Mining Databases for Transaction Classification using Intrusion Detection System

A THESIS
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

DOCTOR OF PHILOSOPHY
IN
Computer Science & Engineering

Submitted by
Indu Singh (2K16/Ph.D/CO/08)

Under the supervision of
Prof. Rajni Jindal
Professor
Department of Computer Science & Engineering
Delhi Technological University

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

JANUARY, 2023

DECLARATION

This is to certify that the work embodied in the thesis entitled **"Mining Databases for Transaction Classification using Intrusion Detection System"** submitted by **Ms. Indu Singh**, Roll No – 2K16/Ph.D/CO/08, as a Scholar in Department of Computer Science & Engineering, Delhi Technological University, is an authentic work carried out by her under my guidance. This work is based on original research and has not been submitted in full or in part for any other diploma or degree of any university to the best of my knowledge and belief.

Place: New Delhi **Ms. Indu Singh**

2K16/Ph.D/CO/08
Department of Computer Science & Engineering
Delhi Technological University, Delhi-110042

CERTIFICATE

I hereby certify that the Ph.D Dissertation titled **"Mining Databases for Transaction Classification using Intrusion Detection System"** which is submitted by **Ms. Indu Singh**, Roll No – 2K16/Ph.D/CO/08, Department of Computer Science & Engineering , Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the research work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Prof. Rajni Jindal**

Professor
Department of Computer Science & Engineering
Delhi Technological University, Delhi-110042

ACKNOWLEDGEMENT

Completion of this Ph.D thesis was achievable with the support of several people. I would like to express my sincere gratitude to all of them. First of all, I am extremely grateful to my Supervisor, **Prof. Rajni Jindal**, Professor, Department of Computer Science & Engineering, Delhi Technological University (DTU), Delhi for her priceless guidance, scholarly inputs, constant and unconditional support I received throughout the research work. This feat was possible only because of the unconditional support provided by her. A person with an amicable and positive disposition, ma'am has always made her accessible to clarify my doubts despite her hectic schedules. I consider it as a big opportunity to do my Ph.D under her guidance and to learn from her research erudition and knowledge. I thank her again for her assistance and support.

I owe my most sincere gratitude to **Prof. Vinod Kumar**, Professor, Head of Department, Department of Computer Science & Engineering, Delhi Technological University (DTU), Delhi, for his constant support throughout the duration of this work. Besides the moral support, he has provided all infrastructural facilities required for successful completion of this work. I would also like to thank all the faculty members and staff members of the Computer Science & Engineering Department, Delhi Technological University, for their encouragement and support.

I am indebted to my family members, especially my parents and my sisters **Ms. Tapasya Singh** and **Ms. Shivangi Singh** for their constant support and encouragement. My parents' encouragement, tenacity, and unwavering dedication were the driving forces throughout my Ph.D journey. They have been epitomes of inspiration and sacrifice throughout my life, before and during Ph.D.

Place: New Delhi

Ms. Indu Singh

2K16/Ph.D/CO/08

Department of Computer Science & Engineering

Delhi Technological University

Delhi-110042

Abstract

Organisations are increasingly employing databases on a large scale to store critical data that is essential for their functioning. Malicious access and modifications of the databases may lead to adverse financial and legal implications. In recent years, security researchers have focused on detecting abuse of access privileges by the employees of an organisation. Identifying threats from insiders is hard because they are aware of the organisation of the database in addition to having authorised access privileges. Therefore, an intrusion detection system(IDS) collects data from a computer system, analyses it for security faults, and reports the results to the administrator.

Most of the current systems are not sufficient in properly classifying the users based on their access privileges and often end up flagging the legitimate users as intruders. These systems based on dependency mining rely on hard-coded values of sensitivity parameters which makes the system prone to a decrease in accuracy over time due to changes in user behaviour. On the other hand, the techniques that rely purely on unsupervised learning methods and data mining generally suffer from a higher false positive rate and supervised learning methods have proven to be ineffective against novel attacks. Majority of the techniques suffer from low recall because of their sensitivity towards changing user access patterns. Thus, there exists a need for a hybrid approach that caters to the pitfalls of both the topologies while retaining their benefits. To overcome the above problems, we propose novel approaches to integrate the benefits of various mining and metaheuristic techniques for reducing the high false positive rate.

The thesis first addresses a new data mining-based approach namely 'The Fuzzy Association Data Dependency Rule Miner (FADDRM) for identifying malicious transactions in databases by mining data dependencies between data items. Our method focuses on extracting association rules using fuzzy association rule mining in combination with Fuzzy Connected Clustering (FCC) and the rules are used to classify the transactions as

malicious or non-malicious.

In the second part, we discuss a novel database intrusion detection system (DIDS) based on Expectation Maximisation Clustering and Sequential Pattern Mining (EMSPM). This method relies on the database pre-existing logs and the data dependency rules that are obtained by mining user information access patterns using modified PrefixSpan algorithm. The Expectation Maximisation clustering algorithm assigns role profiles based on the database user's behaviours. These clusters and patterns are then processed into an algorithm that prevents generation of unwanted rules as well as malicious transactions.

In the third part, we present a BIDE and modified Particle Swarm Optimization clustering based malicious query detection (BPSOMQD) approach.This method incorporates frequent closed sequential pattern mining which forms the basis for generation of data dependency rules. Further to recognise anomalous user activity, modified Particle Swarm Optimization algorithm is proposed which is used to generate role profiles associated with the transaction. To classify a transaction as malicious or non-malicious, a combination of Multilevel Rule Similarity Score (MRSS) between data dependency rules, incoming transactions and Cluster Similarity Index (CSI) with created role profiles is used.

In the fourth part, we propose a Frequent Sequential Pattern mining and a modified metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm (FPG-WWO) to identify malicious transactions in RBAC and non-RBAC supervised databases. We use the CM-SPADE mining algorithm to extract database dependency rules that are used to detect outsider threats. Insider threats are detected by comparing user activities to previously determined role profiles that are assigned using the modified metaheuristic clustering from the past user behaviour. During the learning phase, transactions are labelled malicious on the basis of novel similarity threshold, i.e., the "congruity index". If the transaction is found to be malicious, an alarm is triggered and the database executes a rollback.During the detection phase, a role matcher authenticates role clusters for the incoming transaction. If the role profile is not matched, the transaction is labelled as malicious and aborted.

In the fifth part, we propose Trust factor based analysis of user behaviour using se-

quential pattern mining for detecting intrusive transactions in databases (TFUBID) to prevent misuse of access privileges by insiders of an organisation. Since, groups of users access the organisational database for similar purposes, we cluster user behaviour vectors using fuzzy clustering and define a class of Integral Data Attributes using sequential pattern mining to model trust factor based behavioural patterns of employees accessing the database assigning higher weight to critical elements and Directly Correlated Attributes. Modified Jensen-Shannon distance is used to give weights to data attributes and avoid the curse of dimensionality in dissimilarity calculation.The idea of an "incredulity score" is introduced which quantifies the degree of anomalous behaviour exhibited by each user based on his previous transactions.

Finally, we propose Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining (OIUWSPM) ,a novel method for detecting malicious transactions that follows a sequential flow that begins with outlier detection and continues with various behavioural checks at the role induced rule mining component and user level feature extraction. The idea of Dynamic Sensitivity is used distinctively at role level which complements the access counts of each attribute. We introduce the notion of Coherence Count computed by the application of Longest Common Subsequence(LCS) and the utilisation of Levenshtein distance for calculating Divergence between the user level Relation access paths. We analyse user behaviour by registering user level Relation access path reinforced on user transactions.

Comprehensive implementation of the above models reveal that their performance is better and more efficient than other models.

Contents

List of Tables

List of Figures

List of Abbreviations

ACC	Accuracy
BM	Bookmaker Informedness
BPSOMQD	BIDE and modified Particle Swarm Optimization clustering based malicious query detection
CADS	Community Anomaly Detection System
CCD	Coherence Count Determinator
CDI	Cluster Deviation Index
CERT	Computer Emergency Response Team
CIA	Confidentiality, Integrity and Availability
CMAP	Co-occurrence MAP
CR	Critical Rules
CSI	Cluster Similarity Index
DBA	Database Administrators
DBMS	Database Management Systems
DCA	Directly Correlated Attributes
DIDS	Database intrusion detection systems
EBAM	Experience Based Access Management
EM	Expectation Maximization
EMSPM	Expectation Maximisation Clustering and Sequential Pattern Mining
F1	F1 Score
FADDRM	Fuzzy Association Data Dependency Rule Miner
FCC	Fuzzy Connected Clustering
FDR	False Discovery Rate
FNR	False Negative Rate
FOR	False Omission Rate
FPGWWO	Frequent Sequential Pattern Mining and a modified Metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm
FPR	False Positive Rate
GWO	Grey Wolf Optimizer
HIDS	Host-based Intrusion Detection Systems
ICS	Incredulity Score
IDA	Integral Data Attributes
IDS	Intrusion detection systems
IPPS	Iteratively Pruned Prefix Span
L0S	Level-0-Similarity
L1S	Level-1-Similarity
LCS	Longest Common Subsequence

MCC	Matthew's Correlation Coefficient
MFRAP	Most Frequent Relation access path
mhGW-WOA	Modified Hybrid of GreyWolf Optimizer with Whale Optimisation Algorithm
MK	Markedness
MRSS	Multilevel Rule Similarity Score
NIDS	Network Intrusion Detection Systems
NPV	Negative Predictive Value
OIUWSPM	Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted
OLTP	Online Transaction Processing
PCA	Principal Components Analysis
PPV	Positive Predictive Value
QLAI	Query Log Affinity Index
QTG	Query Template Generator
RAP	Relation Access Path
RBAC	Role Based Access Control Model
ROC	Rule Overlay Count
ROE	Rule Overlay Extent
RR	Read Rules
SGREA	Sensitivity Galvanized Rule Extraction Algorithm
SPM	Sequential Pattern Mining
TFUBID	Trust factor based analysis of user Behaviour using sequential pattern mining for detecting intrusive transactions in databases
TID	Transaction ID
TNR	True Negative Rate
TPC-C	Transaction Processing Performance Council Benchmark C
TPR	True Positive Rate
UBRAPG	User Behaviour RAP Generator
UID	User ID
WOA	Whale Optimization Algorithm
WR	Write Rules
WRF	Random forest with weighted voting

Chapter 1

INTRODUCTION

Databases store vital information of an organization and hence are integral for its efficient functioning. This necessitates the establishment of Database Intrusion Detection Systems (DIDSs) which can detect and prevent unauthorized user access to the critical information stored in the database.

This chapter presents the work layout of the thesis . Section 1.1 introduces the DIDS. The security attacks on DIDS will be discussed in section 1.2. The present DIDS security requirements are discussed in section 1.3. Section 1.4 presents the motivation of the work. In section 1.5, the problem statement is discussed. Section 1.6 elaborates the research objective of the work. Section 1.7 presents an overview of the proposed solutions. The research outcomes are discussed in Section 1.8. The organization of this thesis is presented in Section 1.9. Lastly, in Section 1.10 summary of the chapter is discussed.

1.1 Database Intrusion Detection System (DIDS)

Over the years, as databases evolved, they became the preferred form of storing vast amounts of data. Databases play an indispensable role in the smooth working of an organisation. Databases are responsible for storing crucial enterprise data, and are therefore subjected to a number of threats. Hence, it is integral that databases are defended against attempts of intrusive access. Databases are exposed to two different types of attacks – external and internal.

Intrusion Detection Systems (IDS) can also be classified as network-based and host-based IDSs. Network-based IDSs detects attacks by capturing and analysing network packets. Network-based IDSs consist of uni-functional hosts that are designed to receive network

traffic in specific parts of the network to capture and redirect attacks towards a single management console. Recently, Medhat et al. [1] designed a two-layered and three-layered IDS for wireless sensor networks (WSN) using supervised and unsupervised learning at different levels. Wankhade and Jondhale [2] used the ensemble clustering technique for intrusion detection which increases detection rate and lowers false alarm rate. As described by Kumar et al. [3], Host-based IDS are capable enough to work with high quality informative data. These systems examine clients' activities and behaviour on a given machine. Host-based IDSs are usually aided by highly informative and quality data as described by Nazer and Selvakumar [4].

The architecture of DIDS is shown in Figure 1.1. This section highlights some of the major approaches that have been presented in the field of DIDS in recent times. Before that Santos et al. [5] have given insights for the field of DIDS. The intent is to do the following: highlight the comparisons as well as flaws of the different DIDS approaches, present a discussion on their weaknesses and strengths based on prevention or termination of user action when abnormal access is observed, look for accuracy in detection and the practical limitations of such systems and also discuss intrusion prevention systems, DIDS with proactive response mechanisms. In DIDS, response plays a key role in mitigation of these attacks. Intrusion detection systems without a prompt response are not effective even if they can detect threats and generate alarm.

Database management systems have increasingly made access and concurrent modification of data easy and efficient. These systems have also led to an increase in illicit access to the database by exploiting various vulnerabilities. Although firewalls act as a security mechanism to monitor incoming and outgoing traffic, they have been found to be inadequate to protect the database from a wide range of attacks and therefore need to be complemented with IDS. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of the data as suggested by Sandhu et al. [6].

Considering the importance of data in an economic context, any limitation of these IDSs is a critical issue which can render them unsuitable or even incompetent for many database systems. Hence the development of an intrusion detection system which defends the system against illegal access is crucial. There are two major approaches to conquer the problem of detecting intrusions in a database. One is to check the queries against well-

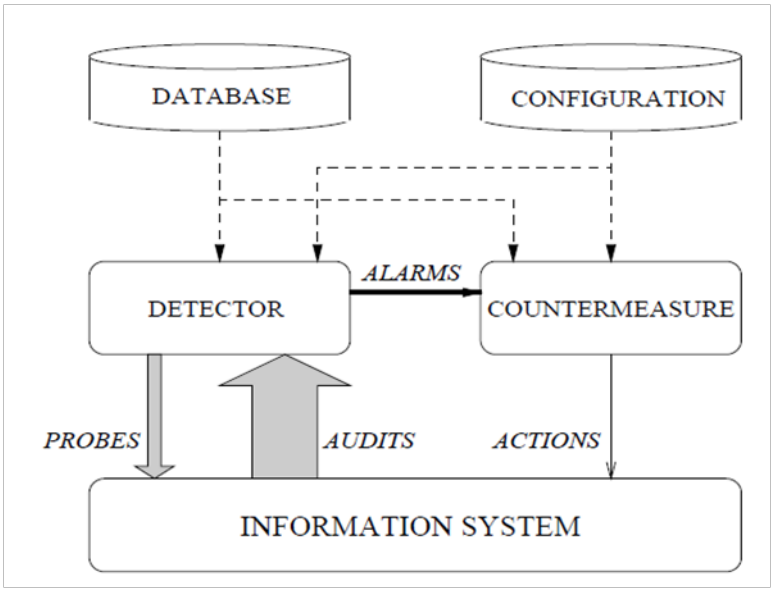


Figure 1.1: Database Intrusion Detection System

known attack patterns (misuse detection), and another way is to check the deviation of observed behaviour from 'normal' or legit user behaviour (anomaly detection).

