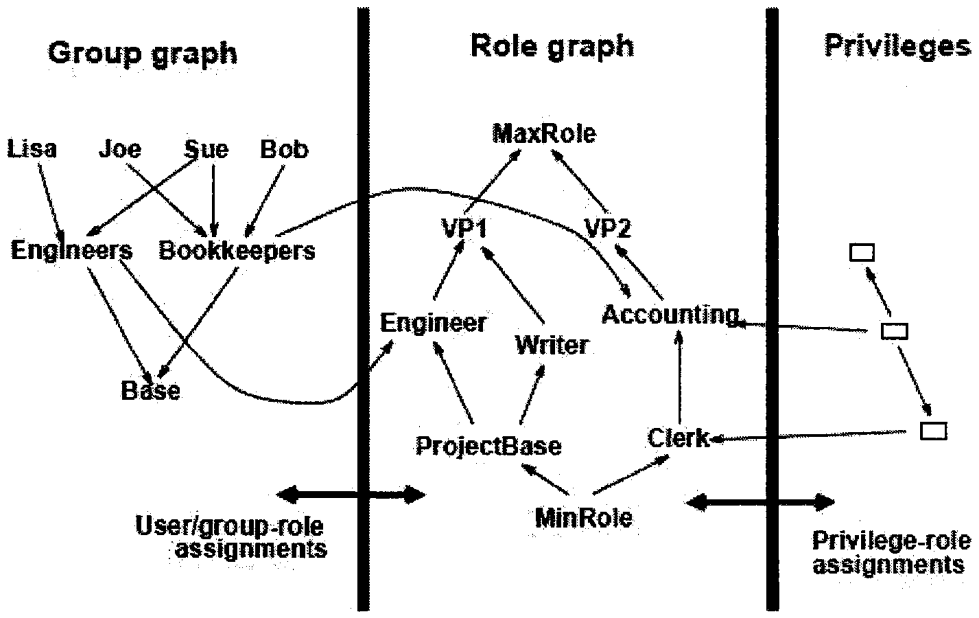


Figure 4.2: Components of the Role Graph Model (From [26])

4.1.1 Role Hierarchy

The concept of role hierarchy is introduced in both the Role Graph Model [24] and the ANSI model as the hierarchical RBAC [14]. In the hierarchical RBAC, a role in the role hierarchy inherits all permissions associated with the roles junior to it. For example, in a company's database system, we suppose the role Sales Manager is senior to another role, Sales Representative, and in this case, the former role gets all permissions the latter one has.

Something we should notice is that while a role is usually the reflection of a job position, a senior job position, however, does not have to associate with a role also senior in the role hierarchy. This can be exemplified by the fact that the CEO is a very high job position in the company who, however, does not have to access to a customer's credit card number that the corresponding Sales Representative must know. Actually, in many cases, a senior job position in the company can correspond to a junior role in the role hierarchy of the database.

4.1.2 Abstract Roles

Having abstract roles is a concept originating from object-oriented systems, and both the ANSI and Role Graph models allow such roles. In general, an abstract role is a role to which no user is assigned. This concept is very helpful when several job positions need common permissions to complete their work. We give an example below to show when we can have the abstract roles. Suppose we have 5 users ($U_1 \sim U_5$) and 6 permissions ($P_1 \sim P_6$) in a system. The users U_1, U_2 and U_3 request for permissions P_1, P_2, P_3 and P_4 while the other two users want P_3, P_4, P_5 and P_6. So we can have three roles – R_1 gets P_1 and P_2, and R_2 gets P_5 and P_6; they are both senior to R_3 to which P_3 and P_4 are assigned. Then U_1, U_2 and U_3 are assigned to R_1 and U_4 and U_5 are assigned to R_2 so that each user gets necessary permissions. In this example, R_3 is an abstract role. Figure 4.3 illustrates the user-role and role-permission assignments

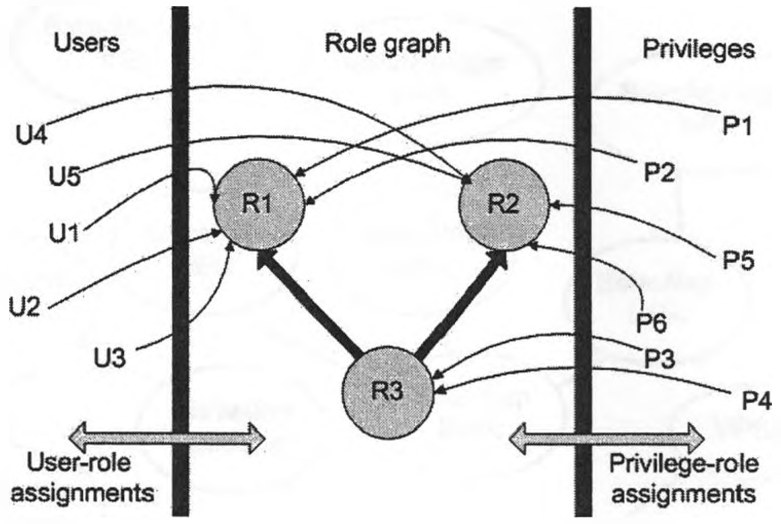


Figure 4.3: An example of the abstract roles

of this example.

4.2 AdventureWorks

AdventureWorks, based on which our system is built and tested, is a sample database provided with SQL Server 2005 by Microsoft [22]. It is the database of a fictitious company that manufactures bicycles and sells them to North America, Europe and Pacific markets. This database contains 290 users and 69 tables in total. As SQL Server 2005 can support RBAC, we design 32 roles with role hierarchy for various job functions. Our system focuses on the Sales and the Marketing Departments which have 18 and 9 employees, respectively (Figure 4.4 shows the hierarchy of the roles within these two departments). In the database, each employee has only one legal account (we use each employee's EmployeeID to represent his/her unique account in the following chapters). In addition, 12 roles belong to the Sales and Marketing Departments, including 2 abstract roles that no users are assigned to; 17 tables or views in total are referenced by the Sales and Marketing users. We number these

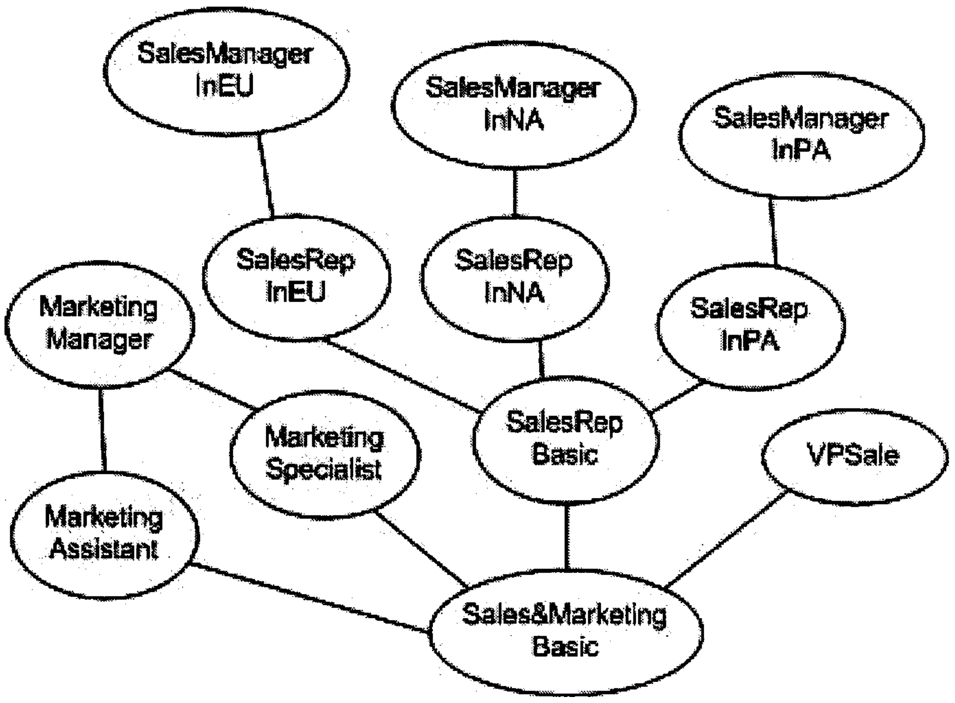


Figure 4.4: Role hierarchy of the Sales Department and the Marketing Department

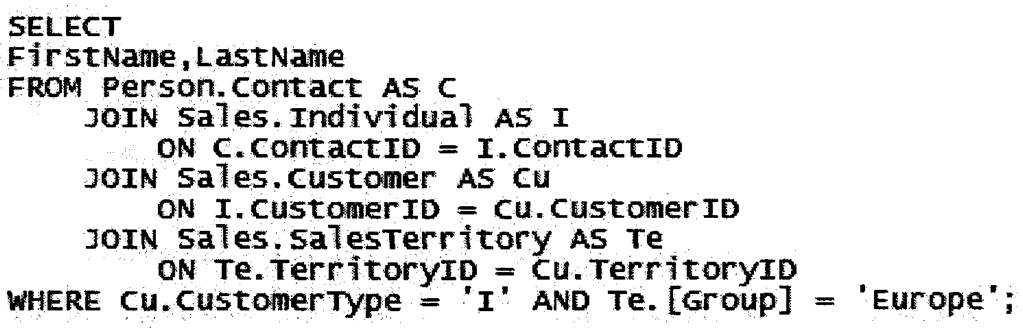
tables (views) from 0 to 16 (see Table 4.2 for details). We also design 6 applications for the Marketing Department and 25 ones for the Sales Department based on the scenarios of the AdventureWorks. For example, Figure 4.5 exemplifies the SQL statement which corresponds to the application that can be used to query the **first name and last name** of each individual customer in Europe. The permissions for invoking certain applications will be assigned to corresponding roles later.

