A FRAMEWORK FOR MONITORING INSIDER MISUSE OF IT APPLICATIONS

Aung Htike Phyo, Steven Furnell and Emmanuel Ifeachor

University of Plymouth

Drake Circus, Plymouth, Devon, United Kingdom

Tel: +44 1752 233520

Email: nrg@plymouth.ac.uk

ABSTRACT

Many security incidents involve legitimate users who misuse their existing privileges, such that they have the system-level right to perform an action, but not the moral or ethical rights to do so. Current Intrusion Detection Systems are ineffective in this context, because they do not have knowledge of user responsibilities, the normal working scope for a particular position, or the separation of duties that should be enforced. This paper outlines a novel framework for solving the problem of insider misuse monitoring. The approach argues that users with similar roles and responsibilities will exhibit similar behaviour within the system, enabling any activity that deviates from the normal profile to be flagged for further examination. Established access control principles are utilised for defining user roles, and the relationships between them, and a misuse-monitoring agent is proposed that will police application-level activities for signs of unauthorised behaviour. Practical implementation of the conceptual framework is considered in the context of a database environment.

KEY WORDS

Intrusion Detection, Misuse Detection, Insider Misuse, Misfeasor Monitoring, Role-based Monitoring, Privilege Abuse

A FRAMEWORK FOR MONITORING INSIDER MISUSE OF IT APPLICATIONS

1 INTRODUCTION

A traditional view of security relates to the specification of user access rights, and the subsequent prevention of violations that would denote unauthorised activities. However, insiders within an organisation will often have legitimate access to sensitive resources, and may therefore have the opportunity to commit misuse within the scope of their existing access rights and privileges. In many contexts it may not be possible to address such weaknesses directly (e.g. the rights and privileges of the user cannot be reduced, because they are required in order to perform legitimately assigned tasks), and therefore some form of monitoring is necessary to identify potential misuse. The typical means of addressing this is via manual supervision and audit procedures. However, reliance upon such methods clearly leaves significant scope for damage to be done before an incident is finally identified [1]. An automated approach would therefore be preferable, and analogous approaches are already used to identify external attacks through the use of Intrusion Detection Systems (IDS). Unfortunately, traditional IDSs are ineffective in detecting users who abuse their legitimate privileges at the application level, because they do not have the knowledge of application level semantics, required separation of duties, and normal working scope of the users. However, if the detection system can be provided with the information of user working-scope, as well as the manner and the context in which the operations can be carried out within relevant environments, then insider misuse may be monitored more effectively.

This paper outlines a novel framework for addressing the problem of insider misuse monitoring. The approach argues that users with similar roles and responsibilities should exhibit similar behaviour within the system, enabling any activity that deviates from the normal profile to be flagged for further examination. Considering a given application context, the legitimate occurrence of particular operations and transaction types may be characterised and assessed according to a combination of misuse signatures and historical profiles. The signatures will encapsulate known or predicted patterns of misuse that insiders may attempt to perpetrate, whereas profiles will be constructed from legitimate user activities occurring over a period of time. Attempts by insiders to misuse the system should then appear anomalous when compared against the historical norms. While the concepts of signature and anomaly- based detection are already accepted approaches within traditional IDS systems [2], the approach proposed in this paper will tailor their use to the application-level within a monitored system, in conjunction with knowledge about user roles and the required working scopes associated with them. In this way, signatures and profiles may be specified at a number of levels (e.g. for applications, for user roles, and for individual users within those roles), enabling cross-correlation between them in order to identify signs of potential insider misuse.

The next section compares the detection strategies employed by traditional IDSs, and considers their applicability within a misfeasor detection context. This is followed in section 3 by a discussion of Role Based Access Controls and the importance of role-relationship management for insider misuse detection. The proposed framework for insider misuse detection is presented in section 4, leading into discussion of practical implementation in section 5. Brief concluding comments are presented in section 6.

2 INSIDER MISUSE DETECTION STRATEGIES

IDS employ two main strategies to identify attacks, namely misuse-based and anomaly-based detection [3], and it is possible to see how each of these could be applied to the insider problem.