1.2 Types of security intrusions and attacks in DIDS

As highlighted earlier, databases are prone to a variety of security threats. This section highlights the major types of intruders and security attacks that databases are exposed to.

1.2.1 Types of Intruders

Intruders attempt to penetrate through security systems of organisations in a number of ways. This may involve misuse of privileges or false identity operations. Intruders are classified into separate categories each with different level of vulnerabilities as expressed by Anderson [7] as follows: -

- Legitimate User: When an authorized user, who has access to the database uses the records with a malicious intent.
- Clandestine User: when the user clutches supervisory control of the system and operates below the level of action of audit trails.

- Masquerade: it is someone who pretends to impersonate another user in the system and uses its privileges to illegally access the data.
- External Intruder: when a user who does not have access to the database bypasses the security of the database through loopholes in the system or SQL injections.

1.2.2 Types of attacks

As per Burtescu [8], attacks are classified into the following three types depending upon the intruders' intentions :

- Attacks aimed at compromising integrity: The intruder attempts to compromise the assurance of authorized modification. Thus, the attacker aims to modify data reliability and accuracy, like abusing the access or malicious use, internal attacks, etc. (e.g., masquerading and modification).
- Attacks aimed at compromising confidentiality: The intruder aims at gaining unauthorized access instead of damaging data, such as stealing business information attempted break-ins, data leaking, etc. (e.g., snooping).
- Attacks aimed at compromising availability :These majorly include denial of service (DoS) attacks where the attacker aims on making database services inaccessible to authorized users(e.g. flooding database services , etc).

1.3 Security requirements in DIDS

In the database, all the data regarding organization is stored and managed. The information stored in databases is very critical for both clients and the organization. Also, the infrastructure of databases is more complicated due to the different and large number of stakeholders. Due to this,the database is imposed to multiple attacks and threats. To handle the database attacks and threats, essentially each user must know about the implemented security ,the privacy strategy and potential risk associated with it. Databases need the following requirements to assure clients of a degree of security against such risks:

- Data confidentiality is needed to prevent unauthorized access that could lead to adverse results.

- Data Integrity to ensure accuracy and consistency of the data.
- Communication entries must be mutually authenticated, which provides secure communication.
- Databases also need to ensure availability of the services and data to the users.
- Databases also need to provide non-repudiation to ensure that after a successful transaction, both parties cannot deny the transaction.
- In some instances, which involve the protection of user identity require anonymity and non-traceability.
- In emergency cases, the devices and system must be compatible for the client so that their data is easily accessible.

The developed rules and protocols should be supported and enforced the above-mentioned properties and security requirements.

1.4 Motivation

Databases are essential to an organization's efficient operation. Databases hold essential operational data, making them vulnerable to many dangers from both inside and outside the organisation. As a result, protecting databases against intrusive access attempts becomes an essential component of the System. Databases are primarily vulnerable to internal and external threats. While it's critical to protect databases from unauthorised access by sources outside of the organisation, it's equally crucial to protect sensitive data from corporate personnel abusing their access rights. By observing how information systems are used and spotting the presence of unsafe situations, IDS can identify assaults on those systems.

In essence, an intrusion detection system gathers data from a computer system, analyses it for security flaws, and reports the findings to the administrator [9]. Debar et al.[10] provided one of the first taxonomies for diverse intrusion detection systems, highlighting important features for the field. Intrusion detection systems can be either host-based or network-based. In order to find dangers, the former work by recording and analysing

network data. The single-purpose hosts that make up these IDSs are designed to intercept network traffic in diverse network areas and transfer it to a single administration panel. As described by Bertino and Sandhu [11], DIDSs are implemented to identify and prevent security violations in a database and observe discrepancies in user-access patterns to detect the intrusive attempts of accessing sensitive information in a database. Lundin and Jonsson [12] advocated that DIDS are categorized into different types. The Signature-based database IDSs by Denning [13] are employed for the detection of known patterns of attacks. However, they are susceptible to novel attacks and require regular updates and these real-time signature updates introduce a large overhead. Another approach for dynamic signature recognition presented by El Rahman [14], has considered stroke related features followed by outlier analysis which tends to enhance run-time performance for differentiating genuine and forgery signatures. In the working of non-signature database-based IDS developed by Vaccaro and Liepins [15] , the input transactions are evaluated against a known user pattern, and the resulting divergence between the probability distributions categorizes the input as intrusive or normal activity. These systems have turned out to be efficient in dealing with insider attacks.

Numerous systems and schemes have been proposed for intrusion detection at the network and OS level [16, 17, 18]. They have shown promising results in detecting a typical behaviour on the part of the user or application. But very few approaches have been suggested keeping in mind the specificities of databases [19]. Many actions that are malicious for database systems are not detected as a threat by systems tailored for OS and network needs. Databases establish clear usage policies and stringent access protocols in place since they need enhanced security measures. So, approaches that suggest patterns and profiling in a non-trivial manner specific to the needs of database systems are more effective.

The hazard posed by intruders who are authorised users is a crucial factor to take into account when designing database systems. They frequently misuse the access rights, private details about the database schema, and security measures. They can therefore last for longer periods of time and do more harm. The OS and network systems'IDS are not very effective at identifying such insider threats. As a result, the requirement for a DIDS increases.

- Not all malicious actions on database applications are harmful for the network system or OS .
- IDS formulated for the network systems is not completely effective for database system protection.
- The greatest hazards are posed by internal intruders who abuse their user rights to access the database system.
- IDS created for OS and network systems are incapable to defend against these threats.
- These risks are far more challenging to eliminate since they are directly tied to the database's authorised users, who have access privileges to the data.

1.5 Problem statement

From the above motivations, the problem statement is stated as follows:

To develop an adaptive DIDS which can effectively detect malicious transactions from both insider as well as outsider threats.

This problem can be further subdivided into the sub-problems as follows.

- Determining appropriate caution level
- Real-time intrusion detection
- Damage evaluation
- Attack type assessment
- ID efficiency progression
- Data burden capacity
- Evolution of general-purpose ID
- Lack of benchmark datasets for DIDS

To study and evaluate the progress of information security techniques specific to intrusion detection systems for databases, the answers to the following questions need to be discussed:

1. What has been the usual methodology for handling security for the wide gamut of intrusions and database security breaches?

2. Are the methodologies used in conjunction with each other or alone and how effective have been these methods for different types of intrusions?

3. How the DIDS approaches have been tailored to account for dynamic data?

4. What has been the effect of present state of the art DIDS solutions on detection and false alarm rates?

1.6 Research objective

From the above discussed problems, two major issues have been found in database security. First one is with the present rule classifier, which is mainly build upon frequent sequential pattern based mining and cannot detect insider attacks. Most of the models do not consider the ever-changing behaviour of users during the accessing time of data. The second is that most of the DIDS don't use user activity parameters, which if used efficiently can turn out to be a good defense against insider attacks.
The main objectives of the work are:

- Developing a novel approach to integrate the benefits of various mining techniques for reducing the high false positive rate.

- Developing a system that dynamically adjusts the sensitivity parameters of the system to make it more adaptable to changes in user access pattern.

- Implementing a mechanism to identify and maintain relevant user profiles and automatically reducing flagging of users with legitimate access.

- Detecting and preventing Intrusions in databases by analysing behavior of user access patterns and privilege abuse identification by Role Based Access Control Model (RBAC) to increase the accuracy of the system.

1.7 Proposed Solution

To solve the above-discussed problems, this thesis introduced a user profile clustering-based approach which will consider changing behaviour of the user and will make DIDS robust towards handling insiders' threats.
Intrusion detection systems are a single problem-single solution approach and cater to specific needs. It is seen that it is incredibly difficult to detect a wide gamut of intrusions and therefore two complementary trends have emerged with their share of strengths and weaknesses. It is quite difficult to design systems that conform to the one-solution-for-all-problems paradigm. For example, when Intrusion Detection Systems were first designed, they were limited to a mainframe computer and its users. Host-based IDSs greatly simplified the intrusion detection task since an interaction from outside was sparse and unusual. As networks and the Internet gained popularity, Network-based IDSs were designed to cater to newer kinds of vulnerabilities that soon transpired.

The following important trends in intrusion detection have been studied:

1. Misuse detection: the search for evidence of attacks based on knowledge of previous attacks

2. Anomaly detection: the investigation for deviations from the normal state of a system.

Anomaly and misuse detection methodologies are distinct, and have their own set of advantages. There have however been very few successful IDSs which can differentiate between the various types of intrusions.

- Role based classifier

  The proposed anomaly-based approach focuses on mining data dependencies between data items in the database using various association rule mining. The data dependencies are mined using the transactions from the database log. The transactions which do not comply with the data dependencies are considered as malicious.

- Role profile clustering

  This approach, unlike any other, does not have records and assumes a predetermined policy to be maintained in an organisational database and can operate seamlessly

on databases that follow Role Based Access Control as well as on those which do not conform to any such access control and restrictions. This is achieved by focusing on pre-existing logs for the database and using the various clustering algorithms to allot role profiles according to the database user's activities. These clusters and patterns are further processed into an algorithm that inhibits generation of unwanted rules followed by prevention of malicious transactions.

- User activity parameters-based clustering
  Cluster user behaviour vectors to model general behavioral patterns of employees accessing the database.

1.8 Research outcomes

The primary contributions of this thesis work, including the research goals and problems, are:

- Proposed a novel data mining-based approach FADDRM. The algorithm extracts data dependency rules above a minimum confidence value for a given set of fuzzy association rules with sufficient support values. These data dependency rules are then used to classify the user query. The accuracy will be evaluated in terms of False Positive Rate and True Positive Rate with respect to dependency factor. The comparison results are discussed in chapter 3 .

- Proposed a novel approach towards DIDS based on EMSPM. Our EMSPM algorithm uses two techniques to classify an incoming transaction as malicious or not namely IPPS (Iteratively Pruned Prefix Span) algorithm and EM (Expectation Maximization) clustering. The performance metrics used to carry out the assessment were accuracy, precision, recall and F1-score. The details of outcomes are discussed in chapter 4 .

- Proposed BPSOMQD, an advanced approach that detects and prevents malicious transactions from disrupting the consistency of the database. This method incorporates frequent closed sequential pattern mining which forms the basis for generation of data dependency rules. To measure the performance of the proposed approach, the three metrics that were taken are precision, recall and F1-score. Experimental

evaluation of the proposed approach in section 5.1 shows remarkable results on a characteristic banking database.

- Proposed a unique method for DIDS build on Frequent Sequential Pattern mining and a modified metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm (FPGWWO) .To evaluate the efficiency of the model Precision, recall, and F1-score were used in section 5.2 .

- Presented a novel approach TFUBID to dynamically determine the malicious transactions using historical data. Since groups of users access the organisational database for similar purposes, we cluster user behaviour vectors based on trust factor to model general behavioural patterns of employees accessing the database. Results of a comprehensive experimental evaluation based on accuracy, precision and F1-score parameters of TFUBID is explained in section 6.1.

- Proposed an OIUWSPM in which the mining of sequences is done according to a weighted approach , and the categorization of a transaction involves several security checks conducted at multiple levels with differing degrees of specification. Performance analysis based on Precision , recall , accuracy and F1- score is done in section 6.2 .

1.9 Organization of the thesis

This thesis has been organized into the following chapters:

Chapter 2 Background and related work

This chapter introduces the notion of Database Intrusion detection system. It has introduced various existing techniques to solve the problem of intrusive transaction detection and recent research work regarding DIDS and various implementations. It also contains several clustering-based approaches for role profile clustering.

Chapter 3 Developing a novel approach to integrate the benefits of various mining techniques for reducing the high false positive rate.

This chapter deals with, a new data mining-based approach Fuzzy Association Data Dependency Rule Miner (FADDRM) has been proposed for detecting malicious trans-

actions. The proposed anomaly-based approach focuses on mining data dependencies between data items in the database using fuzzy association rule mining. The data dependencies are mined using the transactions from the database log. The transactions which are not compliant to the data dependencies are treated as malicious transactions.

Chapter 4 Developing a system that dynamically adjusts the sensitivity parameters of the system to make it more adaptable to changes in user access pattern.

This chapter deals with, a novel approach towards database intrusion detection systems (DIDS) based on Expectation Maximization Clustering and Sequential Pattern Mining (EMSPM). This approach unlike any other does not have records and assumes a predetermined policy to be maintained in an organisational database and can operate seamlessly on databases that follow Role Based Access Control as well as on those which do not conform to any such access control and restrictions. This is achieved by focusing on pre-existing logs for the database and using the Expectation Maximization clustering algorithm to allot role profiles according to the database user's activities.

Chapter 5 Implementing a mechanism to identify and maintain relevant user profiles and automatically reducing flagging of users with legitimate access.

This chapter deals with, two novel approaches. Firstly, BIDE and modified Particle Swarm Optimization clustering based malicious query detection (BPSOMQD), an advanced approach that detects and prevents malicious transactions from disrupting the consistency of the database. This method incorporates frequent closed sequential pattern mining which forms the basis for generation of data dependency rules. Further to recognise anomalous user activity, modified Particle Swarm Optimization(PSO) algorithm is proposed which is used to generate role profiles associated with the transaction. Secondly, we proposed FPGWWO, a frequent sequential pattern mining and a modified metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm to determine malicious transactions in RBAC and non RBAC supervised databases. Our proposed approach extracts data dependency rules from the database logs using CM-SPADE mining algorithm to detect the outsider threats. It then assigns role profiles to the users based on the previous user activities using the modified metaheuristic clustering to detect the insider threats.

Thereby, identifying incoming transactions as malicious by matching the role profile of the user and comparing the adherence of the transaction pattern to the generated dependency rules.

Chapter 6 Detecting and preventing Intrusions in database by analysing behaviour of user access patterns and privilege abuse identification by Role Based Access Control Model (RBAC) to increase the accuracy of the system.
This chapter deals with two novel approaches. Firstly, a novel approach to dynamically determine the malicious transactions using historical data. We propose a Trust factor-based analysis of user behaviour using sequential pattern mining for detecting intrusive transactions in databases called TFUBID. Since groups of users access the organisational database for similar purposes, we cluster user behaviour vectors to model general behavioural patterns of employees accessing the database. Secondly, Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining (OIUWSPM) is a novel method for detection of malicious transactions through a sequential flow from outlier detection followed by different behavioural checks at the role induced rule mining component and user level feature extraction. In the worst case, a transaction has to go through a triple fold security validation directing the model from generalisation to specification.