4.3 Raw Data Format

Although using the log files of the DBMS is a quite direct and easy approach to collecting data, we prefer to use our own data collection mechanism considering our possible future research. For each transaction issued by a user, the information of 6 features is collected, including the **EmployeeID** and associated **RoleID** of the user issuing it, the **time** when the transaction is issued, the **IP address** where the

Table (View) name	Table (View) number
Sales.Individual	0
Sales.CustomerAddress	1
Sales.SalesOrderHeader	2
Sales.SalesOrderDetail	3
Sales.StoreContact	4
Sales.Store	5
Sales.Customer	6
Sales.SalesTerritory	7
Sales.SalesPerson	8
Sales.vSalesPersonSalesByFiscalYears	9
Person.Address	10
Person.StateProvince	11
Person.CountryRegion	12
Person.Contact	13
Person.ContactType	14
Production.Product	15
Purchasing.ShipMethod	16

Table 4.1: The tables (views) numbered



```
SELECT
FirstName,LastName
FROM Person.Contact AS C
    JOIN Sales.Individual AS I
        ON C.ContactID = I.ContactID
    JOIN Sales.Customer AS Cu
        ON I.CustomerID = Cu.CustomerID
    JOIN Sales.SalesTerritory AS Te
        ON Te.TerritoryID = Cu.TerritoryID
WHERE Cu.CustomerType = 'I' AND Te.[Group] = 'Europe';
```

Figure 4.5: An example of applications we design

transaction is from, the **access type** (direct or through application) and the **SQL statement**. They will be further parsed for training and testing (see Table 5.2 and Table 5.3 for details).

4.4 False Positives and False Negatives

We have presented the abstract definitions of the false positive and false negative in Chapter 2. Here we explain these two concepts in more details.

The false positives and false negatives are wildly used to measure the accuracies of some actions. A typical example is that people, in medicine, often evaluate the accuracy of medical tests by measuring their false positives and false negatives. A false positive occurs when the test shows that a healthy person has AIDS; and if the patient indeed has AIDS but the test result is negative, then it is a false negative.

Additionally, people use the false positives and false negatives to measure the behaviors of intrusion detection systems, too. In detection mode, legitimate operations that are recognized as intrusions form false positives, and the intrusions that are considered legitimate become false negatives. In order to reflect the accuracies of detection behaviors of the IDSes more reasonably, people often calculate the false positive rate and false negative rate instead of only considering the numbers of false positives and that of false negatives. We present how to calculate the rates below:

$$False\ positive\ rate = \frac{number\ of\ false\ positives}{total\ number\ of\ negative\ instances\ (legitimate\ operations)} \tag{4.1}$$

$$False\ negative\ rate = \frac{number\ of\ false\ negatives}{total\ number\ of\ positive\ instances\ (intrusions)} \tag{4.2}$$

In the literature of IDS research, we find that most anomaly-based IDSes can achieve relatively low false negatives, but reducing the false positive rate looks more difficult. Therefore, particular attention is paid to how to decrease the false positive rate when we design our database IDS. We may use the terms false positive and false negative to represent false positive rate and false negative rate, respectively.

4.5 Chapter Summary

At the beginning of this chapter, we briefly introduced RBAC including its main components and two other concepts namely role hierarchy and abstract role. The database AdventureWorks and the role hierarchy we build for it are then presented, followed by the introduction to what features we collect for each transaction we monitor. At last, the false positives and false negatives we use to measure our system are explained. Comprehending this chapter well can substantially help understand our approach, proposed in the following two chapters, to building and testing our system.

Chapter 5
System Design

While building our system, the first problem we met is the lack of the data for training and tests. This is a quite common difficulty for inside threats research, because the companies usually do not agree to provide researchers with their real data, especially real cases of attacks, considering their reputations and stock prices. Therefore, our initial task is to generate reasonable data. After that, we need to design the system architecture of the IDS and choose the appropriate algorithms for training and detection. Finally, how to train the system and how to detect intrusions should be considered in quite detail. This chapter presents our approaches to dealing with the above problems.

5.1 Data Generation

5.1.1 Training and Testing Data Set

We initially designed 6 applications for the Marketing Department and 25 for the Sales Department, based on the scenarios of AdventureWorks (see Chapter 4.2 for details); the users and the permissions of invoking the applications are then assigned to the corresponding roles. A user is then allowed to invoke certain applications according to the permissions he/she has. Table 5.1 shows the role-permission assignment and user-role assignment. Because of the role hierarchy, some roles inherit permissions from their junior roles. For example, no permissions are directly assigned to the role Marketing Manager but it inherits all permissions that are assigned to the role Marketing Specialist and Marketing Assistant since the Marketing Manager is senior to both the Marketing Specialist and Marketing Assistant.

All transactions are presumed to be issued through the applications in our current system, but our IDS can be easily extended to be able to monitor users who interact with the database directly, such as DBAs. In the company, a day is divided into three work shifts - Day [7:00:00 to 15:00:00), Evening [15:00:00 to 23:00:00) and Night [23:00:00 to 7:00:00), and each user only works in his/her work shift. Meanwhile, we also assume each department has a unique IP address space, for example, 192.168.1.0 to 192.168.1.255, 192.168.2.0 to 192.168.2.255 belong to the Sales Department and the Marketing Department, respectively.

For each legitimate transaction, first, an employee in either the Sales or Marketing department is picked out randomly. After that, we randomly choose the time within the corresponding work shift (e.g. 10:16:02), the IP address (e.g. 192.168.1.79) within the employee's department's IP space and one application the user can invoke legally among all applications he/she is permitted to invoke. Finally, we assume that the user is not always interested in all attributes the application he/she invokes can access, so a non-empty subset of the attributes is randomly generated. In this way, a transaction is manufactured.

5.1.2 Intrusion Data Set

When RBAC is supported, we accept the assumption of [6, 15] that a transaction (R_i, T_i) becomes an anomaly if it is changed to (R_j, T_i) ($i \neq j$). So the first step of generating intrusions is to manufacture a set of legitimate transactions using the methods described in Chapter 5.1.1. With the consideration of role hierarchy, we change each transaction's associated RoleID to another one that is not equal or senior to the original one. The reason why the new role cannot be senior to the old one is that a senior role has all permissions the junior one has, as presented in Chapter 4.1.

RoleID	RoleName	Application	EmployeeID
5	Marketing Assistant	App1-MA, ..., App3-MA	2, 269, 272
6	Marketing Specialist	App1-MS, ..., App3-MS	46, 106, 119, 203, 271
7	Marketing Manager		6
8	SalesRepInEU	App1-EU, ..., App6-EU	285, 286, 289
9	SalesRepInNA	App1-NA, ..., App6-NA	275, 276, 277, 278, 279, 280, 281, 282, 283, 287
10	SalesRepInPA	App1-PA, ..., App6-PA	290
11	SalesManagerInEU	AppMng1-EU, AppMng2-EU	284
12	SalesManagerInNA	AppMng1-NA, AppMng2-NA	268
13	SalesManagerInPA	AppMng1-PA, AppMng2-PA	288
14	VPSale	App-VP	273

Table 5.1: Role-permission assignment and user-role assignment

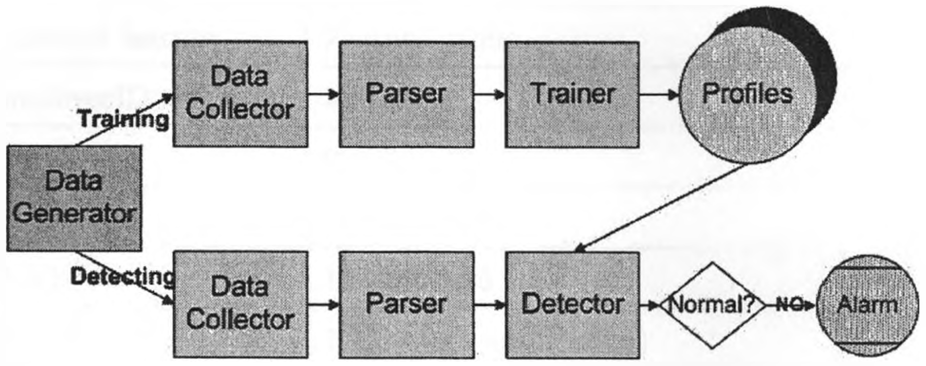


Figure 5.1: System working process

For example, Marketing Manager is senior to Marketing Analyst (see Figure 4.4), so if (Marketing Analyst, T_{MA}) is legal, (Marketing Manager, T_{MA}) must NOT be an intrusion.