2.1 Misuse-based detection

This approach relies upon knowing or predicting the intrusion that the system is to detect. Intrusions are specified as attack signatures, which can then be matched to current activity using a rule-based approach. A similar approach could potentially be incorporated for misfeasor incidents, based upon those methods that employees have been known to exploit in the past, or those that can be anticipated they would attempt based upon the privileges and resources available to them. For example, at a conceptual level, one such misuse signature might relate to a user who is identified as attempting to modify a record about him/her in a database (e.g. the payroll example indicated earlier). The rule here is that no one should modify their own records without someone else's authorisation. The problem with applying misuse-based detection to insider misuse is that the possible misuse scenarios for insiders are wide ranging and could be extremely organisation-specific. Thus it would be difficult to catalogue them all. Misuse-based detection is only as good as the database of signatures it relies on for detection. Therefore, the database would need to be updated constantly to detect new attack methods. This approach would not be suitable for insider misuse detection as it would be too time-consuming in person-hours to create misuse signatures for all possible scenarios and to continually keep them updated.

2.2 Anomaly-based detection

This approach relies upon watching out for things that do not look normal when compared to typical user activities within the system. In standard IDS, the principle is that any event that appears abnormal might be indicative of a security breach having occurred or being in progress. The assessment of abnormality is based upon a comparison of current activity against a historical profile of behaviour that has been established over time. One advantage insider misuse detection system has over outsider attacks is that it is possible to characterise normal activities of insiders according to their job position, as users with the same responsibilities should exhibit similar activities within the system and application environment to complete their daily tasks. The similarities may be profiled to represent normal behaviour for users with the same responsibilities, and different profiles for different job positions. If the user's behaviour deviates from the normal profile that represents his position, the activity should be flagged as suspicious.

The concept of applying the techniques for the detection of misfeasor activity makes the task more difficult, because we are dealing with legitimate users who are not violating access controls. From a misuse-based detection perspective, it is more difficult to identify the ways in which an insider might misuse the resources to which they have legitimate access, while from an anomaly detection perspective the level of behaviour profiling would need to be more precise and comprehensive. When basing the assessment upon a comparison against their behaviour profile, a legitimate user misbehaving will almost certainly be more difficult to identify than a total impostor who is masquerading under the legitimate user's identity, because it is more likely that the impostor's behaviour would deviate by a larger margin, whereas conversely the deviation is likely to be minimal for a legitimate user who abuses existing privileges. In addition, in an adaptive system, the process of profile refinement might be exploited by wily misfeasors that gradually train the system to accept misuse behaviour as normal. As such, this aspect is still an area of active research [4].

3 KNOWLEDGE OF WORKING SCOPES

Another problem associated with insider misuse detection is that current IDSs lack the necessary knowledge of business processes, organisation hierarchy, separation of duties, and the role of the users within the organisation structure. This knowledge needs to be expressed in a form that is understandable to the IDS, if effective misfeasor monitoring is to take place. Role management principles specified by Gavrila [5] are utilised in Role-Based Access Control (RBAC) to support user role assignment, role relationships, constraints and assignable privileges [6]. A role can be thought of as a collection of operations required to complete the daily tasks of a user. In RBAC operations are associated with roles and the users are assigned to appropriate roles. This approach simplifies the task of assigning permissions to the user, as the roles for appropriate job functions are created with the least privileges required to complete the relevant tasks and the users are assigned to the role that reflects their responsibilities. Users can be assigned from one role to another, or assigned multiple roles, and permissions can be assigned at role-level to affect all users associated with the role. The type of operations and objects that can be controlled by RBAC is dependant upon the environment and the level at which it has been implemented. For example, at the OS level, RBAC may be able to control read, write, and execute; within database management systems controlled operations may include insert, delete, append, and update; within transaction management systems, operations would take the form that exhibit all properties of a transaction. The term transaction here means a combination of operation and the data item affected by the operation. Therefore, a transaction can be thought of as an operation performed on a set of associated data items. The ability to control specific transactions, rather than restricting simple read and write operations are very important in database environments. For example, in a bank a teller may perform a savings deposit transaction, which requires read and write access to specific fields in a savings file and a transaction log file. An account supervisor may need to correct transactions, which require the same read and write access to the same files as the teller. However, the operation performed and the values entered may be different. The teller may not be allowed to edit the transactions after they have been completed. The account supervisor may not be allowed to initiate the deposit or withdrawals, but only correct the ones that have been completed. The problem is that determining whether the data has been modified in the authorised manner, for it can be as complex as the actual transaction that modified the data. Therefore, transactions need to be certified and classified before associating them with the roles. To characterise the required transactions for a role, duties and responsibilities of the users need to be specified first.