Chapter 7 Conclusion and future scope
This is the last chapter of this thesis. In this chapter, the contributions of the work are summarized. It also outlines the conclusion and future scope of this research work.

1.10 Summary of the chapter

This chapter presented an introduction of the thesis. It discussed the present Database Intrusion Detection System, technologies used in the past and possible security attacks. The security requirements motivated us to develop a new access security model which recognizes user changing behaviour. The motivation from security model developed, led us to formulate the research objectives. The proposed solution and research outcomes have been included in this chapter. The next chapter summarizes the background and

literature work that has been investigated during the research work.

Chapter 2

Background and related work

This chapter provides the present background knowledge, state-of-art survey of the domain along with the recent trends to understand the concept of Database Intrusion Detection systems(DIDS).

In this chapter, the notion of Database intrusion and prevention are discussed in Section 2.1, DIDS and their limitations are elaborated in Section 2.2. In Section 2.3 models with rule classification and role profile clustering are discussed. In Section 2.4, research challenges are discussed. Section 2.5 discusses the summary of the chapter.

2.1 Intrusion Detection Systems

An intrusion is defined [20] as any set of events aimed at jeopardising the system's authenticity, security, or accessibility.

Intrusion detection is the process of determining whether or not an effort has been made to breach a system, or, whether or not the system has been already hacked. Instead of explicitly detecting intrusions, Intrusion Detection System (IDS) gather evidences or indications of invasion of privacy, either while they're happening or after they've happened. [21]

A *Database Intrusion Detection (DIDS)* is a database security/threat detection technology that examines database transactions to detect vulnerability exploits. Vulnerability exploits are typically delivered as malicious inputs to a target database, which attackers use to disrupt and take control of an application or machine.

Chagarlamudi et al. (2009) [22] proposed an approach to prevent malicious insider activities at the database application level. User tasks were implemented using Petri nets

and tests were carried out to check whether transactions conformed to a predetermined order or not. In another example, Denning (1987) [13] underlined the importance of a real-time intrusion detection system based on the hypothesis that intrusions into the system involve abnormal usage of the system. The abnormal usage may include a high rate of login failures, unusual time access, unusual data access patterns, etc. An intrusion detection expert (IDE) system is discussed as a separate entity which adds an extra layer of security for the database.
A lot of work has been done in the field of DIDSs since their inception. The field has grown at a very rapid pace, with a large number of publications emerging every year to further improve upon the existing state of the art solutions.

2.1.1 Importance of Detection of Intrusive Transactions

Database management systems have increasingly made accessing and concurrently modifying data easy and efficient. These systems have also led to an increase in illicit access to the database by exploiting various vulnerabilities.

- Although firewalls act as a security mechanism to monitor incoming and outgoing traffic, they have been found to be under-prepared to protect the database from a wide range of attacks and therefore need to be complemented with intrusion detection systems.

- An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of the resource as suggested by Sandhu et al. (2000)[6] Given the importance of data in an economic context, any limitation is a critical issue that might make the currently available systems unsuitable or even incompetent solutions for many database systems. Hence the development of an intrusion detection system which defends the system against illegal access is crucial.

2.1.2 Major approaches to detect intrusive transactions

There are two major approaches to conquer the problem of detecting intrusions in a database. One is to check the queries against well-known attack patterns (misuse detection). Another way is to check the deviation of observed behaviour from 'normal' or legit

user behaviour (anomaly detection).

- **Anomaly Detection:** The existing behaviour of the system is defined as acceptable behaviour and the deviations from this acceptable behaviour are detected to find intrusions[23] . Anomaly detection is further classified into static and dynamic anomaly detection.

- Static anomaly detection keeps an eye on parts of the system that should stay the same. Static anomaly detectors can look for differences in system code and data.

- System behaviour is defined by dynamic anomaly detectors. This type of behaviour is frequently described as a series of occurrences. The use of statistical distributions with means and standard deviations can aid in determining what is normal and what is deviant.

- **Misuse Detection:** Certain possible patterns to penetrate inside a system for attacks are first defined in the form of a set of actions or strings. The activities of users are then matched to these patterns and a signal is raised when the match is found, indicating an intrusion in the system.

2.1.3 Types of Intrusion Detection System

2.1.3.1 Network Intrusion Detection

The intrusion detection network monitors packets transmitted over a network. They are also termed sniffers for packets. They usually have a signature database to compare network packets. These solutions were unable to operate in switched contexts, encoded networks and high-speed network environments. By integrating the network intrusion detection directly into the hardware, Cisco's newest range of switches seeks to solve these challenges.

2.1.3.2 Host-based Intrusion Detection

IDS systems that monitor activities on a host are known as host-based intrusion detection systems. Internal threats benefit from their ability to monitor and respond to specific user behaviours and file accesses on the host. They provide audit policy management centralization, as well as forensics and statistical analysis.

2.1.3.3 Hybrid Intrusion Detection

Hybrid IDS systems manage both network and host-based systems. They are some sort of central intrusion detection system and provide NID and HID with a logical layer.

2.2 Types of Detection Methodologies

In order to detect occurrences, DIDS technology uses many approaches. Sections 2.2.1 through 2.2.3 explain the main detection methodology types: signature-based, anomaly-based, and stateful protocol analysis along with their limitations respectively [24].

2.2.1 Signature based detection

Signature-based IDS refers to the detection of attacks by searching for explicit patterns in database transactions as shown in Figure 2.1. Examples of signature are:

- An email with the subject of "application form" however the file attachment filename of "pdf.exe" which resembles malware.
- A log entry for the operating system with a 645 status code, indicating the host audit is deactivated.

Even while signature-based IDS may quickly identify known attacks, it might be challenging to identify novel attacks for which there is no known pattern. As per our previous example of pdf, if the attacker renamed the file to pdf2.exe, a signature seeking for pdf.exe would not be able to detect it. A merchant provides the signatures for all of its items in a signature-based IDS. The IDS must be updated with the signature on time. The simplest detection method is signature detection, which compares the current activity with a set of signatures using string comparison operations. Methods for detecting signatures have a limited grasp of various network or application protocols, and they are unable to follow the status of complicated communication.

2.2.1.1 Limitations of Signature based detection

- Malicious individuals can easily change the attack sequences they utilise in malware and other types of attacks to evade detection.

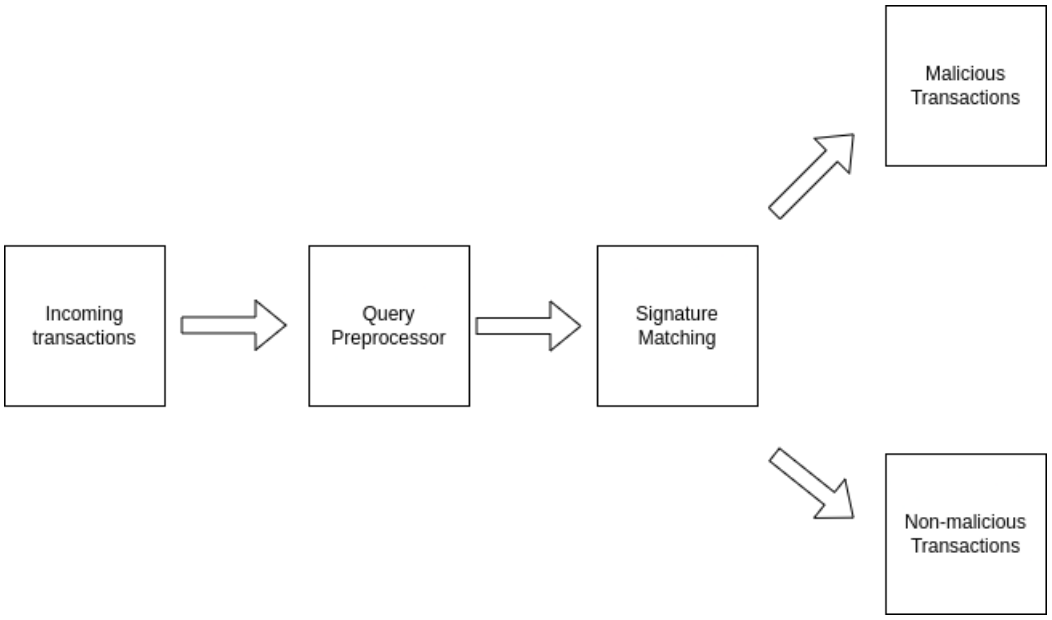


Figure 2.1: Signature based IDS

- The inability of signature-based IDS solutions to detect unidentified attacks is one of their biggest drawbacks.
- In order to entirely avoid signature-based detection, transactions can also be encrypted.
- Intrusive transactions are typically carried out by threat actors who change their signature more than 60% of the time.

2.2.2 Anomaly based Detection:

Anomaly-based IDS systems (illustrated in Figure 2.2) were essentially acquainted with detecting unknown attacks, partially because of the fast malware improvements. The fundamental methodology is to use machine learning to create a model of reliable activity and then compare new transactions or behaviours with this model. Because these models can be trained based on the applications and hardware configurations, machine learning-based methods out perform traditional methods in terms of generalisation.
The main advantage of anomaly based approaches is that they can detect unknown threats easily. Suppose, for instance, that a new sort of virus infects a computer. The infection might eat the computer processing resources, send a wide range of emails, make large numbers of network connections, and conduct a different behaviour than the established computer profiles.

An initial profile is created over a period of time (typically days, also sometimes weeks), which is referred to as a training period. For anomaly detection, profiles can be static or dynamic. After creation of a static profile, the DIDS cannot be modified unless a new profile is specifically generated. As further occurrences are detected, a dynamic profile is constantly modified. Because the systems and networks are changing over time, the appropriate typical behavioural measurements are likewise altered. A static profile will ultimately become erroneous. This problem has no dynamic profiles but they can be evaded by attackers. For instance, an assailant may occasionally execute small quantities of malicious activities thereby slowly increasing the frequency and amount of activity.

If the change rate is sufficiently slow, the DIDS can interpret the malicious actions as

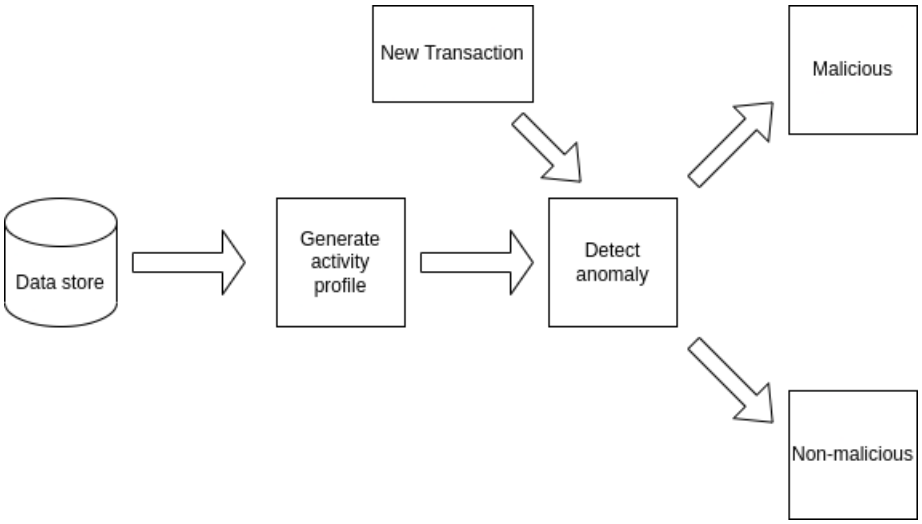


Figure 2.2: Anomaly based Detection

regular behaviour and add it to its profile. Misleading activity can also be seen by a DIDS when its initial profiles are established.

2.2.2.1 Limitations of Anomaly based detection:

- While this method allows for the identification of previously unknown attacks, it may also classify previously unidentified legitimate activities as malicious.
- The majority of existing systems have a time-consuming detection process that degrades IDS performance.
- The classification process employed in detection is more reliable when there is an effective feature selection technique.

2.2.3 Stateful Protocol Analysis based detection

It is a method of deviation detection from observed events by comparison of prepared profiles of commonly agreed innocuous protocol definitions for each state of a protocol, as shown in figure 2.3. In contrast to anomaly-based detection, which uses host or Network profiles, stateful protocol analysis is supported by universal vendor profiles that describe protocol usage and ignorance. The term "stateful"notion of state of network and application protocols.

For example, consider a situation where a File Transfer Protocol (FTP) session is started by a user. Initially the session will be in an unauthenticated state which means the user is supposed to use only some of the specific commands which can be from viewing help information to logging in using username and a password. A key component of the understanding is that the requests are coupled with the responses, so the DIDS may identify the success of the FTP authentication attempt by selecting the status code in the response. The session goes into the authenticated state when the user has been authenticated successfully and now is capable of using all the commands meant for the FTP server.

Stateful analysis of the protocol may uncover unexpected command sequences such as the repetition of the same command. The DIDS may track the authenticator that was used in each session and record the authenticator that was used to conduct suspicious behaviour for protocols that do authentication. Some IDPSs are also able to employ authenticator information for various user or specific user classes to define allowed activities differently.

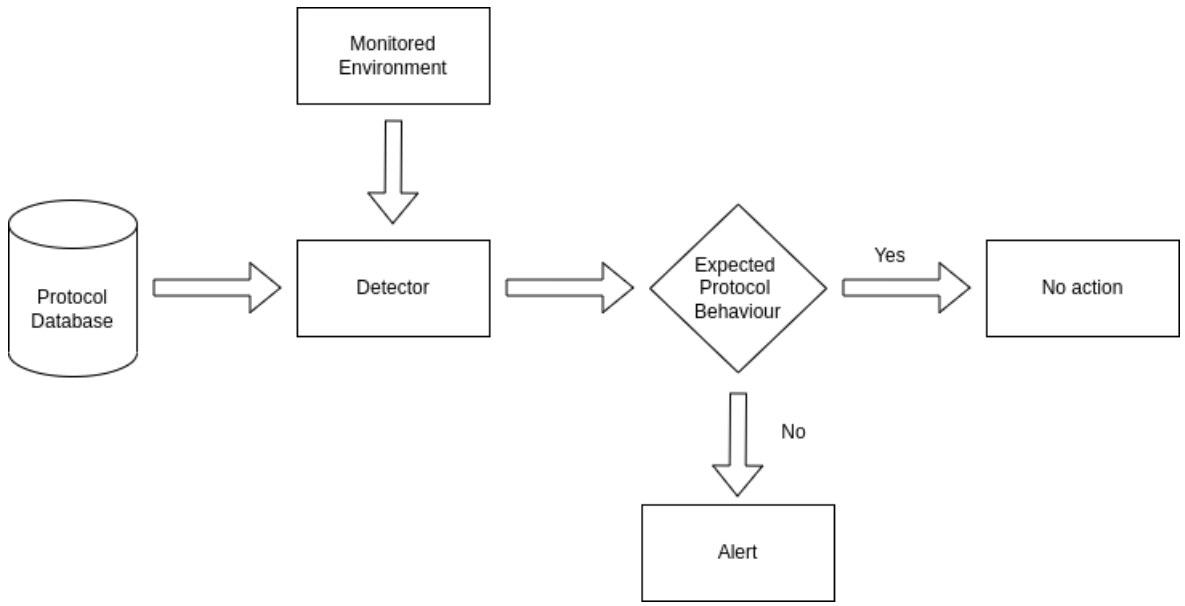


Figure 2.3: Stateful Protocol Analysis based Detection

2.2.3.1 Limitations of Stateful Protocol Analysis based detection

- Intricacy of the analysis and overhead for numerous simultaneous sessions involved in state tracking result in making stateful protocol very resource-intensive.
- The attacks which resemble the characteristics of a generally acceptable protocol can be left undetected by stateful protocol analysis models. Example of DDoS where an attacker performs many innocuous actions in a very short time.
- It may be incompatible with how the protocol is implemented in specific applications and operating systems in various versions.