For the user profiling IDS, we assume role information is unavailable, and therefore, we can only simply change the EmployeeID to a new one while generating an intrusion.

5.2 System Architecture

Up to now, we have transferred our work into a classification problem. The next challenge is to find a classifier, using which we can achieve relatively low false positives/negatives. We also hope the computational costs of the classifier are acceptable, especially for detection mode because we expect short latency when an intrusion occurs. This section and the next section describes the system architecture of our IDS and the classifier we use in detail, respectively.

Figure 5.1 shows the main components of our system, as well its working process. The **Data Generator** generates data for both training and testing. The **Data Collector** collects transactions containing features listed in Chapter 4.3. Each transaction collected is then passed to the **Parser** which further parses the transaction

Collected feature	Feature value
EmployeeID	287
RoleID	9
Time	10:32:09AM
IP address	192.168.1.95
AccessType	1 (through application)
SQL statement	SELECT S.Name FROM Sales.Store AS S JOIN Sales.Customer AS Cu ON S.CustomerID = Cu.CustomerID JOIN Sales.SalesTerritory AS Te ON Te.TerritoryID = Cu.TerritoryID WHERE Cu.CustomerType = 'S' AND Te.[Group] = 'Europe'

Table 5.2: An example of a collected transaction in raw format

and forms necessary features for training or detection. Its duties include changing the exact time the transaction is issued to the corresponding work shift and the exact IP address the transaction comes from to the corresponding DepartmentID. It also transfers the feature SQL statement to four features, including query type, referenced tables, the number of attributes in the answer and area constraints. We use a string to represent the referenced tables according to each table's number. Table 5.2 and Table 5.3 illustrate a collected transaction and its format after being parsed. The string *000001110000000000* in Table 5.3 representing the referenced tables (views) indicates that this sample transaction requests for the access to the Table No.5, Table No.6 and Table No.7 in the database (see Table 4.2 for details). Another point we should state is that the two Data Collectors in the training module and the detection module have the same functions, and so do the two Parsers.

Feature	Feature value
EmployeeID (ignored for role profiling)	287
RoleID (ignored for user profiling)	9
WorkShift	Day
DepartmentID	3 (Sales Department)
AccessType	1 (through application)
QueryType	SELECT
ReferencedTables	00000111000000000
NumberOfAttributes	1
AreaConstraint	Europe

Table 5.3: An example of a transaction after being parsed

5.3 Classifier

5.3.1 Decision Tree

The first machine learning classifier we considered is the decision tree which represents learned functions by a tree [21]. With the learned decision tree, the classification of an instance with the features is turned into sorting through the tree until arriving at the corresponding leaf node. The value of the leaf node is the DECISION. Figure 5.2 is a simple sample decision tree cited from [21]. This decision tree is used to make the decision whether people should go out to play tennis. For example, there are two days shown in Table 5.4. We sort though the nodes Outlook (Rain), Wind (Strong) and finally arrive at the decision NO for Day1; for Day2, we go though Outlook (Sunny), Humidity (Normal) and find the decision YES.

As we actually decided not to use the decision tree classifier at last, we will not introduce many details of its training algorithm, which involves a lot of information

Day	Outlook	Temperature	Humidity	Wind
Day1	Rain	Cool	High	Strong
Day2	Sunny	Mild	Normal	Weak

Table 5.4: Two sample days

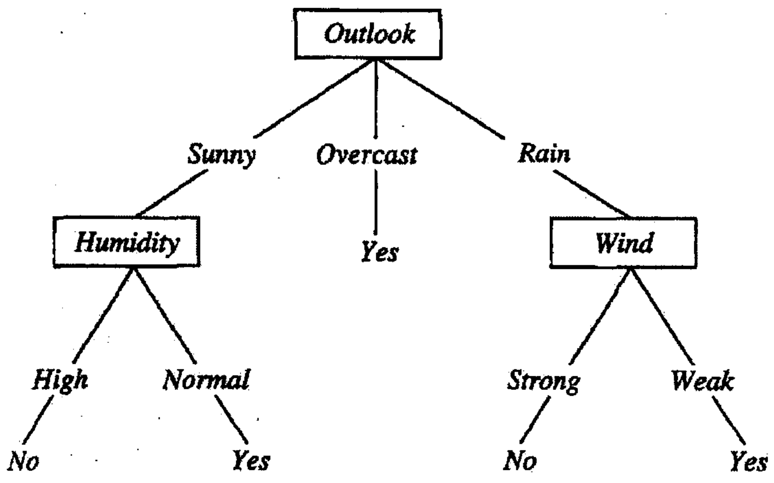


Figure 5.2: A sample decision tree (From [21])

theory knowledge (refer to [21] for details if interested); however, we focus on the reasons why we believe this classifier is not an appropriate one for our IDS in the following paragraphs.

In general, we can make use of the decision tree in two different ways. The first way is more traditional. We build a two outcome class (YES or NO) tree for each role/user. Then when a role/user issues a new transaction, we sort the corresponding decision tree to determine if it is an anomaly. The other way is to build only one tree but this tree has more outcome classes. For such decision tree, each role/user is a possible outcome so that the values of the leaf nodes are no longer only YES or NO, but the RoleIDs/UserIDs. Then every newly issued transaction is classified to a RoleID/UserID by sorting through the tree. If the one we get from the tree is different from the original one associated with the transaction, we raise an alarm.

Unfortunately, we finally tend to believe that both approaches are not satisfactory. The first problem with the former approach is related to the necessary training data. We need both positive and negative instances to build the decision tree in this case, but how can we get the positive instances (intrusions)? One approach is to manufacture the intrusions ourself; however, we find that this makes the behavior of the IDS mainly depend on the quality of the intrusion set, and that we can actually control its behavior by controlling the positive instance set and test data set. For instance, if we include some category of the intrusions in our training data, then the IDS can be very powerful when detecting this kind of intrusions. Obviously, we can reduce the false positives and false negatives just by feeding the transactions we have considered during the training mode to our IDS. This makes the tests meaningless. More importantly, it is very difficult for people to collect enough real intrusions for training purpose in reality. That is one of the reasons why most IDSes prefer the algorithms that require attack-free training samples.

The latter approach to exploiting the decision tree does work. However, the built tree is far too large (imagine how complex it is when we have thousands of users). Apparently, it is unrealistic to build such a decision tree for the individual users. Even when we have role information and build profiles for the roles, we still feel it is unbearable due to its low efficiency. In fact, people are usually only interested in the binary (two outcome classes) decision tree in practice.

5.3.2 Naive Bayes Classifier

Similar to [6, 15], we finally also chose a Naive Bayes classifier to build profiles in the training mode and detect intrusions in detection mode after we notice its advantages listed below. Firstly, its computational cost is quite low. Second, although a Naive Bayes classifier largely simplifies reality by assuming that all features of a class are completely unrelated to each other (independence assumption), it usually performs much more accurately than people expect. The reason is probably that as a probabilistic classifier using the Maximum A-posteriori Probability estimation, a Naive Bayes classifier can reach the right classification without getting accurate probabilities of classes as long as the correct class has higher probability than any other classes. Finally, a Naive Bayes classifier is robust to noise [21].

We present the mathematical background of a Naive Bayes classifier in this paragraph. Abstractly speaking, the probability model for a classifier is a conditional model:

$$p(C|F_1, ...F_n) \tag{5.1}$$

In this model, C is a class, H is the hypothesis space ($C \in H$) and F_i is one of the features that compose an instance x of the data. The perspective of the classifier is to find out the most probable class when an instance x consisting of features $(f_1, ..., f_n)$ is given. A decision rule is needed to combine with a classifier for classification; the most common decision rule is Maximum A-Posteriori (MAP). Using the MAP, the corresponding classifier is defined as follows:

$$classify(f_1, ..., f_n) = \underset{c \in H}{argmax}\ p(C = c|F_1 = f_1, ..., F_n = f_n) \tag{5.2}$$

However, the calculation of $p(C|F_1, ..., F_n)$ is very difficult. In many cases, it is actually infeasible. Therefore, we reformulate $p(C|F_1, ..., F_n)$ using Bayes' theorem:

$$p(C|F_1, ..., F_n) = \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ...F_n)} \tag{5.3}$$

We notice that the denominator of (5.3) is a constant when an instance x with features F_i is given, so we are only interested in the numerator which could be re-written as below:

$$\begin{aligned} & p(C)p(F_1, ..., F_n|C) \\ & = p(C)p(F_1|C)p(F_2, ..., F_n|C, F_1) \\ & = p(C)p(F_1|C)p(F_2|C, F_1)p(F_3, ..., F_n|C, F_1, F_2) \\ & = p(C)p(F_1|C)p(F_2|C, F_1)...p(F_n|C, F_1, F_2, F_3, F_{n-1}) \end{aligned} \tag{5.4}$$