In RBAC separation of duties can be applied by specifying mutually exclusive roles. In the RBAC framework, administrators can regulate who can perform what actions, when, from where, in what order and sometimes under what circumstances. Access controls only allow or deny access to certain resources, however there is a need to monitor and analyse the user actions after the access has been gained and the operations had been carried out. In principle the idea of roles and role-management principles can be applied to misfeasor monitoring. Instead of allowing or denying operations to be performed, common user operations can be associated with roles, and the users can be assigned to appropriate roles. If the user's operations deviate from the common profile, a thorough investigation can be carried out to clarify if the user has misused the system in an inappropriate manner or for unapproved purpose.

4 ARCHITECTURAL FRAMEWORK OF A MISFEASOR MONITORING SYSTEM

It has been mentioned previously that anomaly detection is more suitable for insider misuse detection, because employees' normal behaviour can be profiled. It is assumed that the users with the same responsibilities within the organisation will exhibit similar activities within the system, and their working-scopes may be established. The idea of establishing working-scopes for users with same responsibilities has been tested in relational database environments by Chung et al [7]. However, in order to be able to detect violation of separation of duties, the detection system needs to be provided with the knowledge of organisation hierarchy and relationships between roles. RBAC utilises role-relationship management principles to define role-hierarchy and separation of duties. The authors' proposed system combines the ability of RBAC to provide knowledge of role-relationships with intrusion detection techniques to effectively detect users who abuse their existing privileges. Figure 1 presents the framework of the conceptual insider misuse detection system. Functional modules are explained in subsequent paragraphs.

4.1 Management Functions

All management functions, such as defining roles, characterisation of operations, association of operations to roles and user assignment to roles, are carried out from the Management Console. The working scope of a user is defined by the operations associated with the role(s) the user assumes. Once the separation of duties between roles has been defined, it is expressed in the Role-Relations Matrix, such as inheritance, static separation of duties, and dynamic separation of duties. Static separation of duties occurs at the role level by specifying mutually exclusive roles. When the two roles are in static separation of duties, a user may not be assigned both roles. Dynamic separation of duties occurs at the operations level. The relationships between operations are expressed in the operations matrix, and the conditions can be that operations within dynamically separated roles are:

- Mutually excluded
- Disallowed to execute concurrently
- Disallowed to perform both operations on the same set of data