2.3 Major set of keywords:

The major set of keywords are depicted in Table 2.1.

2.3.1 Data mining & Sequential Pattern Mining (SPM):

Various data mining techniques are used to find relevant patterns in a vast dataset to derive interesting conclusions about the data. One of such techniques is Sequential Pattern Mining or SPM. In sequential databases, SPM is employed to find regularly occurring patterns, significant characteristics, and trends in sequential databases. Data mining techniques including sequential pattern mining have been used in various studies for the purpose of database security [25, 26], Sequential Pattern Mining (SPM) is a data mining subdomain coined by Agrawal et al. and is widely used in areas of database security [11, 19, 26] to discover frequently occurring sequences, interesting features and patterns in sequential databases. Agrawal et al. [88] introduced new algorithms - Apriori and its variations to mine the rules under associations between data items in a database. Apriori Hybrid algorithm provides linear scalability with the size of the database which overcomes the drawback in Apriori algorithm. One major limitation of the algorithm was to settle suitable utility for support and confidence. Agrawal and Srikant [89] proposed AprioriAll and AprioriSome for extracting sequential patterns in a database. Both the proposed algorithms scale linearly with respect to the size of the database but encounter the similar problem of deciding the support and confidence worth's. These limitations were handled when Srikant and Agrawal [90] proposed generalized sequential pattern

	CONCEPTS USED
Data mining	Sequential pattern mining, weighted sequential pattern mining, frequent pattern mining, constraint based FPM, statistical data mining, association rule mining, clustering, temporal analysis, knowledge discovery
	SECURITY CONTROLS ATTRIBUTES
Intrusion detection	anomaly detection, misuse detection, command based, statistics based, alert aggregation, host-based, network-based, signature based intrusion detection, expert system
Database security	Data confidentiality, integrity, availability, accountability, privilege abuse, malicious transactions, insider threats, cyber-attack, information security, non-repudiation
Insider threats	Authorisation abuse, confidentiality compromised, privilege abuse, collaborative attack, weak authentication, information misuse, masquerade, clandestine user, SQL injection attack, hardware misuse, witting/unwitting, data corruption, database misconfiguration
	ACCESS CONTROLS
Access control models	Role-based access controls, access levels, authorisation, user roles, user profiles, transaction profiles, rule based access control
Constraints	Authorisation constraints enforcements, policies, integrity constraints
	APPLICATIONS
Access control application	Authorisation, flexible policies, security policies, privacy preservation, data theft prevention, health management, medical sector, retail sector, military sector, education sector, banking
	ERROR HANDLING
Error detection	Anomalous access patterns, policy violation, intrusion detection, data extraction, audit logs
Error prevention	Damage assessment, regeneration, recovery

Table 2.1: Major Set of Keywords

mining algorithm which was more efficient than the Apriori algorithm as it considers the distinct signs of the items in real-time applications, with the weight being either the cost or the profit of the item being utilized.

2.3.2 Database security:

Database security refers to a set of tools, checks and methods to create and preserve the privacy, integrity and availability of the database. Any related applications to the database management system, the actual database server or virtual database server, the underlying hardware, and the computer and/or network infrastructure needed to access the database must all be addressed and protected by database security [27].

2.3.3 Insider threats:

An insider threat is a security risk posed by one of the three sources with privileged database access [28]:

- A malicious insider who attempted to cause harm
- An unfavourable insider who makes the database vulnerable to attack
- An infiltrator is an outsider who obtains credentials through methods such as phishing or by accessing the credentials themselves

2.3.4 Role based Access Control (RBAC) Model:

RBAC mechanism [6] is a major technique used to detect insider threats in case of database intrusions. It specifies certain roles for various users thus ensuring the database security considering the user behaviour with respect to their specific roles only. The access authority is given to the users only on the basis of their roles in the organization. Any activity outside the access privileges assigned to a role profile is monitored for anomalous behaviour leading to insider threats to the database. The RBAC binds the user to a specific attribute realm as per their roles thus controlling the access to the sensitive parts of the database and ensuring database security at the Role level. This strategy is implemented by extracting sequences and association rules from the valid transactions which are used to verify whether an incoming transaction is legitimate or malicious. The NIST

model further laminated the security system by restricting the user activity to specific roles. Using the Role-based Access Control mechanism the problem of insider attacks is being solved to a great extent. RBAC maps distinct access privileges to users of distinct roles.

2.4 Classification of security controls

Once the main areas for research were established, the next goal was to assess the results to identify the scope of the research, i.e., the research challenges that are to be explored. The classification involved the study of the DIDS approaches, the post detection action, performance metrics and system restrictions.
Thus, the DIDS were categorised according to their underlying principle techniques.Additionally, the system action after detection was used as another classification filter.The classification scheme can also be based on higher-level aspects such as user privacy in DIDS systems, attack classification by the system and access control methods used in the system.
This can be summarised into a four-step pipeline consisting of the following steps:

1. Define and document the scope and goals of the study by defining research questions that are to be answered by the end of the study

2. A thorough and comprehensive literature search which involved both a general as well as a specific search

3. Literature selection by picking publications pertinent to the scope of study

4. Classification of the literature into buckets of the underlying techniques and specific high-level categories resulting in definition of research challenges

2.5 Existing DIDS Approaches:

In the following few sections, developments in the field of database security and DIDS are extensively reviewed, and in the process the years of publication and sources of the selected literature have been outlined. Figure 2.4 traces the progress of the average values of evaluation metrics such as precision, recall and proficiency from 1999 to 2017. Each block's length on the Y-axis represents the average value from low to high, with 1 unit

denoting low, 2 units denoting medium, and 3 units denoting high. Figure below (refer Figure 2.5) illustrates the relative proportions in which such techniques such as statistical analysis, clustering, dependency analysis etc. have been used. Dependency and relation analysis is the most widely used technique in database intrusion detection.

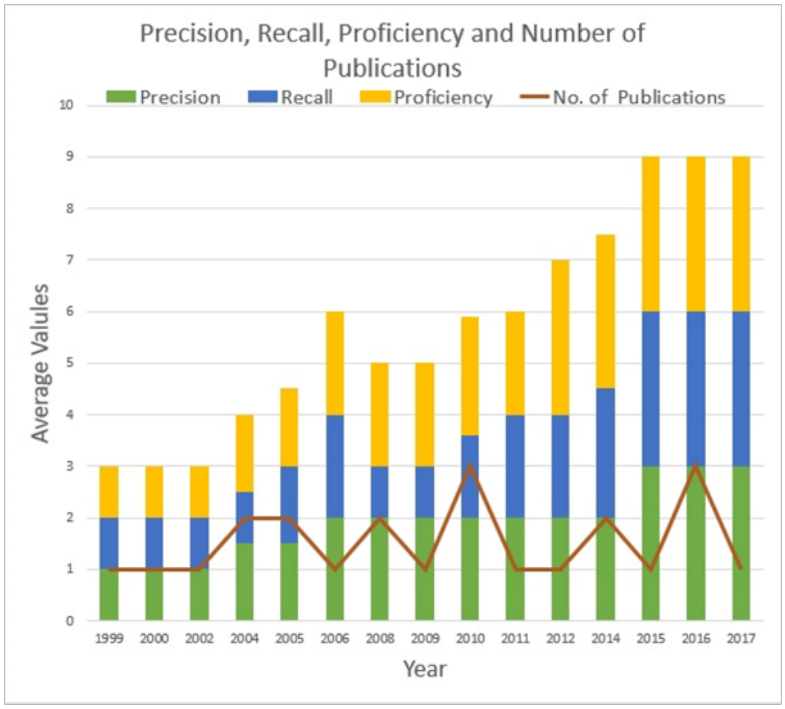


Figure 2.4: Change in accuracy over time

Data mining and knowledge discovery using data access logs have been widely used throughout the length and breadth of research in database intrusion detection. Techniques such as dependency and relation analysis, sequence alignment analysis and information flow patterns use core knowledge discovery principles to mine user access patterns and thereby distinguish intrusions from normal accesses. In the following sections, besides statistical data analysis, some important and landmark research on database intrusion detection involving data mining and pattern discovery have been studied briefly.

2.5.1 Dependency and relation analysis

DIDS techniques that employ dependency and relation analysis use the relations and dependencies between different data items that can be represented by the sequence of accessed items.

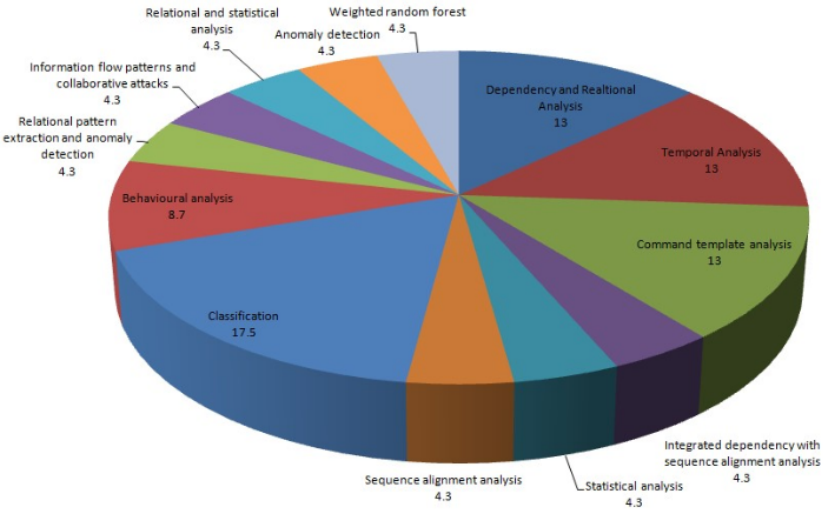


Figure 2.5: Major DIDS Techniques

The DEMIDS approach was proposed by Chung et al. (1999) [29]. It derives user profiles from user access patterns in the database audit logs. The distance of a new query initiated by a user from the derived user profile is measured using distance metrics to detect an anomaly. The user profiles are established through frequent sequences, which are determined by feature vectors. These feature vectors are based on syntactical analysis of user commands containing attributes that are issued together.
Zhong et al. (2005)[30] organise each user query into a tuple. Constraints are used to reduce frequent sequences and improve system performance. Frequent item-sets are formed by mining user query profiles using these tuples. The algorithm uses query distance as a metric to calculate the difference in constraints in any random query and the frequent sequences.

Kamra et al. (2008) [31] proposed a RBAC intrusion detection system as an advancement of Bertino et al.[19] (2005) by developing three models of different granularity to represent SQL queries appearing in the database log files.

Bertino et al. (2005) [11] introduced the concept of sensitivity of attributes, as every attribute is given a measure of importance on the basis of support in the analysis transaction. Rules generated using a weighted data mining algorithm mirrors the dependencies among database attributes. Transactions which do not follow these rules are categorised

as malicious. Subudhi et al. [32] proposed an IDS which utilized OPTICS clustering along with Ensemble Learning comprising of several aggregation methods like Boosting, Bagging and Stacking for the identification of malicious users in the database.

2.5.2 Temporal analysis

These techniques involve use of the real time characteristics of data for intrusion detection, such as the interval between user activities and the time between those actions. For capturing time semantics of data objects time signatures are applied in these techniques to detect violations of the security policy.

The proposed model (Lee et al., 2000) [33] is constructed from time signatures by integrating standard deviation and mean in the system. The real time system works on a previously defined time span in a real time system. In the proposed approach, transactions are defined as a series of read and write queries for every data attribute performed in a specified update time-period. It also results in an efficient performance on a distorted dataset.

2.5.3 Command template analysis

Command template-based DIDS matches the raw transactional log of SQL queries against incoming database transactions efficiently. The repeated occurrence of a set of SQL statements is listed according to the frequency of each learnt fingerprint. A parse tree of commands was used to correlate commands with applications and to distinguish between malicious and legitimate queries.

Lee et al. (2002) [34] proposed a system in which a set of legitimate queries are summarised into SQL templates which represents the models of all the queries. A Kolmogorov-Smirnov test, at 90% confidence level is implemented to decide whether an unrestricted variable or limited number of values should be employed. The frequency of each fingerprint is also calculated.

Bockermann et al. (2009)[35] proposed a grammar-based analysis of frequently used vector-based data using tree-kernel based machine-learning techniques. Commands are

correlated using the parse- tree structure of SQL. The changes in command syntax tree are analysed to determine the authenticity of the user-entered command. In the learning phase, SVM is implemented. Clustering is used for outlier detection and determining the authenticity of user-entered commands.

2.5.4 Integrated dependency with sequence alignment analysis

Some data items may be read or written prior to updating a particular item. Data items continue to be read and written after the update. These updates read and write database access operations are stored in DBMS audit trail that can be used for finding data dependency relationships. These dependency relationships are then quantified and evaluated using Sequence Alignment technique. It detects the longest common subsequence and measures the similarity of a new database access sequence with existing normal sequences and helps in anomaly detection.

In 2006, Srivastava et al. [36] proposed a weighted data dependency rule miner (WDDRM) intrusion detection algorithm for interpreting data items dependencies. Transactions that did not adhere to the data dependency rules which were extracted were considered as malicious. Every attribute is associated with its sensitivity value in the form of weights.

In 2004, Hu and Panda [25] proposed a method to establish dependency relationships in transaction-level attributes with high support and confidence. An update statement triggers a progression of events, i.e., updating a given attribute can result in reading or writing other attributes in the database. A data mining approach for detecting malicious and intrusive database transactions was proposed by the model. The approach focuses on determining data dependencies between data items present in the database. Transactions that do not satisfy the dependencies mined are flagged as malicious. The approach works effectively provided data dependencies exist in the database.