The next step is to apply the independence assumption to simplify Equation (5.4). According to this assumption, F_i is completely independent from F_j when $i \neq j$, so we have $p(F_i|C, F_j) = p(F_i|C)$. Then the Equation (5.4) can be simplified to be:

$$\begin{aligned} & p(C)p(F_1, ..., F_n|C) \\ & = p(C)p(F_1|C)p(F_2|C, F_1)...p(F_n|C, F_1, F_2, F_3, F_{n-1}) \\ & = p(C)\prod_{i=1}^{n} p(F_i|C) \end{aligned} \tag{5.5}$$

The classifier now can be transfered to:

$$\begin{aligned} & classify(f_1, ..., f_n) \\ & = \underset{c \in H}{argmax} p(C = c|F_1 = f_1, ..., F_n = f_n) \\ & = \underset{c \in H}{argmax} \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ...F_n)} \\ & = \underset{c \in H}{argmax} \frac{p(C)\prod_{i=1}^{n} p(F_i|C)}{p(F_1 = f_2, ..., F_n = f_n)} \\ & = \underset{c \in H}{argmax} p(C)\prod_{i=1}^{n} p(F_i|C) \end{aligned} \tag{5.6}$$

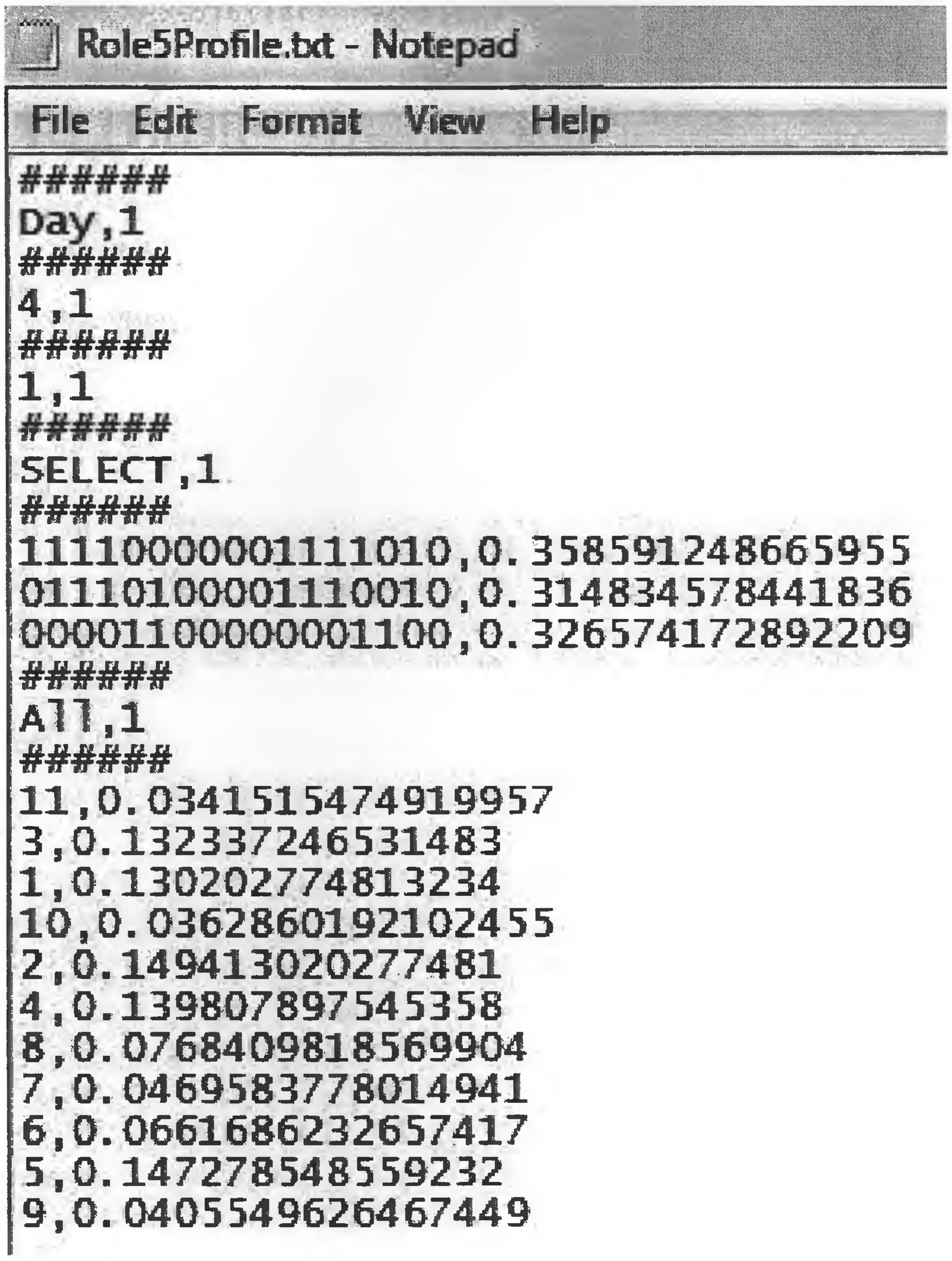
The general principle of a Naive Bayes classifier is stated above. When it is applied to our IDS, features of each transaction obtained from the Parser form the features F_1 to F_n in the Naive Bayes classifier; each role (for role profiling) or each user (for user profiling) is a class. In the training mode, we calculate for each class the possibility of each observed value of each feature, based on the training samples. The detection task here is then turned into finding the most possible role/user who may issue the transaction when a new transaction with the features is given, and check if it equals the original one associated with the transaction. If NOT, the transaction is recognized as anomalous.

5.4 User Profiling Vs. Role Profiling

Using Naive Bayes as our classifier, user profiling and role profiling are applied to build the IDS, respectively. For the user profiling IDS, we assume role information is unavailable, so we build profiles and detect intrusions based on EmployeeID. The role profiling IDS is quite similar to user profiling, and the only difference is that we use the RoleID instead of EmployeeID to construct profiles and detect anomalies.

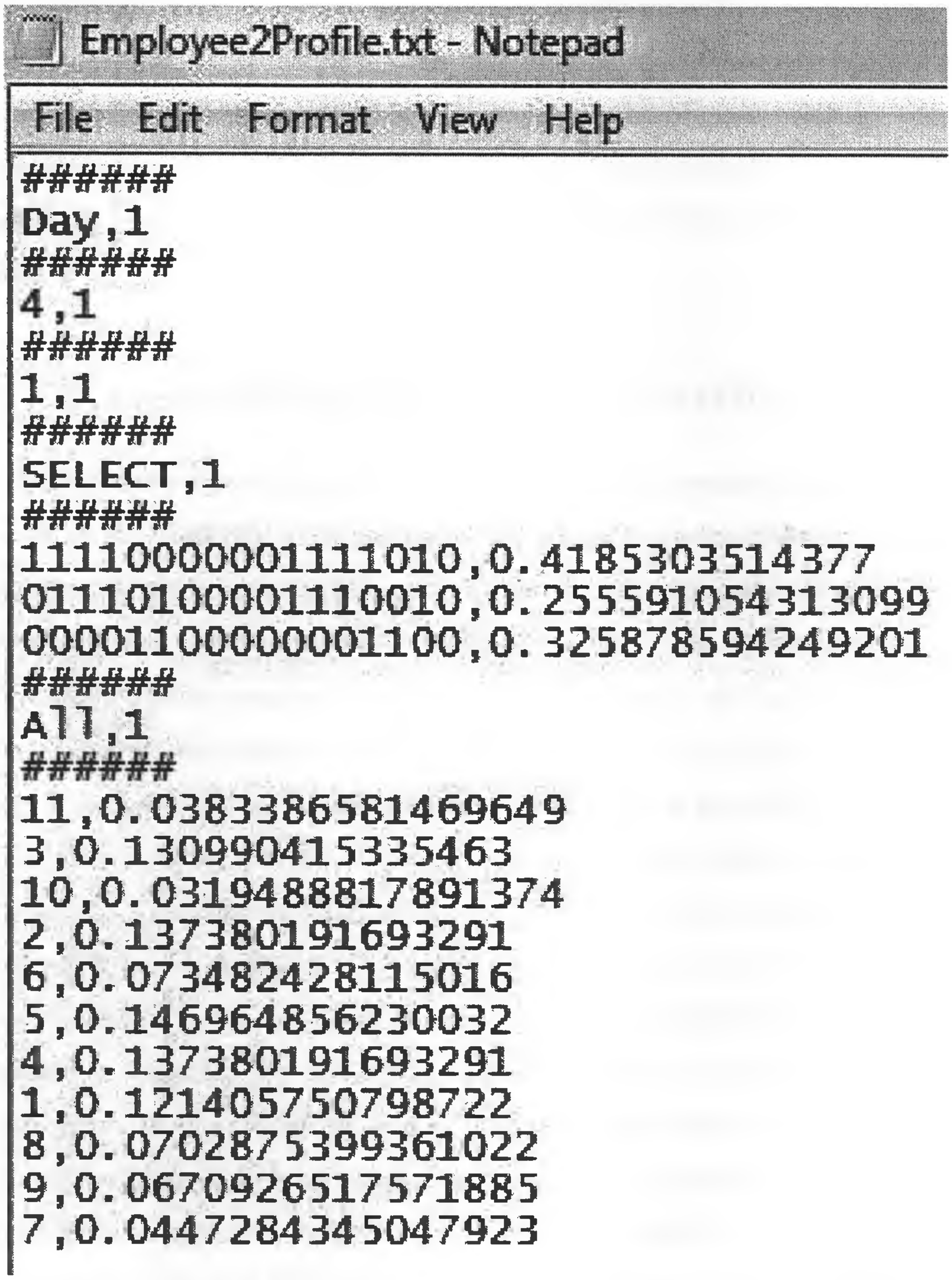
The detailed profiling work is very straightforward. We calculate the probability of each observed value of each feature for every role/user and write them down in files (profiles). As only the values observed of each role/user appear in the profiles, the number of values and the values themselves of every two roles/users can possibly vary a lot. Figure 5.3 (RoleID = 5) and Figure 5.4 (EmployeeID = 2) are the examples of a role's profile and a user's profile, respectively. We use the strings ###### to separate the values of the features; each line of the profile (except the ######) includes the value and its probability associated with the corresponding role/user. For instance, when we look at the fifth feature of the role whose profile is presented in Figure 5.3, we can find that this role (actually the users assigned to this role) invokes three different applications with the probabilities of 0.358591248665955, 0.314834578441836, 0.326574172892209, respectively, during the training period.