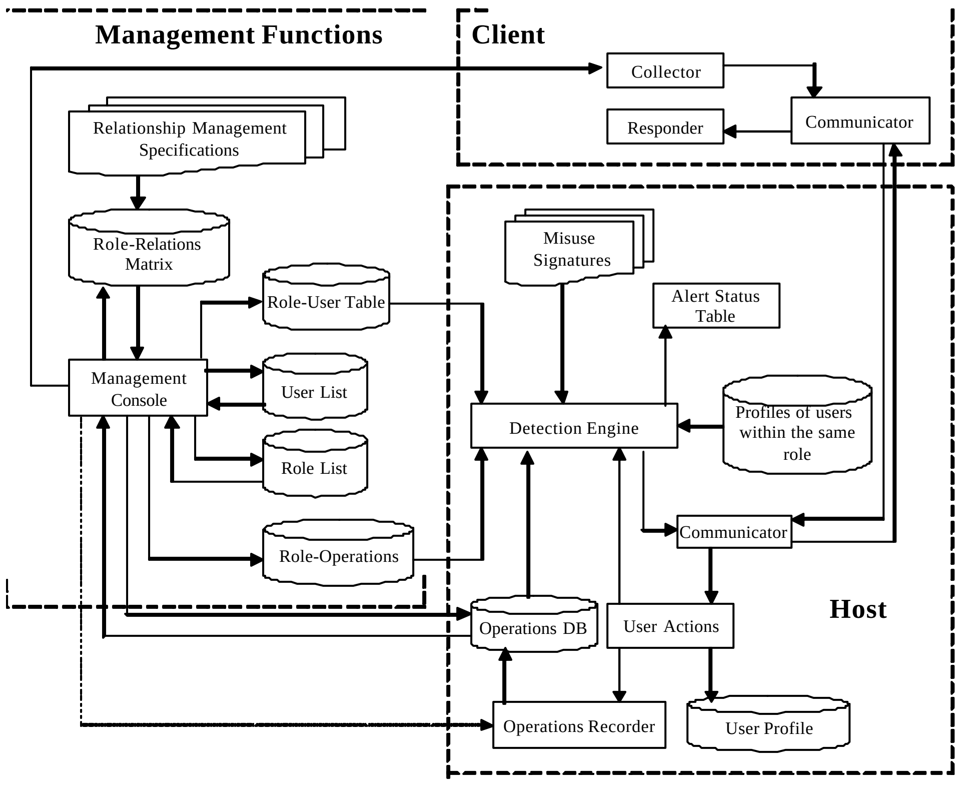


Figure 1. Conceptual Framework of Misfeasor Monitoring System

When the two roles are in dynamic separation of duties, the user may not execute the operations that are mutually exclusive or on the same set of data. The relationships expressed in the Role-Relations Matrix are checked against the rules specified by [5] for consistency.

4.2 Host

This is where the actual profiling of user(s) and the detection process takes place. Characteristics of each operation are stored in the Operations DB along with an appropriate name for each operation. The characteristics are dependent upon which level of the system they are being profiled at. Characteristics of the operations may be in the form of file access, sequence of system calls, SQL queries, API calls, User interactions, and Network access. Recording the characteristics of each operation is controlled from the Management Console. The profiling should be done at all three levels of the system namely: network, system, and application level. At the network level, roles should be profiled based on the essential access to subnets in order for the users of the role to complete their daily tasks. At the system level, roles should be profiled on the use of applications required to complete the tasks. It should also be established which machines the users of the role can/cannot perform the task from. Again, at the system level, roles should be profiled based on what files need to be accessed in order to complete the task, along with the access mode and the application/process from which the files are accessed. Once the user has gained access to the file, and if the file is accessed from an application in which the file can be modified or manipulated (e.g. Databases), the application level monitoring should commence. At the database level, user queries and the associated values should be monitored. The problem is that determining whether the data has been modified in the authorised manner, for it can be as complex as the actual transaction that modified the data. Therefore, transactions need to be certified and classified before associating them with the roles. The Detection Engine then checks the roles available to the active user, and next checks the RoleOperations table for the names of the operations available to the user. After which the characteristics of the ava ilable operations from the Operations DB are compared to the current user actions. If current user actions do not match the characteristics of operations available to the user, the administrator is alerted. This alert may indicate the user performing a totally new operation, or performing a valid operation in the Operation DB but is violating separation of duties because the operation is not listed under any roles the user may assume.

The envisaged detection flow is as follows:

1. Detection Engine gets the name of the user from the Client. Looks for the roles the user's name is associated with, in the Role-user table.
2. After acquiring the list of roles for the user, the Detection Engine looks for the names of the operations associated with each role in the Operations DB. (Note: only names of the operations are associated with the Roles.)
3. After acquiring the names of operations available to the user, the Detection Engine reads the characteristics of available operations from the Operations DB and they are compared against current user actions.
4. If the current user actions match with the characteristics of operations available to the user, the user is not in breach of static separation of duties.
5. If OpA belongs to RoleA, OpB belongs to RoleB, and RoleA and RoleB are in dynamic separation of duties. Condition of the separation is checked to clarify whether the operations are:
   - Mutually excluded
   - Disallowed to execute concurrently
   - Disallowed to perform both operations on the same set of data

If the user violated the specified condition, the system security officer is alerted. In addition, the misuse rules employed in expert systems within traditional IDSs can also be included. These rules may then be associated with an operation, such as modifying the payroll database to increase one's own wages. In this case, the process is as follows: If modification is performed on the payroll database, check that the employee ID of the user is not the same as that of the record being modified.