Fonseca et al. (2008) [37] employed directed graphs to represent transactions. Transactions which do not comply with learnt execution paths are considered malicious. The command type, target objects, selected columns, and restriction attributes are used to

build the execution paths. These attributes are retrieved from typical DBMS audit entries.

2.5.5 Statistical analysis

For dealing with the insider threat problem, statistical analysis is used by building a statistical profile of normal user's behaviour and raising an alert when a user deviates from his/her routine.

Spalka and Lehnhardt (2005) [38] employed arithmetical functions on values retrieved from relations for detecting anomalies. Reference values include average, count, standard deviation, minimum, maximum, ranges, bit counting, zero length checking and computed ratios. Misuse is detected by inspecting objects and operations on them. Each pair of database objects and operations is pre-defined as malicious or not.

Mathew et al. (2010) [39] delineated those statistical values such as maximum, minimum, counting, median, mean, cardinality values and standard deviations are computed for all attributes of a dataset, as a result of each user command. A vector with pre-set dimensions, namely S-vector is used to store these statistics. For large datasets, S-vector is constructed using initial k or randomly picked k tuples. This maintains performance and scalability. Using each user's S-vector, techniques such as naïve Bayes, clustering, support vector machines and decision trees are applied. Statistical deviation and outlier detection is implemented in the detection phase to detect malicious user commands.

2.5.6 Sequence alignment analysis

Sequence alignment is a technique used to evaluate the similarity between a number of sequences. Sequence alignment can thus be applied to find whether a new database access sequence is similar to existing normal access sequences, and can therefore help in distinguishing between database access patterns of genuine users and intruders.

In the approach proposed by Kundu et al. (2010) [40] user profiles are built by analysing sequences of retrieved commands, attributes and tables. The user is also provided with an option to select between organisation-based, role-based or user-based profiles, provided the operational framework of the dataset is available. For deciding the

maximum number of differences, a threshold was given through which sequence patterns were built in the learning phase. The maximum difference between the incoming sequences and the learnt ones was used to detect intrusions in the detection phase. Hashemi et al. [84] extended the existing dependency mining approaches [25, 81, 82, 85] to enhance intrusion detections. The proposed model mined dependencies existing among data items, hence discovering transactions repudiating them and employ- ing a behaviour similarity test between these and the valid transactions in logs.

2.5.7 Classification

Classification is used to classify things/objects, e.g., user role profiles which are helpful in distinguishing anomalous user activity from normal user activity.

In 2010, Rao et al. [41] proposed a novel transaction level IDS to detect data privilege abuse by users of databases. The IDS is well suited for a Role Based System and implements a classifier to generate user role profiles from the normal user activity of a large number of users. The major drawback of the approach is its dependence on the order of attributes referred together in transactions and its inability to obtain correlation among various queries in a transaction.

Sohrabi et al. (2014) [42] overcame this drawback by using leverage as rule value measure to extract infrequent but important rules. This also helped decrease the overall interesting data dependencies and increase the system performance. Rao and Singh (2014) [43] detected the exploitation of user privileges by obtaining data dependencies between attributes in a RBAC administered database.

The roles need to be defined explicitly, therefore reducing the platform independence of the proposed system. This is the major limitation of this approach.

2.5.8 Behavioural analysis

Behavioural analysis is helpful in solving the problem of intrusion attacks due to malicious insiders by evaluating user behaviour leading to formation of groups helping in creation

of routine behaviour pattern eventually leading to ease in anomaly detection.

Sun et al. (2016) [44] illustrated a data driven evaluation model for detecting malicious insiders. They used both parallel and incremental views along with probabilistic models for auditing user behaviours. Anomalies were examined from diachronic view for comparing a user's behaviour with previous data of the user.

2.5.9 Relational pattern extraction and anomaly detection

Community-based anomaly detection system was introduced by Chen and Malin (2011) [45] for detecting malicious insider threat. The assumption that the users form community structures, and users at a significant distance from the community, and having low affinity to the community are indicative of anomalous behaviour. The two major components used were relational pattern extraction along with combination of graph-based modelling, dimensionality reduction techniques and anomaly detection with nearest neighbour and clustering model. CADS deduces community structures from CIS access logs in the relational pattern extraction module and obtains communities that form the core pattern. Following that, CADS employs specific statistical formulations to access whether or not users deviate from the core pattern of communities.

2.5.10 Information flow patterns and collaborative attacks

To address the problem of collaborative insider attacks where two or more insiders work together in order to compromise important data in the information systems, illegal information's flow is used to extract relations among the system components which are fed to the detection algorithm.
Viet et al. (2012) [46] proposed a probabilistic model to tackle collaborative insider attacks by observing the bridge data items and mutual record access probability of a data item. If a hot item, i.e., a mutually accessed data item has probability less than a specified threshold, the access is marked as a normal occurrence; else it is flagged as an anomaly.

2.5.11 Relational and statistical analysis

Chen and Malin (2011) [45] later improvised the community anomaly detection systems (CADS) for detecting anomalous insiders. CADS used an unsupervised learning scheme to identify insider threats using the access logs of community environments. CADS consisted of relational pattern extraction and deviation detection which used k-nearest neighbour model to measure inconsistency. CADS was extended into Meta CADS which was more efficient when users were low. But Meta CADS was sensitive to the number of assumed users and the number of subjects. Another flaw is that it uses an equivalent k for each user, though it is known that size of networks is variable. CADS is also prone to mimicry attacks where an attacker imitates collaborative group behaviour or the behaviour of another user.

2.5.12 Anomaly detection

An anomaly-based IDS generates a normal user profile and any transaction that deviates from this normal behaviour is rendered malicious. Instead of using patterns or signatures, the classification employs heuristics or rules to capture user's normal behaviour.

Hussain et al. (2015) [47] proposed an IDS that worked on an anomaly detection mechanism. They tried to detect any diversion from regular data access by applications. This was done at a transaction level. In the first phase, profiles of applications are created. The profile of every application consists of a record of all benign queries along with their constraints. This phase is called the profile generation phase. The next phase is the anomaly detection phase in which the test query is matched with the profile of the given application. If a match is found, the query is allowed, else an alarm is raised.

Sallam et al. (2016) [48] proposed a technique for detecting anomalies in user accesses by analysing profiles of normal access patterns based upon syntactic and semantic features of past and historic user queries stored in database logs. Any deviation from previous patterns is considered an anomaly. Naive Bayesian classifier was used in RBAC models to detect anomalies. When RBAC is not used, anomaly is detected using the COBWEB clustering algorithm. When there is significant overlap between access patterns, multi-labelling classifier (MLC) is used.

Sallam and Bertino (2017) [49] proposed real time techniques for keeping a track on users' access frequencies and detecting anomalies in reference to insider threats in databases. User access profiles were generated with the help of past access logs in three stages. First stage includes the standard periodicity detection algorithm. Determining the interval of time when each of the periodic queries is likely to be issued constitutes the second stage, while the third stage involves identifying relationships between queries that are generally issued together. These access profiles are then used on subsequent users' accesses for the purpose of anomaly detection.

Sallam et al. (2015) [50] proposed an anomaly detection design particularly for RDBMS. The profiles of users which were used to detect anomalies were based on their interaction with the monitored database during the training phase. They used a naïve-based classifier to search for inconsistency. One of the major flaws in this design occurs when users submit similar queries within different roles. The MAP rule does not produce accurate results if the same queries are submitted during detection, since the probability that any of these queries will occur gets biased towards the roles that submit the queries.

Aljawarneh et al. (2018) [51] gave a hybrid approach to detect network intrusions. Two main components are involved – the first component filters the data using a vote algorithm with information gain and the second component applies a multitude of classifiers like – AdaBoost, random tree, naïve Bayes, etc.
Kim et al. [52] suggested a novel approach of classifying the user's role and influence by extricating characteristics from SQL queries with the help of CNN-LSTM neural networks which in the anomaly detection technique, automatically extricates essential characteristics of database query and LSTM creates the temporary data of the SQL sequence.
Panigrahi et al. [53] used a deep learning model which mainly concentrates on exploiting data dependencies, the normal behaviour of users, and data sensitivity of different transactions to identify intrusion.

2.5.13 Weighted random forest

Random forest described by Chen et al. (2004) [54] is an ensemble machine learning technique involving unpruned classification or regression trees. These trees are generated bootstrap samples of the data by randomly selecting a subset of features from the data. Random forests may suffer from the problem of learning from extremely imbalanced data. Since these random forests are constructed in order to reduce error, these generally tend to be biased towards predicting the majority class. This results in poor accuracy with the minority class.

Ronao and Cho (2016) [55] proposed an IDS especially customised for RBAC-administered databases. Their approach exploits random forest with weighted voting (WRF) and principal components analysis (PCA) as a feature selection technique for detecting database access anomalies. PCA produces uncorrelated and relevant features along with reduced data dimensionality. Non-uniform database query access is administered by RF as it exploits the intrinsic tree-structure syntax of SQL queries.

2.6 Evaluation of the proposed approaches

Intrusion response and prevention may be described as the capability of ceasing intrusions both at the time of their occurrence and also before that. Intrusion response aims at handling an attack in a manner such that the damage caused to the system is minimised. Lee et al. (2000) [33] presented a temporal analysis methodology to detect queries that demand execution outside a pre-described time schedule and prevent their execution to prevent intrusion. Sequence analysis methodology introduced by Kundu et al. (2010) [40] prevents intrusions by withholding user activity once a suspicious set of actions is detected. The methodology however requires a considerable number of actions which make up the set, implying that an intrusion will be detected only after execution of some actions. Solutions constructed upon relational and dependency analysis as described in Bertino and Sandhu (2005) [11] , Chung et al. (1999) [29] , Zhong et al. (2005) [30] , Kamra et al. (2008) [31] and Bertino et al. (2005) [19] are completely capable of implementing prevention of intrusions, as each individual user command is checked and if any of them is found to be suspicious, the execution of their queries is stopped. Therefore,

solutions merging a mixture of sequence and dependency analysis are adequate in performing partial prevention of intrusion.

Kamra et al. (2008) [31] proposed an intrusion detection and prevention mechanism, improving upon the approach they proposed in 2005. They deployed policy matching to deal with the potential intrusions and clearly defined database response procedures. Kamra proposed a collection of rules syntactically similar to SQL.

The abnormality attributes that are used as characteristics for intrusion detection are – user ID, user role, IP address of different sources, client application, object type, schema, date and time of the user's action along with SQL commands with its attributes. The resolutions described by Spalka and Lehnhardt (2005) [38] and Mathew et al. (2010) [39] construct upon statistical analysis and rely mostly on analysis of changes made in data or on their execution results and are thus incapable of preventing intrusions. This implies that an intrusion can be detected only after an attack.

However, Spalka and Lehnhardt (2005)[38] proposed a method which can be used to verify apriori data concerned with the rows that are to be processed. Due to this reason, information theory analysis methodology that has been presented in Lee and Xiang (2000)[56] might accomplish partial prevention of intrusion. A solution built upon command and template analysis as proposed in method devised can completely enable the prevention of the intrusion because of the same reasons as Bertino and Sandhu (2005) [11], Chung et al. (1999) [29] and Kamra et al. (2008) [31].

Apart from the methodologies described previously for prevention and detection of intrusions, various other works on intrusion detection have been published. For example, the approach by Motwani et al. (2008)[57] presents a methodology for auditing SQL queries so as to assess their suspiciousness through a confidentiality and privacy point of view. Pei et al. (2004) [58] present a generic survey describing how mining methodologies can be applicable for detection of intrusions. A detailed survey of SQL injection is presented in from Kindy and Pathan (2011) [59].

The temporal approach presented in by Lee et al. (2000) [33] showcases both intrusion detection and prevention capabilities. Hu and Panda (2004) [25] presented a highly efficient solution which could partially detect and prevent malicious transactions. While the approach presented in from Fonseca et al. (2008) [37] is feasible, it only provides partial intrusion detection and prevention capabilities. Sallam and Bertino (2017) [49] mitigate temporal insider threats by tracking access frequencies providing a real-time detection framework. Bockermann et al. (2009) [35] presented a complete intrusion detection and prevention approach, having lower efficiency.

Viet et al. (2012) [46] presented a highly efficient intrusion detection and prevention platform dependent approach. It is inferred from the comparative analysis that there lies a trade-off between platform dependence, efficiency and intrusion detection and prevention capabilities. Tables 3 and 4 represent the evaluation of the proposed approaches.

References	*Technique*	*RBAC used*	*Threat level*	*Detection level*	*Accuracy*			*Major Flaws*
					Proficiency	*Recall*	*Precision*	
Chung et al. (1999) [29]	Dependency and relation analysis	No	External and user threats	OS Level	Low	Low	Low	• Distance measure among attribute values not defined. • User profiles considered but role profiles are not considered.
Lee et al. (2000) [33]	Temporal analysis	No	External threats towards illegal access of data	Transaction Level	Low	Low	Low	• Assumes a single fixed rate for time updation.
Lee et al. (2002) [34]	Command template analysis	No	External and user threats	Application Level	Low	Low	Low	• Model is not adaptive, as it cannot dynamically adjust to different length based on run time information.
Zhong et al. (2005) [30]	Dependency and relation analysis	No	External and user threats	Transaction Level	Medium	Low	Low	• Low accuracy and low adaptability to user trends
Hu and Panda (2004) [25]	Integrated dependency with sequence alignment analysis	No	External and user threats towards corrupting data	Transaction Level	Low	Medium	Medium	• Works only when data dependencies exist. • Emphasizes on malicious modification of data, while malicious read operation cannot be detected.
Bertino et al. (2005) [19]	Dependency and relation analysis	Yes	External and user threats	Transaction Level	Medium	Medium	Medium	• Does not cover transaction management or semantic integrity.
Spalka and Lehnhardt (2005) [38]	Statistical analysis	No	User threats	Transaction Level	Low	Low	Low	• Data mining techniques in analysis of group of attributes absent.
Srivastava et al. (2006) [36]	Integrated dependency with sequence alignment analysis	No	External and user threats towards corrupting data	Transaction Level	Medium	Medium	Medium	• Only update sequences are considered for malicious checks. • Poses a problem in selection of appropriate value of support and confidence.
Fonseca et al. (2008) [37]	Integrated dependency with sequence alignment analysis	No	External threats towards illegal access of data	Transaction Level	Medium	Medium	Medium	• Low accuracy and low adaptability to user trends. • Dependencies need to be maintained.
Kamra et al. (2008) [31]	Dependency and relation analysis	Yes	Insider through RBAC, external through clustering	Transaction Level	Medium	Medium	Medium	• Scalability issues considering that f-quiplet takes enough memory space for each log entry. • Query semantics have not been taken into account during the generation of quiplets.