There are two reasons why we add several strings ###### into the profiles: the first one is simply because we want the values of one feature to be separated from other values so as to make the profiles easier to read; the second one is to build the indexes for each ###### (actually for each feature) so that we can locate them faster when we need to look at the values of some specific feature during the detection. In the detection mode, the IDS imports all profiles and their indexes in advance and keeps them in the program memory all the time. This can decease the times of reading files (reading files is very time-consuming). Then every new transaction will be checked by calculating the possibility of each c (a role or a user) and finding



```
######
Day,1
######
4,1
######
1,1
######
SELECT,1
######
1111000000111010,0.358591248665955
01110100001110010,0.314834578441836
00001100000001100,0.326574172892209
######
All,1
######
11,0.0341515474919957
3,0.132337246531483
1,0.130202774813234
10,0.0362860192102455
2,0.149413020277481
4,0.139807897545358
8,0.0768409818569904
7,0.0469583778014941
6,0.0661686232657417
5,0.147278548559232
9,0.0405549626467449
```

Figure 5.3: An example of one role's profile



```
######
Day,1
######
4,1
######
1,1
######
SELECT,1
######
1111000001111010,0.4185303514377
0111010000111001 0,0.255591054313099
0000110000001100,0.325878594249201
######
All,1
######
11,0.0383386581469649
3,0.130990415335463
10,0.0319488817891374
2,0.137380191693291
6,0.0734824281150 16
5,0.146964856230032
4,0.137380191693291
1,0.121405750798722
8,0.0702875399361022
9,0.0670926517571885
7,0.0447284345047923
```

Figure 5.4: An example of one user's profile

the one with biggest possibility. Notice that the probability of a new value that cannot be found in the corresponding profile is simply set as 0; if we find the $p(C = c|F_1 = f_1, ..., F_i = f_i)$ for EVERY possible c in the hypothesis space is 0, an alarm is directly raised. Then as stated above, whenever the original role/user (denoted as $ORIGINAL_R/ORIGINAL_U$) differs from the most probable one obtained by using the classifier and MAP decision rule (denoted as MAP_R/MAP_U), an anomaly is detected.

5.5 Role Profiling with Role Hierarchy

In this chapter, we present our novel intrusion detection approach taking into consideration the role hierarchy. It is based on the role profiling described in Chapter 5.4 while an extra rule is applied. Using role profiling, apparently when a role R_H issues a transaction and the Naive Bayes classifier and MAP rule say the most probable role that issue the transaction is another role R_L while $R_L \neq R_H$, an alarm will be raised. However, we notice that if $R_L \prec R_H$ (means R_L and R_H are comparable and R_L is junior to R_H in the role hierarchy), the alarm is probably a false positive. The reason is that R_H inherits all permissions R_L has, and when a user assigned to R_H exploits the permission that R_H inherits from R_L, we can consider that this user is now acting as a member of R_L. In this case, we can say the real role associated with a transaction is the role equal or junior to the role to which the user issuing the transaction is assigned. We also need to point out that we are interested in the real role only when $MAP_R \neq ORIGINAL_R$. This is because when a user assigned to R_H is taking advantage of R_L's permission and acting as a member of R_L, there is still possibility that the $MAP_R = R_H$, and certainly, this situation is legitimate.

In summary, the new IDS performs exactly the same as the IDS using role profiling (refer to Chapter 5.4 for details) when $MAP_R = ORIGINAL_R$. The system will check if $MAP_R \prec ORIGINAL_R$ when $MAP_R \neq ORIGINAL_R$, and an alarm will be raised only if MAP_R is not junior to $ORIGINAL_R$. In the following chapters, we use the terms **user profiling**, **simple role profiling** (no role hierarchy) and **advanced role profiling** (with role hierarchy) to represent the three profiling approaches mentioned above.

5.6 Chapter Summary

This chapter explained the details of our IDS. Starting with our approach to generating data for both training and detection modules, we described the architecture of our system, including its main components and their functionalities, too. Our effort then fell into finding an effective and efficient classifier for the IDS. The Decision Tree was briefly described, followed by the explanation about why it does not work well. We then described the general principals of a Naive Bayes classifier we finally chose as our classifier and how this classifier was applied to the user profiling and the role profiling for both training and detection purposes. We ended this chapter by presenting our novel advanced role profiling approach. The detailed evaluations of the system will be shown in the next chapter.

Chapter 6 Experimental Evaluation and Related Analysis

This chapter presents the detailed results of our evaluations. The primary objectives are to measure the false positives and false negatives of our IDS when using user profiling, simple role profiling and our advanced role profiling, respectively, and to test the overhead to the original database system. A significant principle of the testing is to make the comparisons relatively fair.

We describe our test methodologies as follows. First, we evaluate the three profiling approaches by measuring the **false positives** and **false negatives** each approach causes. Training sets containing different numbers of transactions are used to build the profiles using user profiling, simple role profiling and advanced role profiling, respectively. We then generate 800 illegitimate transactions for false negatives testing and 1688 legitimate transactions to test their false positives. How we generate data in detail can be found in Chapter 5.1. Second, attention is paid to the **overhead** compared to the original database system. Although we plan to design the system architecture that can conduct real-time detection in our future work, currently we simply test the detection time for some numbers of transactions and their response time without the deployment of the real-time architecture. We can predict that the potential overhead should be light if the detection time is short compared with the response time. Finally, we focus on training time.

6.1 False Negatives/Positives Test Results and Discussion

6.1.1 False Negatives/Positives Test Results

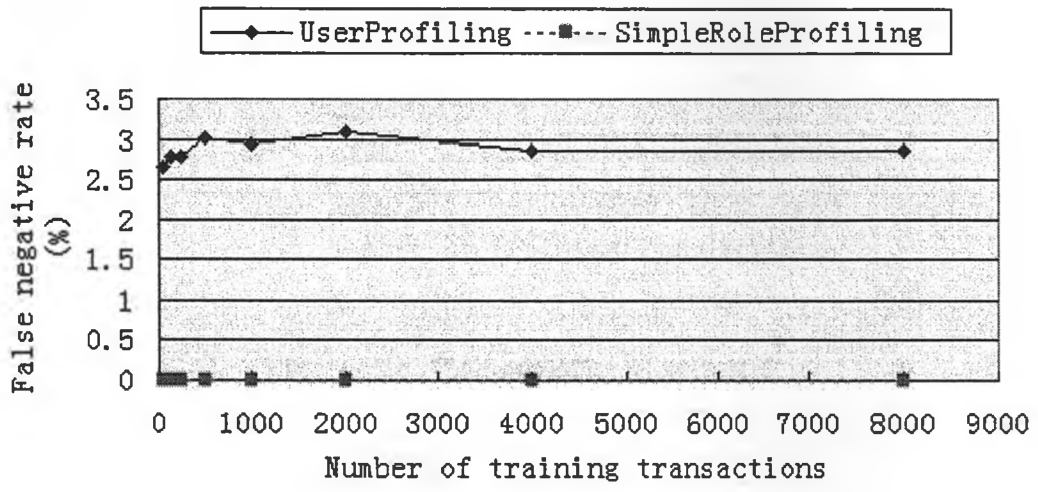


Figure 6.1: False negative rate test 1

Figure 6.1 and Figure 6.2 show the false negatives testing results. Generally speaking, either profiling approach achieves low false negative rate, which illustrates that they can detect most of anomalies. Moreover, we find the false negatives remain 0 even when we have only as few as 50 training transactions for either simple role profiling or our advanced role profiling. We present the false positives with respect to the three profiling approaches in Figure 6.3. While many anomaly-based IDSes can arrive at relatively low false negatives, low false positive continues to be a difficult objective for this category of IDSes. As expected, user profiling results in the much poorer performance (seems almost useless in practice) compared with the other two alternatives. Unsurprisingly, the false positive rate of simple role profiling is very near the false positive rate in [6, 15] in which similar role profiling is used. Our advanced role profiling approach, however, largely improves the performance of the IDS. We