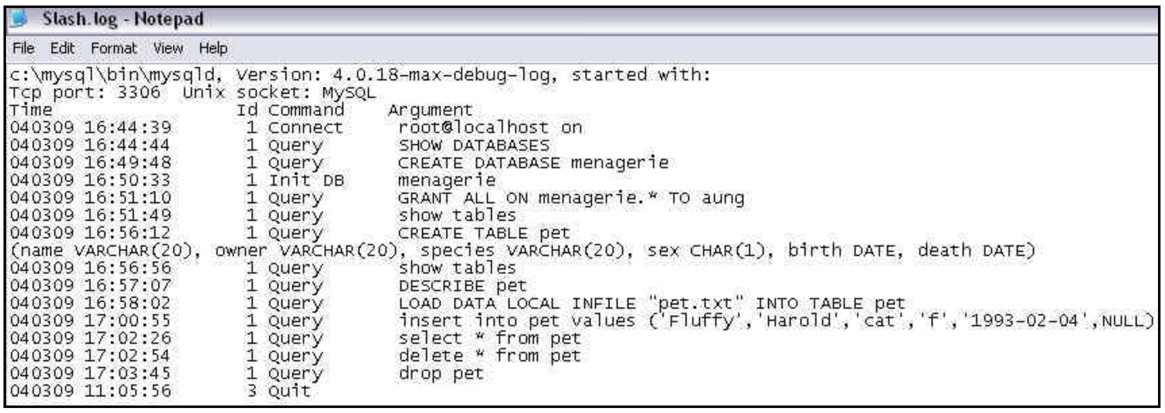
4.3 Client

This is where the actual data is collected and transferred to the Host for analysis. The Clients can be network server systems or end-user workstations. The nature of the data collected may vary depending on the type of the Client. For example mail logs can be collected from the mail server, user queries from the database server, and application logs from user workstations. The data to be collected is specified by the system administrator from the Management Console. The collected data can then be refined to a standard format by the Communicator module before sending the data to the Host, so that data from heterogeneous Client systems is in a standard format. The Client may also have a Responder module to respond to detected incidents, and the appropriate response for each incident can be specified from the Management Console. For example, when a misuse is detected, the Responder may be configured to terminate the user session, revoke privileges, deny further access, alert the security officer, or terminate the anomalous process [8].

5 PRACTICAL IMPLEMENTATION & CONCEPT VALIDATION

5.1 Choice of Environment

The practical implementation begins with the collection of data for insider misuse analysis. On the Client side of the monitoring system, the Clients can be end-user workstations, web, mail, and database servers. The data collected depends upon the type of client. For the practical implementation and validation of the framework, log data from a database environment is used. There are several reasons for choosing logs from such an environment to validate the concept. First of all, databases are multi-user client-server environments with multi-level security requirements, where data compartmentalisation and separation of duties is required to preserve the confidentiality and integrity of the data. In addition the privilege system in database environments is more complex than the privilege systems in many other Operating System (OS) and application environments. It is crucial for the detection system to be able to understand who has legitimate access in such a complex privilege-based system, because insider misuse involves monitoring successful access by legitimate users. For the choice of database management system (DBMS), MySQL is used. The reason for choosing MySQL is that its general query log contains the information necessary to effectively monitor insider misuse (i.e. the date, time, and audit ID for the user session, the type of query and the actual queries the user has issued). Looking at the general query log (Figure 2), the information regarding the date and time can provide the detection engine with the knowledge of when the event occurred. The audit ID can help identify the user responsible for the activities. The most useful parts are the Command and Argument columns; these provide the commands and arguments, as the users issued them, without any modification. This provides the knowledge of how the user interacted with the application environment. As stated in the previous sections, the advantage of insider misuse monitoring is that once the user is in an application environment, the manner in which they can legitimately interact with the given environment can be characterised, from which the normal user behaviour can be profiled. This normal profile of legitimate interaction can help the monitoring system detect the users who interact with the environment in an unacceptable manner, by comparing current user activities against the normal user interaction profile of a given role.



```
c:\mysql\bin\mysqld, Version: 4.0.18-max-debug-log, started with:
Tcp port: 3306  Unix socket: MySQL
Time                 Id Command    Argument
040309 16:44:39       1 Connect     root@localhost on
040309 16:44:44       1 Query       SHOW DATABASES
040309 16:49:48       1 Query       CREATE DATABASE menagerie
040309 16:50:33       1 Init DB     menagerie
040309 16:51:10       1 Query       GRANT ALL ON menagerie.* TO aung
040309 16:51:49       1 Query       show tables
040309 16:56:12       1 Query       CREATE TABLE pet
(name VARCHAR(20), owner VARCHAR(20), species VARCHAR(20), sex CHAR(1), birth DATE, death DATE)
040309 16:56:56       1 Query       show tables
040309 16:57:07       1 Query       DESCRIBE pet
040309 16:58:02       1 Query       LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet
040309 17:00:55       1 Query       insert into pet values ('Fluffy','Harold','cat','f','1993-02-04',NULL)
040309 17:02:26       1 Query       select * from pet
040309 17:02:54       1 Query       delete * from pet
040309 17:03:45       1 Query       drop pet
040309 11:05:56       3 Quit
```

Figure 2. MySQL's general query log

5.2 Generating Signatures and Profiles

For the purpose of identifying authorised operations within the database environments SQL query signatures are used. Generating SQL query signatures can be done using regular expressions as described by Low et al [9], where the variables that the user may change within the SQL queries are replaced with regular expressions. These signatures are then used to detect patterns of authorised transactions from the queries saved in the user's activity profile. The fingerprints generated represent the normal profile of a legitimate operation and are assigned a unique operation ID. The filename represents the operation ID and is saved in the Operations DB folder.