References	*Technique*	*RBAC used*	*Threat level*	*Detection level*	*Accuracy*			*Major Flaws*
					Proficiency	*Recall*	*Precision*	
Bockermann et al. (2009) [35]	Command template analysis	No	External and user threats	Database level	Medium	Low	Medium	• Derivation of application specific SQL policies.
Kundu et al. (2010) [40]	Sequence alignment analysis	No	External and user threats	Transaction level	Medium	Medium	Medium	• Detailed transaction query leads to increase in the number of false positives.
Mathew et al. (2010) [39]	Statistical analysis	No	Insider	Transaction level	Low	Low	Medium	• Computationally slow if query results are very large. • Assumes the database is stable i.e. no updates.
Rao et al. (2010) [41]	Classification	Yes	External and insider threats given, roles are defined	Transaction level	High	Medium	High	• Binary representation of information does not scale well as historical data gets longer. Model accuracy reduces with smaller data. • The number of roles needs to be pre-defined. Role profiles are not created automatically. • Does not detect malicious read or write operations in individual transactions.
Chen and Malin (2011) [45]	Relational pattern extraction and anomaly detection	No	Insider, collaborative threats	Transaction level	Medium	Medium	High	• Adjacency matrix representation inefficient for pattern extraction of potentially millions of users. • Performance is sensitive to the number of simulated users in a community.
Viet et al. (2012) [46]	Information flow patterns and collaborative attacks	Yes	Insider	Transaction level	Medium	Medium	High	• Algorithm fails when a large majority data items in the data filed are updated masking malicious write operations.
Chen et al. (2012) [60]	Relational and statistical analysis	No	Insider, collaborative threats	Transaction level	Medium	High	High	• Collaborative attacks by large groups might remain undetected as new social structures might evolve.
Doroudian and Shahriari (2014) [61]	Mining dependencies	No	External and user threats	Transaction level	Medium	High	High	• Algorithm assumes that only a finite set of transactions issue requests to database. • Experimental determination of appropriate settings for minimum support and related constraints which does not necessarily lead to strong data dependencies.

Reference	*Technique*	*RBAC used*	*Threat level*	*Detection level*	*Accuracy*			*Major Flaws*
					Proficiency	*Recall*	*Precision*	
Sohrabi et al. (2014) [42]	Mining dependencies	No	External and user threats	OS Level	Medium	Medium	High	• Detects only malicious write operations. • Does not consider abnormal patterns in the update of the data items.
Hussain et al. (2015) [47]	Application anomaly detection	No	Insider	Transaction Level	Medium	High	High	• The generation of application profiles is static and the system cannot handle systems with dynamically changing query types. • Considerably high FPR.
Rao and Singh (2014) [43]	Mining dependencies	Yes	External and insider threats given, roles are defined	Transaction Level	High	High	High	• Multiple passes over database to generate rules leading to computational complexity. • Roles need to be explicitly defined.
Ronao and Cho (2016) [55]	Weighted random forest	Yes	External and user threats	Root Level	High	High	High	• Limited to insider attacks at the transaction level. Does not consider collaborative attacks. • No explicitly defined mechanism to prevent malicious read operations.
Sun et al. (2016) [44]	Behavioural analysis	Yes	Insider	Transaction Level	Medium	High	High	• In the anomaly measurement, the results highly rely on selection of the parameter k.
Sallam et al. (2015) [50]	Behavioural analysis	Yes	Insider	RDBMS Level	High	High	High	• Semantic features of the queries has not been taken into account for the profile generation. • Attribute independency has been assumed which is not the case in real databases. • Fails when common roles have different access patterns
Sallam and Bertino (2017) [49]	Anomaly detection	Yes	Temporal insider threats	RDBMS Level	High	High	High	• No provision for recovery or prevention. • User profile updation not incorporated. • Restricted application

Author	*Intrusion detection capabilities*	*Detection paradigm*	*Approach for query evaluation*	*Data sensitivity*	*Prevention capability*	*Provision for recovery*	*Platform independence*	*Maintainability*	*Efficiency*
Chung et al. (1999) [29]	Yes	Misuse	Syntax centric	Yes	Partial	No	No	Low	Low
Lee et al. (2000) [33]	Yes	Anomaly	Syntax centric	No	Partial	No	No	High	Medium
Lee et al. (2002) [34]	Yes	Anomaly	Syntax centric	Yes	No	No	Yes	Low	Low
Zhong et al. (2005) [30]	Yes	Anomaly	Data centric	No	Yes	No	Yes	Low	Low
Hu and Panda (2004) [25]	Partial	Anomaly	Syntax centric	No	Partial	No	Yes	High	Medium-high
Bertino et al. (2005) [19]	Yes	Anomaly	Syntax centric	No	Partial	No	Yes	Medium	Medium-high
Spalka and Lehnhardt (2005) [38]	Partial	Both, anomaly and misuse detection	Data centric	No	No	No	Yes	Low	Low
Srivastava et al. (2006) [36]	Partial	Anomaly	Syntax centric	Yes	Partial	No	Yes	High	High-medium
Fonseca et al. (2008) [37]	Partial	Anomaly	Syntax centric	No	Partial	Yes	Yes	High	Medium
Kamra et al. (2008) [31]	Yes	Anomaly	Syntax centric	No	Yes	Yes	Yes	High	High-medium
Bockermann et al. (2009) [35]	Yes	Anomaly	Syntax centric	No	Partial	No	Yes	Low-medium	Low-medium
Kundu et al. (2010) [40]	Yes	Anomaly	Syntax centric	Yes	Partial	No	Yes	Medium-low	High-medium
Mathew et al. (2010) [39]	Partial	Anomaly	Data centric	No	Yes	No	Yes	Low	Medium-high
Rao et al. (2010) [41]	Yes	Anomaly	Syntax centric	No	No	No	Yes	High	Medium
Chen and Malin (2011) [45]	Yes	Anomaly	Data centric	No	No	No	Yes	High	Medium
Viet et al. (2012) [46]	Yes	Anomaly	Syntax centric	Yes	Yes	No	No	High	High-medium
Chen et al. (2012) [60]	Yes	Anomaly	Data centric	No	No	No	Yes	Low	High
Doroudian and Shahriari (2014) [61]	Yes	Anomaly and specification-based	Syntax centric	No	Partial	No	No	Medium	Medium-high
Sohrabi et al. (2014) [42]	Yes	Anomaly	Data centric	Yes	No	No	No	Low-medium	High
Rao and Singh (2014) [43]	Yes	Anomaly	Syntax centric	Yes	Partial	No	No	High	Medium
Ronao and Cho (2016) [55]	Yes	Anomaly	Syntax centric	No	Yes	No	No	Medium-high	High
Sun et al. (2016) [44]	Yes	Anomaly	Data centric	No	No	No	Yes	Low	Medium
Sallam et al. (2015) [50]	Yes	Anomaly	Syntax centric	No	Yes	Yes	Yes	High	High
Sallam et al. (2016) [48]	Yes	Anomaly	Both data and syntax centric	No	No	No	Yes	Medium	High

2.7 Research challenges

This section discusses the challenges encountered during our research.

2.7.1 Determining appropriate caution level

DID Systems normally use thresholds to ascertain that some action is an intrusion or not characterised by their probabilities. Determining the appropriate threshold is the major concern as a lower threshold leads to a higher FPR and similarly, a very high threshold can cause high FNR. The aim ought to be that no caution should be discarded instead they can be classified using caution level ranking techniques.

2.7.2 Real-time intrusion detection

The issue with most DIDS is that they detect a threat a-posteriori, i.e., after intrusion has occurred so it presents a serious challenge given the value of data for organizations. That is why a DIDS should be able to detect, react and preclude in real-time.

2.7.3 Damage evaluation

Most DIDS are not able to assess the ramifications of an attack on the database. Considering the importance of databases for various organisations, a decisive issue is to quickly deal with dangerous insider threats to security. So DIDS should have the capability to measure or estimate the potential impact of such an invasion. This will allow classifying invasions as tolerable or critical.

2.7.4 Attack type assessment

Attacks can be categorised into four major classes:

1. Snooping: snooping refers to illegal access of private or confidential information which can be used for personal corporate benefits. Snooping may also involve blackmailing.

2. Modification: it is simply changing the database state which may cause loss of integrity and moreover could be performed in non-obvious ways making it difficult to find a trail.

3. Masquerading: masquerading or spoofing occurs when the attacker impersonates to be an authorised user in order to gain much more privileges than he is authorised for. The attacker basically fools the target system or network into validating it as a genuine device. As a result, the attacker gets illegal access to sensitive information in the databases.

4. Denial of service (DoS): This attack is used to make a service completely inaccessible to the users including the attacker. The purpose of a DoS is to shut down the target by flooding it with a large number of requests, thereby causing huge losses. Impact on performance of web server architecture is investigated using DDoS attack strategies [62].

Therefore, determining the type of attack beforehand and acting accordingly should be a major feature of the Database Intrusion Detection Systems.

2.7.5 ID efficiency progression

Any DIDS should have the capability to improve its performance over the course of time learning from FPR and FNR outcomes. Techniques that allow to progressively adjust the ID parameters to increase efficiency are very important and helpful.

2.7.6 Data burden capacity

It is important to know the limitations of the number of users and the amount of data the IDS can handle efficiently without crashing the main server.

2.7.7 Evolution of general-purpose IDS

Most of the IDS depend on the features of the database so there is a need to develop a general-purpose IDS which is independent of the database attributes on which it is applied.

2.7.8 Lack of benchmark datasets for DIDS

It is well known that performing an actual analysis of IDS techniques would help verify the work carried out and conclude certain facts about it like performance, efficiency, etc. But this requires certain benchmark dataset as prerequisites to be available corresponding to which this evaluation can be performed. For the DIDS techniques there are a very small number of datasets available. Most authors have used TPC-C [63] and TPC-E with the latter being used rarely. Otherwise, authors have generally used synthetically generated workloads and datasets to resemble real world scenarios which make it difficult to generalise the performance expectation from any type of IDS.

2.8 Applications of intrusion detection in databases

Applications of IDS systems depending on the various sectors where they are applied.

2.8.1 Military sector

Military databases store highly confidential information that requires top notch security. Advancements in storage management have also brought about an urgent need to protect databases. Information in a military database must not be accessed by an outsider (outsider attack) or by an unauthorised official (inside attack) because leakage of sensitive information in such cases can be perilous and life threatening. It is integral that consistency in platform, syntactic and semantic consistency along with interoperability and data storage is ensured in military databases.

Intrusion detection systems coupled with Intrusion-resilient DBMS can be utilised for protecting the database against intrusions and also for restoring its consistency in case of malicious attacks. Capabilities of systems must be enhanced by creating tools to search, transcribe, connect, assess, divide, analyse, and proceed on the right data within as small a time-frame as possible. Some of the key areas where the system's capabilities must be improved includes partnership, inspection and decision support tools, pattern analysis tools, foreign language and predictive modelling tools. The system desirable characterstics include adaptable middleware for data location, transparency, query tuning for knowledge discovery and scalability, schema development and metadata management, and ordering unstructured data, thereby preserving all the effects of legitimate transactions executed

between intrusions and their detection [64].

2.8.2 Medical sector

The traditional file/folder concept of processing of data in the healthcare industry has been overshadowed by convenient database management systems. Amongst the various types of databases, the most recurrent are online transaction processing (OLTP) databases which enforce data integrity and prevent data loss.

Automated database systems can be used for syndrome detection [65] by means of techniques like data mining and other ontology-based techniques. Also, biologically-inspired computing [66] plays a key role in the e-health domain. Patients are being categorised for their risk of having Diabetes by using clustering based on evolutionary inspired methods. The varied and complex datasets pose a considerable challenge to the management of medical services. Critical investigations lead to the discovery of fraudulent attempts to disrupt the traditional medical records. Because of the colossal amount of unprocessed data there is a need for techniques to augment and elevate the security of the database to avoid malicious attempts to access/modify this data. Security of a medical database can be enhanced by implementing both mandatory and discretionary security policies of a database to effectively limit unauthorised access to the medical database. The severity of the action taken varies from system to system depending on the degree of confidentiality of a particular data. Classification of various users depending on their requirements and user role hierarchy can effectively solve this problem.

Electronic medical records (EMRs) provide easy access to the patient data. Since this data is often sensitive in nature, it must be ensured that it can be accessed by authorised users only. RBAC and experience based access management (EBAM) have been extensively used in order to enforce authorised access to sensitive patient data [67]. Traditional RBAC model is not well-matched for managing impromptu and dynamic events like patients being shifted among wards, doctors taking second opinions from colleagues, or simply unexpected patient arrivals. Inevitably, most of these systems have anomaly mechanisms in state along with the usual role-based access control for managing these circumstances, which makes these apparatus much more suitable to use. However, with respect to security the operations of anomalies lead to added difficulty and a requirement to perform constant auditing to ascertain that the system is not misused. With an

extraordinary system in place, it allows the users to overrule the normal access control mechanism since technical actions themselves cannot ensure privacy and security. This increases the demand for manual control mechanisms and realisation training for users to reduce the application anomaly mechanisms. However, scrutinising how these access control mechanisms are employed - in what circumstances, to handle what cases – may convey something about how the existing normal access control mechanisms should be altered to enhance them so that they suit the needs of the users, thereby removing or at least decreasing the use of such exceptional mechanisms. Also, it is fascinating to probe if the audit logs contain the required information to track any misuse of such extraordinary mechanisms, or if not – what data is lacking [68], [69].

2.8.3 Retail sector

High-profile attempts to breach data have been plaguing retail in previous years. It is believed by many security experts that the prominent technique of SQL injection has been a major component of such attacks. SQL injection came into heavy use after 1998. It is an attempt to breach security and exploit the weakness of a Web Application linked to the database through insertion of malicious SQL queries into a URI Stem/form field/cookie value for its execution. Processing such SQL injections by a vulnerable application results in rogue SQL statements being issued to the database for accessing, modifying or deleting the content that it will not generally have authorised access to. In worst scenarios, SQL injection can provide an attacker with complete control of the server where the database resides.

These high-profile attacks along with other database breaching techniques against the retailers is not only devastating for merchants but for consumers as well. Merchants suffer in terms of damaged reputation, consumer confidence and lost sales while customers suffer in terms of Personally Identifiable Information such as credit card details etc. being compromised. Data breaches present a significant risk and therefore make every possible defensive avenue imperative in such a scenario. As malicious intruders are becoming increasingly well organised and trained to ape the patterns of legitimate customers in order to steal consumer data, detailed security strategy has become the key to this problem.

Companies must establish broader security systems incorporating IDS containing elements to efficiently contain a breach, preventing it from disrupting the database structure and

restoring the data to its consistent form. General RBAC models can be employed focuses on systemic automated approaches.