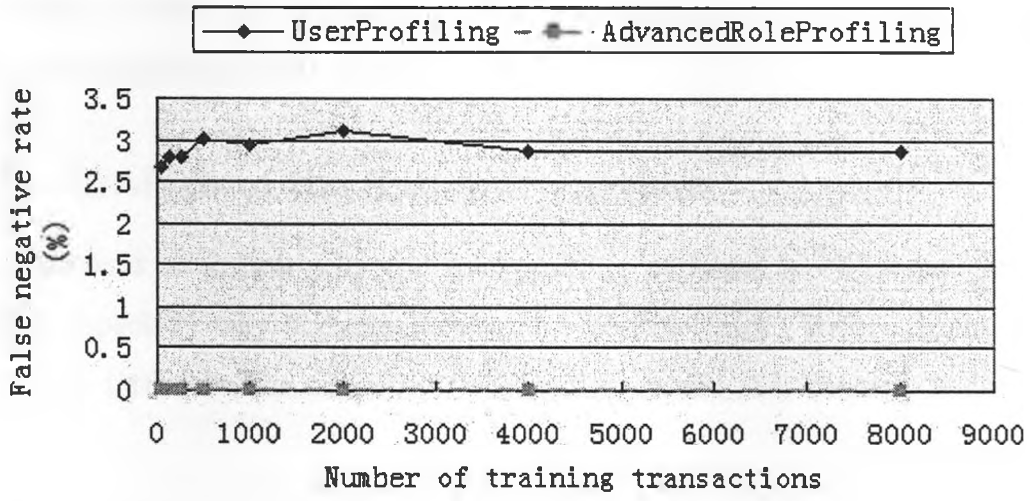


Figure 6.2: False negative rate test 2

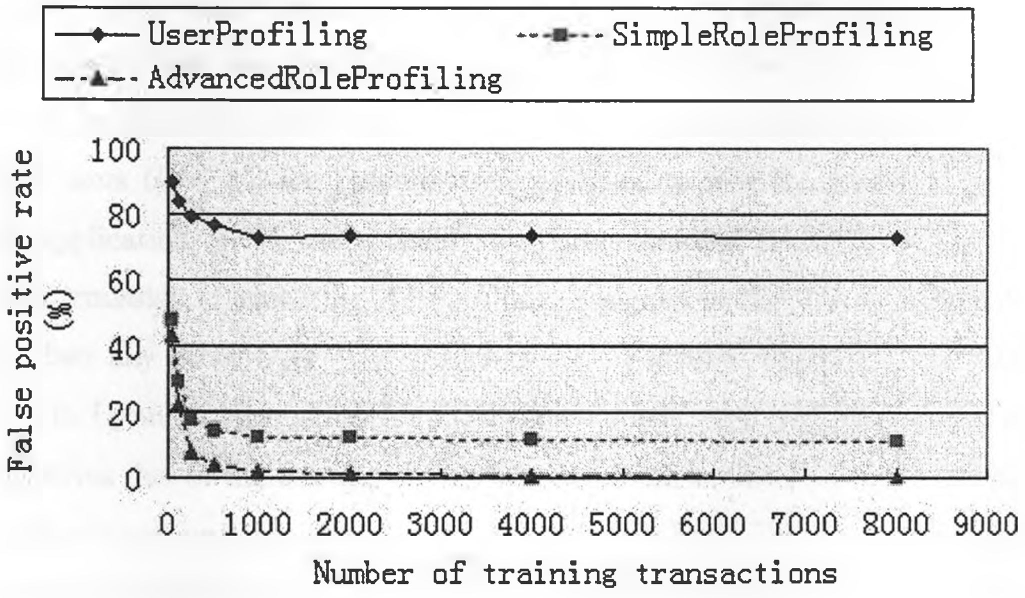


Figure 6.3: False positive rate test

find the false positive rate of our approach drops dramatically to a very low level with the increase of the training samples, and it reaches as low as 0.24% when we have a training set containing 8000 transactions.

6.1.2 Analysis of the System Behaviors

It is significant to answer why the role profiling performs much better than user profiling, especially why it causes lower false positive rates. Before discussing the reasons, we introduce the concept of kindred users and this concept is defined as follows:

Definition. Kindred users *in role-based access control are users who are assigned to the same role.*

Apparently, the kindred users have many permissions in common and often do many similar operations. Therefore, their probabilities of the values of the features can be quite near each other, and in this case, when a user issues a transaction, there is a big chance that the Naive Bayes classifier mis-recognizes it as the behavior of one of his/her kindred users. Besides, we can also explain it in a simplified way. For instance, users $U_1 \sim U_n$ are kindred users assigned to role R_x, and $U_i(1 \leq i \leq n)$ invokes application APP_a more times than other kindred users do in the training set (the permission of executing APP_a is only assigned to R_x); then in the detection mode, when any other $U_j(1 \leq j \leq n \text{ and } j \neq i)$ invokes the APP_a, the IDS will match it to U_i and a false alarm is raised. In summary, user profiling causes a lot of false positives due to the mis-matchings between kindred users when we assume role information is not available. Further more, even when RBAC is indeed not supported, there must be some users who have similar operation duties, and mis-matchings can occur frequently among them.

We have just explained why user profiling results in more false positives, and notice that the mis-matchings among kindred users are the primary reasons. Then is it possible that there is also high mis-matching rate among roles? Unfortunately, this may happen if we do not design roles reasonably and have too many "similar roles". Therefore, we expect good design of roles for the satisfactory behaviors of the IDS. The success of our advanced role profiling can strongly highlight our previous statement that IDS should be a supplemental mechanism but not a replacement to traditional security mechanisms because the IDS can perform much better when working together with other mechanisms. Security is a hybrid problem, and we must even think about many non-technological aspects such as policy making and human factors. One principle we must remember is that we should not expect any single security mechanism to perform perfectly for data protection.

We have stated why our advanced role profiling approach could reduce the false positives. Here we give more detailed explanations about how the false positives reduced by advanced role profiling occur so that people can understand why we can reduce false positives with a check of the role hierarchy. When a role R_H is senior to another role R_L, and when a user U_i assigned to R_H is exploiting the permissions that R_H inherits from R_L, U_i can be viewed as an acting member of R_L. Certainly, U_i can do it legally, and the system raises a false alarm if it categorizes the transaction into R_L and does not check the role hierarchy. In addition, we can use another simplified case to exemplify this issue, too. We assume two users U_h and U_l are members of the roles R_H and R_L, respectively, and R_L gains the permission of invoking application APP_b; U_h invokes APP_b m times and U_l does that n times ($m \prec n$) according to training data. We then find when U_h invokes APP_b later, it will be mis-matched to R_L with an alarm being raised. Therefore, it is meaningful for the IDS to conduct extra checks of the role hierarchy so that we can prevent the IDS from raising the category of false alarms explained above.

6.1.3 Training Threshold

Another crucial discovery is that the false positive rates decrease much faster when there are less than 1000 training transactions than when we have more than 1000. Actually, with more than 1000 training samples, the false positives, for all three profiling approaches, maintain their stabilities even while the training set keeps growing. Obviously, 1000 is a diving line (probably this number varies in other IDSes), and we name this number of training samples the **training threshold**. This indicates two points: firstly, we need enough training data to exceed the training threshold in order to achieve ideal behavior of the IDS; secondly, we are able to find out the reasonable trade-off between the detection capability and the costs of data collection and training.

6.2 Overhead

The performance test of our IDS is conducted to quantify the overhead. We generate five testing sets containing different numbers of transactions, and in each set, 10% are intrusions while others are normal. Using these testing sets, we firstly test the query response time of the original database system without the deployment of the IDS; then we test the system's examining time when using user profiling (denoted as UP), simple role profiling (SRP) and advanced role profiling (ARP), respectively. The result (measured in second) is shown in Table 6.1. Obviously, the system's examining time is very short compared with the query response time whichever profiling approach among the three is used, due to the low computational task of a Naive Bayes classifier. Additionally, we can find that using role profiling requires less examining time than using user profiling. That is because the hypothesis space of the former approach is smaller than that of the latter one (the number of roles is smaller than the number of users). Moreover, using advanced role profiling costs slightly more time than using simple role profiling because of the extra process of checking the role hierarchy; we,

Samples	Query	UP	SRP	ARP
1000	227.313	0.625	0.625	0.625
2000	457.375	0.141	0.094	0.094
3000	678.321	0.203	0.141	0.156
4000	945.578	0.281	0.188	0.203
5000	1080.239	0.328	0.250	0.250

Table 6.1: Performance test (in Sec)

however, find the extra checking time is quite short (sometimes even too short to be reflected by the computer).

6.3 Training Time

Although we feel that it is bearable even the training time is not so short for IDSes, we do still like shorter training time, especially if we want our IDS to be able to update dynamically and automatically. For example, we may prefer to re-train our system every m hours or when every n new transactions are collected. This approach makes the IDS 'understand' better the users'/roles' most current behaviors, enhancing its detection accuracy. However, a significant side affect of this approach is that we probably have to temporarily shut down the detection service of our IDS during the re-training processes. Certainly, we hope that this no-service time is as short as possible. So obviously, shorter training time can result in fewer side affects while a dynamic IDS is deployed.