5.3 Operations Management

After the signature generation for all authorised operations, the relationships between operations need to be identified. The operations matrix represents the relationship between operations, and this knowledge helps in assigning operations to roles and detecting violation of separation of duties. The possible expressions in this matrix can be:

- The two operations cannot be assigned to the same role (mutually exclusive)
- The two operations cannot be performed on the same set of data
- The two operations cannot be executed concurrently

The first condition is checked when the operations are assigned to roles, and all the conditions are checked again during detection for violating separation of duties.

5.4 Defining Roles and Assigning Operations

After generating authorised operation signatures and defining relationships between operations, various roles that would exist in the organisation need to be defined. The type and number of roles that exist will differ from one organisation to another. Each role would be given a name and a unique ID. Next the tasks needed to be performed by the user need to be identified, and the relationships between roles need to be established. The knowledge of role-relationship can help in assigning users to roles and detecting static separation of duties. The relationships between roles are expressed in the role-relations matrix. Possible relationships between roles are:

- Static separation of duties (mutually exclusive)
- Dynamic separation of duties
- Inheritance

Gavrila and Barkley [5] formulated the rules for checking the consistency of role-role and role-user relationships. Their paper described seventeen properties that need to be satisfied in order to maintain consistency of relationships between roles, such as inheritance, and separation of duties. Finally, the required operations are assigned to each role in the role-operations table (an example of which is shown in Table 1).

Table 1. Role-Operations Table

Role Number	Operations ID	Operation ID	Operation ID
1	12345	49586	92190
2	38298	44890	37929
3	89203	38003	39298

5.5 User Role Assignment

After the roles have been defined and the appropriate operations have been assigned, the users need to be assigned to role(s). First the user is assigned a primary role, after which the role-relations matrix is checked and any role(s) that are in static separation of duties with the user's primary role is removed from the list of assignable roles to the user. Now, the user's secondary roles can only be selected from the roles that are not in static separation of duties with their primary role. If a secondary role is assigned to the user, the list of assignable roles will be expressed as: roles that are not in static separation of duties with user's (primary role and secondary role). The assignable roles are the roles that are not in static separation of duties with the user's existing roles. Thus if the user has too many roles with separation of duties constraints, the list of assignable roles will decrease as a result. In the User-Role table, the IDs of roles assigned to the user would be associated with the user's name and user ID. The role assignment may be performed by the Human Resource personnel with the acknowledgement of the manager of the role the user has been assigned to.

5.6 Filtering Logged Data & Generating Individual User Profiles

MySQL's general query log contains the queries from every user's session. Therefore, the queries for each user session need to be filtered and extracted. Then the extracted log data is saved in username@host.log files. It is fairly trivial to extract the log data of each user session as a unique audit ID is assigned every time a user connects to the MySQL server. When the user initiates a connection, MySQL associates a unique user ID for the user@host, and this can be seen in the general query log. By having separate profiles for each unique user, a user's profile can be compared against the others and also against the normal profile(s) of the role(s) the user may assume.

5.7 User Profile Management

Information related to user activity needs to be recorded and properly managed to allow efficient analysis of the data and effective detection of misuse. For each unique user, a folder with the user ID is created within the User Profiles directory. The folder contains three subfolders: Net, OS, and Apps. The Net folder contains the user profiles with information related to network access. The OS folder contains the user profile with information related to file access, resource usage, and command usage (any behavioural characteristics that the user may exhibit at the OS level are stored here.) The Apps folder may contain subfolders with the name of the applications where the user profiles has been extracted from. Each application folder contains audit logs of the user within its application environment. For the purpose of our experiments, the filtered log data from the MySQL general query log is saved in the MySQL sub-directory within the Apps directory, which contains user@host log files. Each user@host file will contain all the activities the user performed within the database environment from each host machine.