2.8.4 Banking sector

In an online banking system with traditional banking methodologies, security is one of the primary concerns. Database transactions are well defined sequences of commands that performs a limited set of predefined operations. Precaution is necessary to ensure that the information is transmitted in a safe and secure manner. The upcoming methods in Online banking system database security are incorporated to monitor and increase the security and integrity of the system. Users in a banking application are only permitted to perform operations available through the application interface. End users are not allowed to execute commonly used commands.

Protecting highly sensitive data against unauthorised excess is crucial for the banking sector. Examples of sensitive data includes social security numbers, bank account and credit card numbers. This is because such data exists in multiple databases making it prone to theft.

It is important to realise the high amount of risk that is associated with the sensitive information stored within the databases through internal audits. In banking and financial organisations, it is essential to:

- safeguard sensitive financial data against data breaches and external attacks on the financial infrastructure (outsider attack)
- protect the database against the attempts made by insiders and privileged users having network authorisation that may cause compromise of sensitive data (insider attack).

All these issues can be tackled by implementing IDS wherein by providing least privilege to employees, the company can ensure that they can access appropriate amounts of data that is essential for them to perform their routine duties.

2.9 Discussion

In this section, the focus is on the significant findings and impact of the research work accomplished in the field of DIDS.

2.9.1 Principal findings

Intrusion detection systems are a single problem-single solution approach and cater to specific needs. It is seen that it is incredibly difficult to detect a wide gamut of intrusions and therefore two complimentary trends have emerged with their share of strengths and weaknesses. It can only be imagined how difficult it would be to design systems confirming with the one-solution-for-all-problems approach. For example, when Intrusion detection systems were first designed they were limited to a mainframe computer and its users. Since external action was rare and sparse, host-based IDSs had significantly simplified the task of intrusion detection. As networks and the internet gained popularity, Network-based IDSs were designed to cater to newer kinds of vulnerabilities that soon transpired. The following complimentary trends in intrusion detection have been studied:

1. Misuse detection: the search for evidence of attacks based on knowledge of previous attacks.

2. Anomaly detection: the investigation for deviations from the normal state of a system.

Many intrusion detection techniques have been designed; however, most of the commercially available designs incorporate only one of the trends. Misuse detection is often easier to implement than anomaly detection systems. Besides, the false positives of Anomaly detection systems have made them inappropriate for commercial intrusion detection. There have been, however, intrusion detection systems that have combined both the trends, such as SecureNet (from SecureNet Consortium).

Anomaly and misuse detection methodologies are distinct, and have their own set of advantages. There have however been very few successful IDSs which can differentiate between the various types of intrusions.

Intrusion detection systems like all data-driven techniques require a large number of verified (as secure) audit trails to model the normal state of a system. Therefore, extra

care needs to be taken in the training phase, based on which IDSs generate the normal acceptable states of the database. With dynamic IDSs that adapt and model expected normal system states as newer verified transactions are committed to the database, come across a novel set of vulnerabilities. An attacker over time might feed valid transactions in the audit logs that increasingly make the IDS vulnerable to a certain attack. A useful analogy for such an attack can be constructed for banking transactions: an attacker might slowly and steadily increase spending from a particular account. The dynamic IDS would thus gradually expect a higher spending from the account. In this way the attacker can eventually spend a lot more from the account in effect fooling the system into thinking the transactions to be valid. Misuse-detectors are often based on signatures, and therefore fail to detect well-known attacks when these are masked and modified with a little variation. Therefore, both dynamic as well as static techniques have their share of problems. In many such circumstances Intrusion Prevention Systems in comparison to IDSs can effectively identify and prevent intrusions.

As far as detection and false alarm rates are concerned, many authors have started to focus more on them and have tried to increase detection rates while decreasing rates of false alarms. While increasing detection rates helps to mitigate threats which might pose danger to the system, decreasing false alarm rates will ensure that useful system resources are not wasted on unnecessary detections thus hampering the performance of the system. Both these help to contribute to the efficiency of the database intrusion detection deployed.

2.10 Summary of the chapter

In this chapter, the notion of database detection and prevention are discussed. After that, signature-based models and their limitations are discussed. Then the literature moves towards the anomaly-based DIDS models, especially rule classification techniques. Then recent advancement in DIDS models is discussed which includes models utilizing both rule-based classification and role profile clustering. In the next chapter 3, proposed fuzzy association rule mining based DIDS is discussed.

Chapter 3

Fuzzy Association Data Dependency Rule Miner

In present days, databases have become a very crucial part in all organizations and hence database security has become very essential. The information stored in databases is quite valuable to the organizations and hence suitable protection is required against a diverse array of malicious attacks. To reduce the susceptibility of the data to these malicious attacks, an Intrusion Detection System [70] has been deployed which enables the early detection of the attacks. But, most of the recent approaches emphasized on detecting malicious attacks at network or operating levels [13]. Intrusion detection systems agglomerate and interpret information from different parts of a network or within a computer to recognize possible security altercation. Conventionally, intrusion detection systems have been categorized as anomaly-based or signature-based. The anomaly-based IDS compares patterns of normal user transactions against observed events to recognize significant deviations. The advantage of such anomaly-based IDS is that the attacker never certainly knows the malicious activities that can go undetected but it makes the alarm mechanisms and the system more complex. In signature-based detection systems, certain patterns are matched to known attack signatures. Unfortunately, it is difficult to maintain state information of signatures of all types of intrusions and also very ineffective at detecting previously unknown threats. The recent hybrid approaches combine the merits of the two types of IDSs [24]. Such systems can also be used against insider threats. To tackle this issue this chapter introduces an anomaly-based approach using fuzzy association rule mining [71] in combination with Fuzzy Connected Clustering (FCC) [72]. To generate the association rules,existing algorithms focus on discretizing the attributes into intervals. However, the intervals might not give a succinct and significant interpretation of the patterns of data. Hence, a probabilistic distribution using fuzzy logic has been considered

in our approach.

The contributions of this chapter are as follows-:

- A new data mining-based approach Fuzzy Association Data Dependency Rule Miner(FADDRM) has been proposed for detecting malicious transactions.
- The proposed anomaly-based technique leverages fuzzy association rule mining to mine data dependencies between database elements. The data dependencies are mined using the transactions from the database log. Transactions that are not consistent with the data dependencies are considered malicious transactions.
- A data set for a typical banking organization is used to demonstrate the proposed technique, and the results show that FADDRM can detect malicious transactions more successfully than the other approaches mentioned.

In section 3.1, the proposed FADDRM model and its working mechanism is discussed. The architecture is thoroughly explained in section 3.2, the algorithm is extensively covered in section 3.3. Lastly, the implementation and performance evaluation of the model is discussed in section 3.4.

3.1 Proposed FADDRM Model

The proposed IDS model works independently and without compromising the operations of the DBMS. The system is built to tackle the transactions submitted to the DBMS with malicious intentions by the attacker.The situation could arise due to a hacked legal account, SQL injection-prone web portal, or user rights violation. If the preliminary security fails, then sensitive information can be acquired by the attacker in any of the scenarios described previously.

We devise an intrusion detection system named as Fuzzy Association Data Dependency Rule Miner (FADDRM) for a relational database management system. The model deploys a fuzzy data mining approach for determination of data dependency rules. The proposed model is less susceptible to change in user behaviors and is less rigid than the other models. For the given set of fuzzy association with appropriate support values, the algorithm extracts data dependency rules above a certain minimum confidence value. These data

dependency rules are then used to classify the user query as either an attack or a regular query, depending on whether it doesn't or does comply with the data dependency rules. The proposed system works independently of the native RDBMS environment and improves system security without obstructing its functioning.

3.2 System Architecture

The architecture of our proposed system consists of two modules i.e the Learning Phase and Detection Phase module. Figure 3.1 shown below illustrates our proposed architecture.

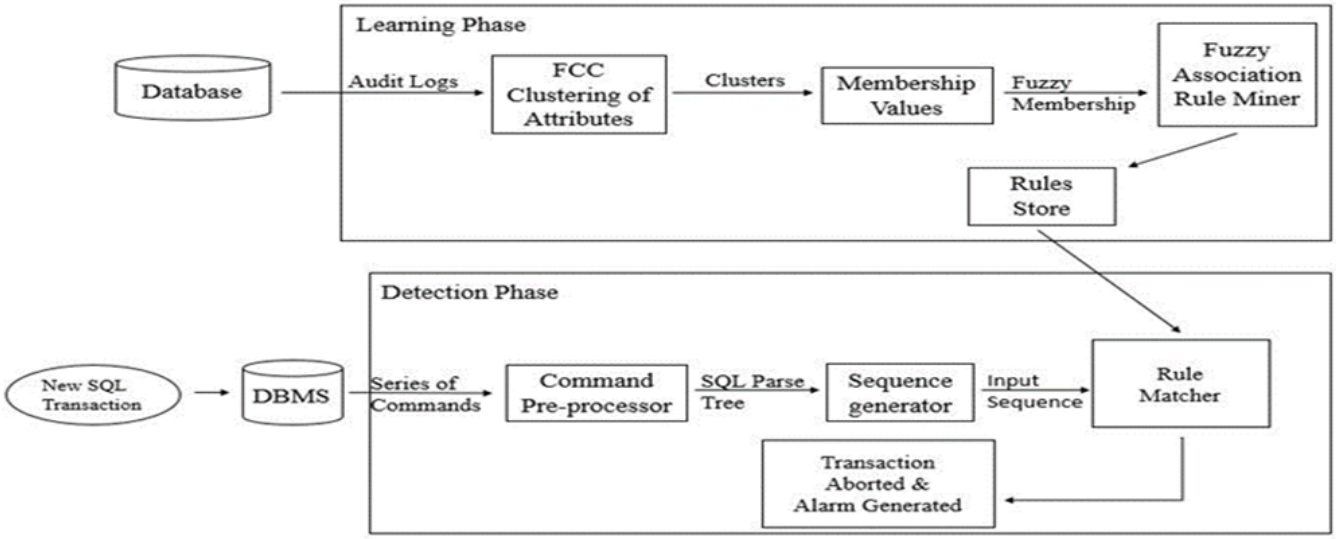


Figure 3.1: Architecture of the proposed IDS

3.3 Algorithm Description

The algorithm 1 begins by limiting the number of attributes being considered for the dependency rules. The next step is the conversion of crisp dataset into fuzzy dataset with membership values assigned to each attribute for a particular cluster. We employed FCC algorithm on the fuzzy dataset to obtain the required clusters. In the next step, fuzzy association rules based on the membership values obtained in the previous step and support for various sequential patterns are determined. In the following section, we demonstrated the procedure for computation of support values. Hence, a set of association rules is obtained. In the next phase, confidence values for these association rules are

determined and only those rules with confidence higher than 80% are kept and rest are discarded.
The following are the major steps involved in the Fuzzy Association Data Dependency Rule Miner (FADDRM) -:

- Step 1:

The attributes from the audit logs are clustered using the fuzzy connectedness clustering technique, FCC. Fuzzy affinity between a new instance (new attribute) and an existing instance belonging to a cluster is determined by spatial cluster distribution and the distance between the new instance and its nearest neighbor for the given cluster. In this way we take into consideration both neighborhood vicinity and global distribution of samples for determination of membership values.

Let E (c, d) be the Euclidean distance between c and d. The mean and standard deviation of Euclidian distance between every pair of correspondences already in the current cluster U_k is given as M_k and S_k. Then fuzzy affinity of (c, d) is given as:

$$\Psi_k(c,d) = \begin{cases} 0, & \text{if } c \notin NN(d) \\ \frac{\rho_{M_k}, s_k(E(c,d))}{\rho_{M_k}, s_k(M_k)} & \text{otherwise} \end{cases}$$

The fuzzy affinity for a given pair is taken as the membership value of the attribute for the given cluster (chain).

- Step 2:

The support for an itemset with a single element is taken as the mean of membership values. For multi-element itemset, the support is determined by the aggregation of products of membership values in each log. For a given set of values, we have:

$$< Y, 1.0 >$$

$$< W, 0.5 >< X, 0.5 >$$

$$< W, 0.5 >< Y, 0.5 >$$

$$< W, 0.5 >< X, 0.5 >$$

The calculated support values will be:

$$\{W\} = 0.375$$

$$\{X\} = 0.375$$

$$\{WY\} = 0.0625$$

$$\{X\} = 0.25$$

$$\{WX\} = 0.125$$

Applying the above procedure, starting with 1-item sets and then going further to N-item sets in Breadth First Manner, we get the set of required association rules. Sequences (item sets) with insufficient support are dropped at each step. Here, we have used single letters to represent attributes.

Algorithm 1:

```
1  begin
2      Initialisation
3      Rd_set{}, Wr_set{}, Rd_rule{}, Wr_rule{}, P = mined patterns with support 'σ'
4      for each pattern S in P do
5          if there exists a sensititve element in S then
6              for each sensitive element X in S do
7                  if RdSq(X) ∉ Rd_set and RdSq(X) ≠ φ then
8                      add RdSq(X) → Rd_set
9                  end
10             end
11         end
12     end
13     for each RdSq(X) in Rd_set do
14         if confidence(RdSq(X)) > confidence_min then
15             add RdSq(X) → Rd_rule with key O(X)
16         end
17     end
18     for each WrSq(X)inWr_set do
19         if confidence(WrSq(X)) > confidence_min then
20             add WrSq(X) → Wr_rule with key O(X)
21         end
22     end
23 end
```

3.4 Implementation and performance of the proposed model

3.4.1 Implementation of the algorithm

The primal requirement for the performance evaluation of an algorithm in the field of intrusion detection is the presence of a standard dataset. However, a dataset which con-

sists of transactions occurring in a pragmatic setting does not exists, as organisations are bound under confidentiality constraints. Hence, we created our own synthetic dataset, mimicking real world transactions. Several experiments were performed on a typical bank database schema generated from TPC-C benchmark [63] to evaluate the performance of the proposed approach. In the given dataset, there are two sets of logs, one containing malicious transactions, and the second includes normal user transactions. The two logs, as described, generated a total of 200,000 transactions.

Suppose we have a database T, with a number of transactions consisting 4 attributes W, X, Y, Z of the form:

```
SELECT W, X, Y
FROM table1, table2
WHERE table1.W = table2.X;
```

All the transactions are scanned to create tidlists for the attributes consisting of the membership function μ which gives the probability of each attribute belonging in all the clusters.

TID	Attributes
1	WXY
2	XYZ

Table 3.1: Tidlists of attributes

W	X	Y	Z
1	1	1	0
0	1	1	1

Table 3.2: Vector representation of attributes

The vector representation of the attributes of two transactions m ("WXY") and n ("XYZ") is defined below.