We present the comparison of the training time in Figure 6.4. Please notice that the training data collected is separated by the RoleIDs/EmployeeIDs in advance. This is infeasible if we directly use the log files provided by the DBMS since the DBMS cannot separate the transaction records by the RoleIDs or EmployeeIDs; in this case, it is predictable that longer training time is necessary. However, we make the pre-

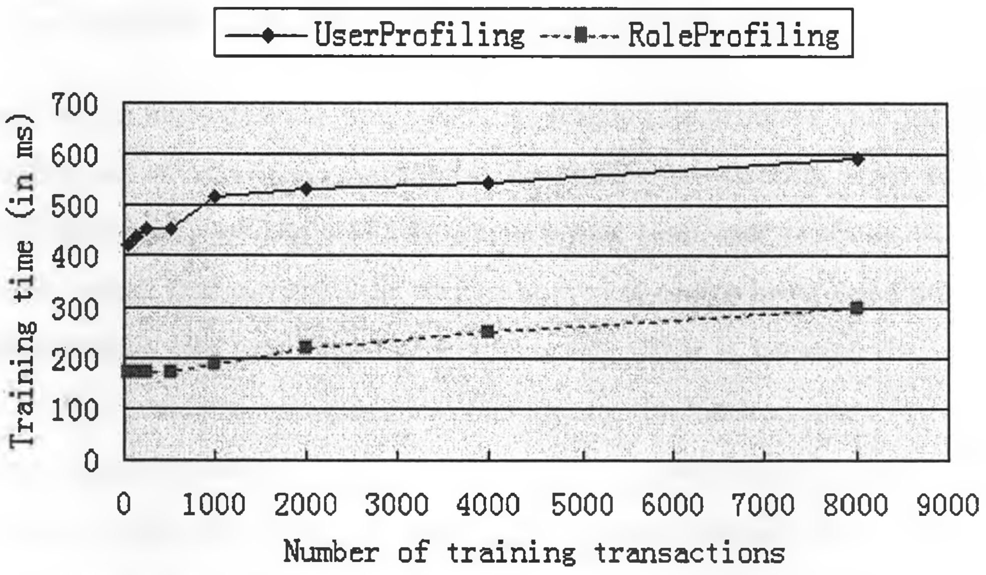


Figure 6.4: Training time

separation work by using our own data collection mechanism. As stated in Chapter 4.3, we collect 6 features, including EmployeeID and RoleID, for each transaction. If we plan to use role profiling to build the IDS, we construct a log file for each role; then when a transaction is newly issued, it is directly recorded into the log file corresponding to the role who issues the transaction. Similarly, we do this for the users if user profiling is going to be used. Therefore, each role/user owns its unique log file. Unsurprisingly, role profiling requires less training time due to the fewer profiles that we need to build. Also because of fewer profiles needed, the overhead becomes lighter (see Chapter 6.2). Actually, the preference of fewer necessary profiles forms people's original interest of introducing RBAC into the database IDS. We would like to highlight this point again because this is a very meaningful advantage of role profiling if the IDS is tailored to a very large database with relatively small role-user ratio (for instance, we have only 30 roles but 1000 users in total in the database).

6.4 Chapter Summary

This chapter has evaluated the user profiling, simple role profiling and our advanced role profiling, all of which can be potentially applied to our IDS. The results have illustrated that role profiling performs much better than user profiling when examining newly issued transactions (the former approach causes lower false positive rate and false negative rate than the latter approach). This is because the role profiling can largely avoid the mis-classifications among the kindred users (or users who have many duties in common). Meanwhile, we have found that role profiling requires shorter training time and adds lighter overhead to the original system, too, resulting from fewer profiles we need (lower computational costs for both training and classification). This also makes role profiling preferred when we design the IDSes for the databases of those very large organizations.

Additionally, we have also found that our advanced role profiling can remarkably reduce the false positive rate while maintaining the extremely low false negative rate, and that it causes almost no extra overhead compared with the simple role profiling. Moreover, we also introduced the concept called training threshold so as to balance the behavior of the IDS and the costs of collecting data and training the system.

Finally, we want to stress that the test results prove that the data we generate can simulate real application well. The reason is that the test results of our system, while using simple role profiling are quite near the results obtained in [6, 15] where real data was used for tests (the profiling approaches used in [6, 15] are similar to the simple role profiling approach). Therefore, we believe that our conclusions about the three profiling approaches, based on our evaluation results, are trustworthy.

Chapter 7
Conclusions

In this chapter, we present the conclusion of our research. We begin with the summary of our work, followed by the outline of our primary contributions. We discuss some possible future direction of our research at the end.

7.1 Summary

Database security problems seriously threaten organizations storing invaluable data in their databases. An intrusion detection system provides another layer of security; however, few IDSes are specifically tailored to the DBMS so that their behaviors of detecting intrusions or misuses towards databases are quite poor. Therefore, in this thesis we present an anomaly-based IDS that takes into account the characteristics of DBMS, and our system is able to monitor every inside user continually so that insider threats can be substantially lessened.

We generate necessary data for both the training and detection modules at first. After that, our attention is turned to the system design. We apply a Naive Bayes classifier to build profiles in advance, and each new behavior is compared against the profiles in order to find out the most probable role/user ($MAP_{role/user}$) who may issue the transaction. An alarm will be raised as long as the $MAP_{role/user}$ is different from the original one associated with the transaction (with a check of the role hierarchy added for the advanced role profiling).

Although RBAC is supported in the DBMS we use, we initially built profiles for users (user profiling) by assuming role information is unavailable; then we take advantage of the RBAC and build our system using simple role profiling (without role hierarchy) and advanced role profiling (with role hierarchy).

By comparing our system's (when simple role profiling is used) test results with those in [6, 15] in which they use a profiling approach similar to the simple role profiling to build their system (real data is tested), we notice that the test results of the two IDSes are close. Therefore, we believe our data can simulate the real data ideally and our test results are trustworthy. Our evaluations also illustrate that role profiling is more effective (lower false positives/negatives) and more efficient (less training time) than user profiling in general; the results also prove that using the advanced role profiling we present can further improve the behaviors of the IDS since the false positives are reduced on a large scale again. We also prefer to point out that our IDS can monitor the DBAs as well by building profiles for the role DBA, although we have not done so yet.

The final point we need to highlight is that the database IDS can provide database systems with a supplementary security layer; however, it cannot replace the traditional database security mechanisms. In addition, the IDS can perform much better if it can exploit some features of the traditional security mechanisms such as RBAC.

7.2 Contributions

We propose a prototype of the anomaly-based database intrusion detection system in this thesis. In particular, we have made the following contributions:

- We develop a method that can be used to generate the data which can simulate real applications well.

- While looking for a suitable classifier, we analyze the reasons why the Decision

Tree is not satisfactory but the Naive Bayes classifier can satisfy our needs.

- By conducting detailed and fair comparisons between the user profiling and the role profiling, we prove that role profiling is more effective and more efficient than user profiling; the concept Kindred User is introduced in order to explain why (when role information is not available, there are still users having many duties in common).
- A novel advanced role profiling approach is introduced, taking the role hierarchy into account. This approach can largely reduce false positives.

7.3 Future Work

Our method has shown its promising applicability. We are currently extending our work in various directions.

7.3.1 Mine Referenced Tables From Collected Data

The referenced tables of each transaction are represented by a string in our system. Apparently, We need to number all the tables that are referenced in advance. This task is done manually. However, we have to admit that our current method affects the generic applicability of our system because we have to re-number the tables manually every time the IDS is deployed for a new database system. One solutions is to simply number all tables in the database, but we notice that many tables are never referenced by the applications provided to the users. Another fact is that the longer the strings are, the less efficient the IDS is.

In this case, shorter strings are preferred (only the tables really referenced are numbered). We are planning to develop a tool that can mine all the tables that appear in the collected data files (our own log files) automatically. Every distinct table (and only these tables) in our log files will be numbered.

7.3.2 Roles with Overlapped Permissions

Sometimes roles may have a lot of overlap (have more permissions in common among roles) in practice. This can potentially affect the behavior of the IDS. Therefore, we plan to set up another experiment to compare how well the simple and advanced role-profiling work when there are more closer roles.

7.3.3 Monitor the DBAs

Another significant direction is to pay extra attentions to the DBAs. Although our system is able to profile the DBAs and monitor them (treat them the same as the average users), we must notice that the DBAs have more flexible behaviors than average inside users. So our current system may result in higher false positives/negatives while examining DBAs' behaviors. However, there are some unique characteristics of the DBAs that we can take advantage of; for example, they are usually not expected to access the detailed data within user tables. We can check if a DBA is trying to do so, and if yes, an alarm is raised. We can apply such additional rules for DBAs to improve our IDS.