5.8 The Pattern Matching Process

The detection engine obtains the identity of the user, checks the roles the user may assume from the user-role table, obtains the list of operations associated with the roles the user may assume from the role-operations table, and finally reads the operation signatures. The system then searches for the operation that has the longest sequence of SQL queries in the user@host logs. The sequence of queries that matched with the legitimate operation is then replaced with a blank line or a matched identifier. After matching current user activity with legitimate operations/transactions, the arguments used with the operation are saved in the "user@host.aut" file along with the identifier for the type of operation. The system then searches for the operation with the second longest sequence of queries and so on. After the operation with the shortest sequence of queries has been searched, the system then checks if there are any unmatched queries left. Any queries left unmatched after the last search are considered unauthorised operations for any role(s) the user may assume. All the operation signatures available are read and then compared against the queries left in the user@host file. The sequence of queries that matched with the legitimate operation are then replaced with a blank line or a matched identifier. Each time a match is found, the ID of any operation that matches is saved in the "user@host.sop" file. Any queries still left in the user@host file are considered as anomalous operations. With the help of the operations matrix, the operation IDs saved in the user@host.sop file are checked for dynamic separation of duties with the authorised operations for the user. If they are found to be in dynamic separation of duties with the authorised operations for the user, then the conditions of separation are checked such as, mutually exclusive, performed concurrently, or performed on the same set of data. If an operation “OpA” from user@host.aut and an operation “OpB” from user@host.sop are found to be mutually exclusive, the role-operations table is checked to ensure that the mutually exclusive operations are not assigned to the same role (accidentally or intentionally). Then the role-operation table is checked for any role(s) that have OpA or OpB as part of their authorised operations. The role IDs of the roles that employ OpA or OpB are obtained, and then the role-relation matrix is checked to determine whether proper separation of duties has been defined between roles that employ OpA and roles that employ OpB. The returned results from each stage of checks are then sent to the database administrator.

6 CONCLUSIONS

It is clear that insider abuse can have a major impact upon an organisation since the perpetrators have a good idea of what is sensitive and valuable within the company. Knowing where these resources are stored, and what security mechanisms are used to protect them, also helps insiders in circumventing controls and evading detection. As such, it is essential for organisations to be cognisant of the threat, and for mechanisms to be available to facilitate detection of these incidents. The proposed framework provides a basis for future work in this area, and the authors’ research will proceed to address further aspects of practical implementation and validation within the MySQL environment. The findings from this work will be the focus of future publications.

REFERENCES

1. Dhillon, G. and Moores, S. ‘Computer crimes: theorizing about the enemy within’, Computers & Security, Vol.20, No. 8, pp715-723 (2001)

2. Axelsson, S. ‘Intrusion Detection Systems: A survey and taxonomy.’ Technical Report (99-15). Department of Computer Engineering, Chalmers University, March (2000)

3. Amoroso, E. ‘Intrusion Detection: An Introduction to Internet, Surveillance, Correlation, Traceback, Traps and Response’, First Edition, Intrusion.Net books, NJ, ISBN: 0966670078 (1999)

4. Anderson, D. Frivold, T. and Valdes A. ‘Next-generation intrusion detection expert system (NIDES): A summary’, Technical Report. Computer Science Laboratory, SRI International. May (1995)

5. Gavrila, S.I. and Barkley, J.F. ‘Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management’, Third ACM workshop on Role Based Access Control, Fairfax, Virginia, October 22-23, pp81-90 (1998)

6. Sandhu, R.S. Coyne, E.J Feinstein, H.L and Youman, C.E. ‘Role-based access control models’. IEEE Computer, Vol.29, No. 2, February, pp38-47 (1996)

7. Chung, C.Y. Gertz, M. and Levitt, K. ‘DEMIDS: A Misuse Detection System for Database Systems’, Proceedings of the 3rd International Working Conference on Integrity and Internal control in Information Systems, pp159-178 (1999)

8. Papadaki, M. Furnell, S.M. Lines, B.M and Reynolds, P.L. ‘A Flexible Architecture for Automated Intrusion Response’, Proceedings of Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Turin, Italy, October 2-3, pp65-75 (2003)

9. Low, W. L. Lee, J. and Teoh, P. ‘DIDAFIT: Detecting Intrusions in Databases Through Fingerprinting Transactions’, Proceedings of the 4th International Conference on Enterprise Information Systems, Ciudad Real, Spain, April 2-6, pp121-128 (2002)