7.3.4 Real-time Detection

Additionally, we will develop a system architecture, based on which the new transactions can be checked in real-time and very little overhead will be caused to the original database system. Extra hardware may be necessary in order to support the new architecture. A rough architecture is proposed in Figure 7.1.

DBAs are assumed to be able to access the database server directly or through the network. We use the **Sniffers** to catch all transactions transferred through the network between the users and the database server. Therefore, the database server and the IDS can work concurrently so that the real-time detection for these users becomes possible. For DBAs' direct-access operations, the real-time detection is more

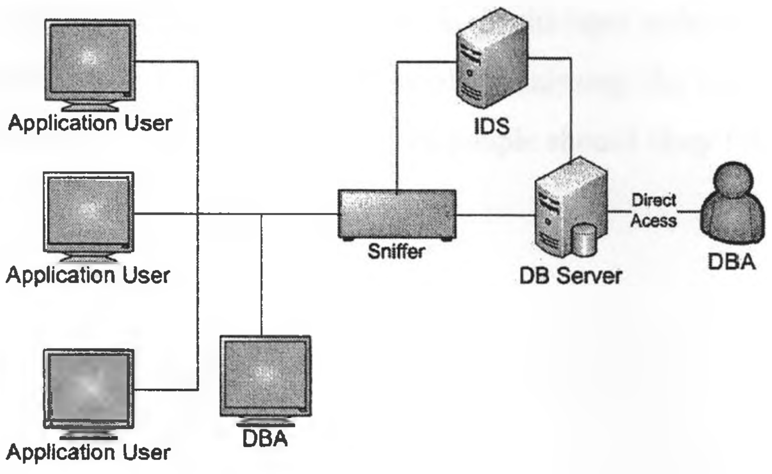


Figure 7.1: System architecture for real-time detection

difficult. Probably we must modify the DBMS; in this case, only when we use those open source DBMS, real-time detection can be achieved.

7.3.5 Response Mechanism

Part of our future work will be related to investigating response mechanisms. We hope that the IDS can take actions automatically and appropriately even when it is the DBA who conducts an attack.

7.3.6 More Ambitious Plans

Our further objective is that we want as few insider threats as possible. To achieve this goal, we should have a deeper understanding about insider threats. This is a hybrid topic which can be divided into two sub-topics: non-technology and technology. The former one is related to the company organization and policy making. The technology used to control insider threats can be further divided into two aspects: prevention and detection [36]. From the point of view of prevention (mainly access control), we hope that access control is flexible enough but the insiders are still unable to abuse their privileges. When the access control fails to deny the unwanted accesses, we need the detection mechanism (IDS) to do this work. Multi-layer security can help reduce the insider threats. Our future work will involve analyzing the insider threats more comprehensively and proposing the principles people should obey for the decrease of insider threats.

Bibliography

[1] Land attack. `http://en.kioskea.net/contents/attaques/attaque-land.php3`, October 2008.

[2] D Allchin. Error types. *Perspectives on Science*, 9(1):38–58, 2001.

[3] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P Anderson Co., Fort Washington, PA, April 1980.

[4] Steven M. Bellovin. The insider attack problem nature and scope. In S.J. Stolfo, S.M. Bellovin, S. Hershkop, A.D. Keromytis, S. Sinclair, and S.W. Smith, editors, *Insider Attack and Cyber Security: Beyond the Hacker.* Springer-Verlag, 2008.

[5] Ron Ben-Natan. *Implementing Database Security and Auditing.* Elsevier, 2005.

[6] Elisa Bertino, Ashish Kamra, Evimaria Terzi, and Athena Vakali. Intrusion detection in RBAC-administered databases. In *ACSAC*, pages 170–182. IEEE Computer Society, 2005.

[7] Elisa Bertino, Teodoro Leggieri, and Evimaria Terzi. Securing dbms: Characterizing and detecting query floods. In Kan Zhang and Yuliang Zheng, editors, *ISC*, volume 3225 of *Lecture Notes in Computer Science*, pages 195–206. Springer, 2004.

[8] Christina Yip Chung, Michael Gertz, and Karl N. Levitt. DEMIDS: A misuse detection system for database systems. In Margaret E. van Biene-Hershey and Leon Strous, editors, *Integrity and Internal Control in Information Systems,*

IFIP TC11 Working Group 11.5, Third Working Conference on Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control, Amsterdam, The Netherlands, November 18-19, 1999, volume 165 of *IFIP Conference Proceedings*, pages 159–178. Kluwer, 1999.

[9] Dorothy E. Denning. An intrusion-detection model. In *IEEE Symposium on Security and Privacy*, pages 118–133, 1986.

[10] D. Ferraiolo and R. Kuhn. Role-based access controls. In *Proc. 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[11] Gordon, L. A. Loeb, M. P. Lucyshyn, and R. W. Richardson. The 2005 csi/fbi computer crime and security survey. *COMPUTER SECURITY JOURNAL*, 21(3), 2005.

[12] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, University of New Mexico, Department of Computer Science, August 1990.

[13] Albert J. Hoglund, Kimmo Hatonen, and Antti S. Sorvari. Computer host-based user anomaly detection system using the self-organizing map. In *Proceedings of the International Joint Conference on Neural Networks*, volume 5, pages 411–416, Piscataway, NJ, 2000. IEEE.

[14] American National Standards Institute. For information technology - role-based access control. ANSI INCITS 359, Jan 2004.

[15] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. Detecting anomalous access patterns in relational databases. *VLDB Journal: Very Large Data Bases*, 17(5):1063–1077, August 2008.

[16] Sandeep Kumar. *Classification and Detection of Computer Intrusions.* PhD thesis, Purdue University, August 1995.

[17] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Daniel Barbará and Chandrika Kamath, editors, *SDM*, pages 25–36. SIAM, 2003.

[18] Sin Yeung Lee, Wai Lup Low, and Pei Yuen Wong. Learning fingerprints for a database intrusion detection system. In D. Gollmann, G. Karjoth, and M. Waidner, editors, *Proceedings of the Seventh ESORICS*, volume 2502 of *Lecture Notes in Computer Science*, pages 264–280, Zurich, Switzerland, October 2002. Springer-Verlag.

[19] V. Lee, J. Stankovic, and S. Son. Intrusion detection in real-time database systems via time signatures. In *Proceedings of the Sixth IEEE Real-Time Technology and Applications Symposium (RTAS '00)*, pages 124–133, Washington - Brussels - Tokyo, June 2000. IEEE.

[20] Wai Lup Low, Joseph Lee, and Peter Teoh. DIDAFIT: Detecting intrusions in databases through fingerprinting transactions. In *ICEIS*, pages 121–128, 2002.

[21] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[22] Microsoft MSDN. Adventureworks sample OLTP database. `http://msdn.microsoft.com/en-us/library/ms124659.aspx`, February 2009.

[23] Jerzy Neyman and Egon Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part i. *Biometrika*, 20A:175–240, 1928.

[24] Matunda Nyanchama and Sylvia L. Osborn. The role graph model. In *ACM Workshop on Role-Based Access Control*, 1995.

[25] Matunda Nyanchama and Sylvia L. Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1):3–33, February 1999.

[26] Sylvia L. Osborn. Role-based acess control. In Milan Petkovic and Willem Jonker, editors, *Security, Privacy and Trust in Modern Data Management.* Springer, 2007.

[27] Sylvia L. Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 3(2):85–106, May 2000.

[28] Donn B. Parker. *Crime by Computer.* Charles Scribner's Sons, New York, 1 edition, 1976.

[29] P. Ramasubramanian and A. Kannan. An active rule based approach to database security in e-commerce systems using temporal constraints. In *Proc. of TENCON 2003*, volume 3, pages 1148–1152, 2003.

[30] P. Ramasubramanian and A. Kannan. Intelligent multi-agent based database hybrid intrusion prevention system. In *Proce. of ADBIS 2004*, volume 3255 of *Lecture Notes in Computer Science*, pages 393–408. Springer, 2004.

[31] P. Ramasubramanian and A. Kannan. Quickprop neural network ensemble. In *Proc. of WSEAS 2004*, Miami, Florida, USA, 2004.

[32] P. Ramasubramanian and A. Kannan. Quickprop neural network short-term forecasting framework for a database intrusion prediction system. In *Proc. of ICAISC 2004*, volume 3070, pages 847–852. Springer, 2004.

[33] P. Ramasubramanian and A. Kannan. Intelligent multi-agent based multivariate statistical framework for database intrusion prevention system. *Int. Arab J. Inf. Technol*, 2(3):239–247, 2005.

[34] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.

[35] E. Eugene Schultz. Intrusion prevention. *Computers & Security*, 23(4):265–266, 2004.

[36] Sara Sinclair and Sean W. Smith. Preventative directions for insider threat mitigation via access control. In S.J. Stolfo, S.M. Bellovin, S. Hershkop, A.D. Keromytis, S. Sinclair, and S.W. Smith, editors, *Insider Attack and Cyber Security: Beyond the Hacker*. Springer-Verlag, 2008.

[37] Fredrik Valeur, Darren Mutz, and Giovanni Vigna. A learning-based approach to the detection of SQL attacks. In *Proc. of DIMVA 2005*, volume 3548 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2005.