Each member of the vector contains the membership value, μ of the attribute corresponding to the member index of the transaction. Therefore, the i^{th} member of the vector contains the μ for the i^{th} attribute of the transaction. However, if a transaction does not

0.50	0.50	0.50	0.00	WXY
0.00	1.00	0.50	0.50	XYZ

Table 3.3: Vector representation of the attributes of two transactions

involve the attribute, the member corresponding to that attribute contains a value of 0. The support for an itemset with a single element is taken as the mean of membership values. For multi-element itemset, the support is determined by the aggregation of products of membership values in each log. For a given set of values, we have:

$$< W, 0.75 >< X, 0.25 >$$

$$< W, 0.75 >< Y, 0.25 >$$

$$< W, 0.75 >< X, 0.25 >$$

$$< Y, 1.0 >$$

The calculated support values will be:

$$\{W\} = 0.5625$$

$$\{Y\} = 0.3125$$

$$\{W, Y\} = 0.046875$$

$$\{X\} = 0.125$$

$$\{W, X\} = 0.09375$$

3.4.2 IDS Models for comparisons

Figure 3.2 shows the results of our experiment in this chapter. As the dependency factor increases, the false positive detection rate of the proposed approach also increases. However, it can be followed that the false positive rate of the proposed approach is lower than that of the contending proposed approaches at all values of the dependency factor. The false positive rate of the proposed approach is 26 %, when the dependency factor is 3, compared to 38 % for Hu and Panda [25] and 29.1% of ODADDRM [42].

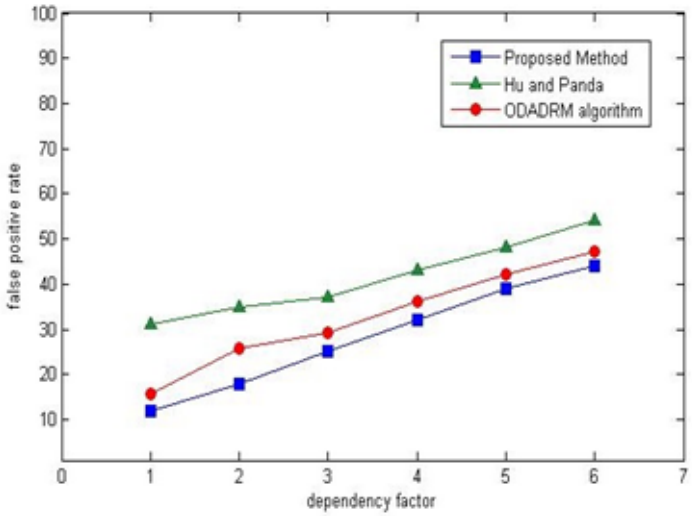


Figure 3.2: False Positive rate of FADDRM with respect to dependency factor

3.4.3 Performative Evaluation

Figure 3.3 represents the true positive rates of the proposed approach at various dependency factors. the true positive detection rate of our proposed approach increases with the increase in dependency factor. However it can be followed that at all the values of the dependency factor, the true positive rate of our proposed approach is greater than that of the contending proposed approaches. When the dependency factor is 3, the true positive rate of the proposed approach is 71.6% whereas, the rate of Hu and Panda approach is 55.4% and that of ODADDRM is 68.1%.

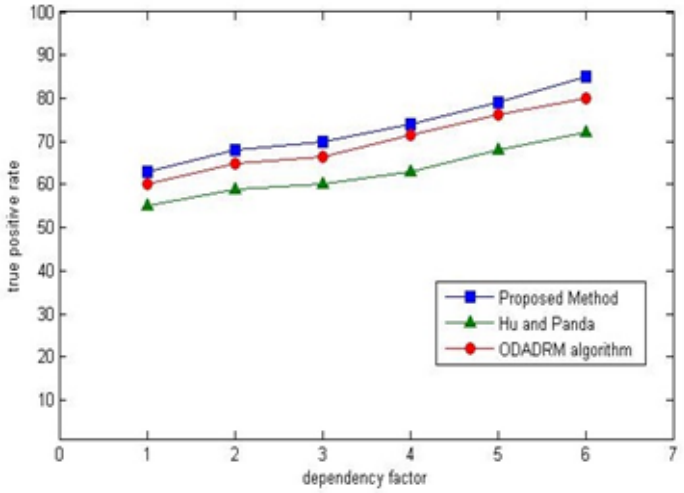


Figure 3.3: True Positive rate of FADDRM with respect to dependency factor

3.5 Summary of the chapter

In this chapter, a novel data mining based approach FADDRM was proposed to detect malicious transactions in a RBAC enabled database. Our method focuses on extracting association rules using fuzzy association rule mining in combination with Fuzzy Connected Clustering (FCC) and the rules are used for detecting malicious transactions. The experiment consisting of 2,00,000 transactions run on a typical banking dataset proves that our algorithm performs better than Yi Hu and Panda's algorithm [25] and ODADDRM algorithm [42]. The proposed anomaly-based approach focuses on mining data dependencies between data items in the database using fuzzy association rule mining; these data dependencies are mined using the transactions from the database log. The transactions which are not compliant to the data dependencies are treated as malicious transactions. The proposed approach is exemplified using a dataset for a typical banking organization and the result shows that FADDRM can detect malicious transactions more effectively as compared to other approaches cited in literature.

Chapter 4

Expectation Maximization Clustering and Sequential Pattern Mining

The previous chapter deals with dynamically determining the malicious transactions using historical data. In this chapter, a novel approach towards database intrusion detection systems (DIDS) based on Expectation Maximization Clustering and Sequential Pattern Mining (EMSPM) is proposed. This approach, unlike any other, does not have records and assumes a predetermined policy that is to be maintained in an organisational database and can operate seamlessly on databases that follow Role Based Access Control as well as on those which do not conform to any such access control and restrictions. This is achieved by focusing on pre-existing logs for the database and using the Expectation Maximization clustering algorithm to allot role profiles according to the database user's activities. These processed clusters and patterns are provided to the algorithm which prevents generation of unwanted rules, followed by prevention of malicious transactions.
The security of data is to be managed for both outsider and insider threats [11]. The CERT Guide to Insider Threats [28] offered specified measures and guidance implemented from executives, managers and other staff in any operational organisation and that too with agile representations. Various measures for cyber frauds, insider threats and theft of information identifications were elucidated with reference to the software life cycle. The effectiveness of the previous security tools was enhanced through rules, configurations and business processes.
The working of an IDS is established on the hypothesis that due to anomalies in usage patterns, the system becomes sensitive towards numerous types of exploitation evident as masquerading, privilege abuse, legitimate privilege abuse and privilege elevation. Any malicious interruption or violation is centrally collected and forwarded to either the common

Admin or the specified Security Information and Event Management System (SIEM).A dominant obstacle in detection and prevention of internal threat arises due to the fact that intruders are legitimate system users with access rights. They are conversant with the information schema and therefore the security mechanism that is employed in situ. Threats posed by corporate executives will therefore continue for extended periods of time without being detected and cause extensive damage to information systems. Such complex environments require a high security level to ensure integrity of information communicated between various entities in an organisation.
The major contributions of this chapter can be stated as follows-:

1. We design a robust DIDS, with the ability to identify intrusions in RBAC as well as non-RBAC administered databases, that employs a rule mining component along with a role-independent anomaly detection component to determine whether a query is malicious or not.

2. We use data dependency mining to explore read and write rules based on the previous access patterns to determine the legitimacy of the transaction under consideration. Further it uses EM clustering to form role profiles from the transactions available in database logs and subsequently assign the incoming transaction to one of these profiles.

3. We propose an approach that reports a substantial enhancement in the nature of attacks that are detected and shows overall improvement in performance than ancient IDS.

4.1 Proposed EMSPM Model

We propose an approach called EMSPM which is a database intrusion detection system consisting of an initial training phase, rule mining followed by clustering and then classification. The algorithm does not assume any fixed access control policy to be implemented on the organisational database, i.e., it functions both when the policy of the database follows Role Based Access Control (RBAC) as well as those that don't follow RBAC. EMSPM uses existing database logs containing transactional details, to mine read and write (dependency) rules along with Expectation Maximization (EM) clustering, to generate role

profiles by aggregating user activity parameters from the information logs. The algorithm extracts relevant patterns, which are then pre-processed to prevent unwanted rules from being generated. First in the learning phase, generation of sequential patterns succeeded by generation of data dependencies occurs from the preprocessed transaction logs making use of the PrefixSpan [73, 74, 75, 76] algorithm. Classification rules above the predefined confidence value threshold are generated related to the mined data dependencies. Further in the learning phase, database logs are analyzed to generate role profiles (clusters) from defined parameters using Expectation Maximization clustering[77, 78, 79, 80]. During this phase, it is assumed that only legitimate transaction logs and database logs are provided. Existing intrusion detection systems are unable to stay robust to dynamic user patterns and do not possess the ability to familiarize themselves with previously unknown attacks. However, the approach described in this chapter has the advantage of being less sensitive to changes in user patterns and capable of dealing with new attacks.

4.2 System Architecture

Our proposed approach i.e. EMSPM comprises of two parts, which are:

1. Learning Phase
2. Detection Phase

4.3 Learning Phase

The proposed learning phase architecture of the DIDS is demonstrated in Figure 4.1. In this phase, transaction logs are used by the rule generator to remove data dependencies, which are then processed during later phases for compliance computation.
Database logs are used to evaluate user access trends in conjunction with conformity estimates, that are further grouped into role profiles. These clusters of roles and rules produced are then transferred to the detection process. The learning phase comprises of two essential components which are -

1. Rule Mining
2. Expectation Maximization (EM) Clustering

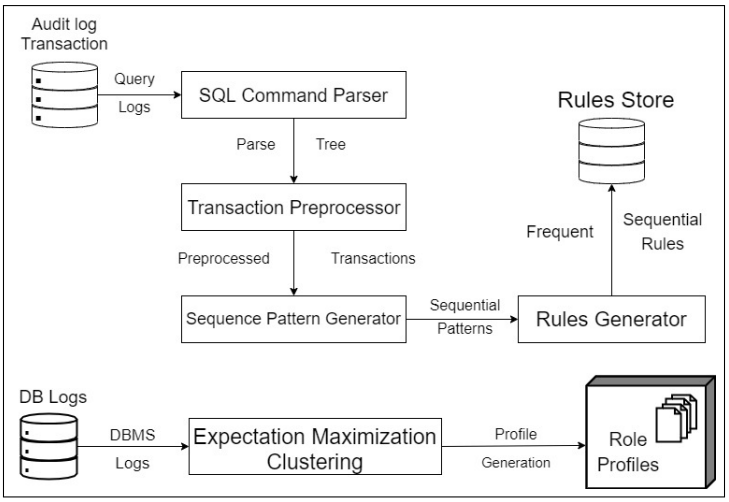


Figure 4.1: Learning Phase architecture of proposed EMSPM approach

4.3.1 Rule Mining

Rule mining refers to the discovery of patterns that represent the intrinsic and important properties of a database. The algorithms extracts relevant patterns which are then pre-processed to prevent unwanted rules from being generated. In the first part of the learning phase, generation of sequential patterns succeeded by generation of data dependencies occurs from the pre-processed transaction logs making use of the PrefixSpan [73, 74, 75, 76] algorithm. Classification rules that are above the defined confidence value threshold are generated from the mined data dependencies.

4.3.1.1 SQL command parser

This parsing module takes standard SQL command as its input and then parses and analyses them to convert the given input into the form of read and write sequences. The parser assigns a unique transaction ID to every transaction in the database. The commands are executed for tokens taken from the source token list. A parse tree is generated if no errors occur while parsing the input into read and write sequences. The generation of parse tree occurs recursively till the base case is encountered.

For example, If the following transaction is examined: -

```
accno=select acc_no from customer where name='Mark' and id=4321;
update Account set bal=bal + 4000 where acc_no=accno;
```

The output after processing would be of the form: -

```
Tid = <R(id),R(name),R(acc_no),R(acc_no),R(bal),R(bal),W(bal)>
```

4.3.1.2 Transaction preprocessor

The transaction pre-processor asserts that transactions are defined in a suitable manner. Each logical transaction should be viewed as a series of queries. Then each query can be interpreted on item-sets as a series of operations (Read or Write). Individual elements in an item-set are lexicographically sorted to avoid multiple representations for the same query.

4.3.1.3 Sequential patterns

Sequence is defined as the order of occurring of transactional operation. Sequential patterns are defined as items that appear in the database log in the same order multiple times.

Definition 1 (Transaction)**.** *is an ordered list of operations to a given database instance, wherein the user operation can be represented with Op* ϵ *{Read, Write}. The execution of the operations must occur in a discrete manner for the transaction to be completed. Transactions are assigned with their unique identification TID.*

Definition 2 (Operation Op(x))**.** *represents the relation with the action of the data item x. Operation consists of action to the set {read , write}.*

Definition 3 (Sequence (Seq))**.** *refers to the ordered set of read and write operations with respect to time. We define a sequence Seq as*
$Seq = \{Op_1(it_1), Op_2(it_2), ..., Op_n(it_n)\}$, *where* $Op_i \in \{R, W\}$ *and* it_i *is an attribute of data item. The data dependency sequences that are extracted from transaction logs and can be classified into two categories:*

Definition 4 (Read Sequence RSeq(x))**.** *is defined for a particular attribute x. We denote read sequence with the form* $RSeq(x) = \{R(it_1), R(it_2), ..., R(it_n), Op(x)\}$ *where* $it_1, it_2, \ldots, it_n$ *represent the attributes that need to be read prior to the transaction performing operations* $Op \in \{R, W\}$ *on attribute x and* $R(it_j)$ *represents a quantum unit of read operation on* it_j.

Definition 5 (Write Sequence WSeq(x)). *is defined for a particular attribute x. We denote write sequence with the form* $WSeq(x) = \{Op(x), W(it_1), W(it_2), .., W(it_n)\}$ *where* $it_1, it_2, ..., it_n$ *represent the attributes that need to write after the transaction performing operations* $Op \in \{R, W\}$ *on attribute x and* $W(it_j)$ *represents a quantum unit of write operation on* it_j.

Definition 6 (Confidence). *provided for a sequence* $Seq \in \{RSeq, WSeq\}$, *is defined as the ratio of the number of occurrences of an attribute* a_i *in a given sequence to the total number of occurrences for that attribute* a_i. *Formally, the definition for Confidence is represented by the given formula:*

$$Conf(Rseq(a_i)) = \frac{Count(Rseq(a_i))}{Count(a_i)} \tag{4.1}$$

Definition 7 (Read Rules(RR)). *For each read sequence* $\{R(it_1), R(it_2), ..., R(it_n), Op(x)\}$, *generated from sequential pattern mining, we generate read rule with the format* $< R(it_1), R(it_2), ..., R(it_n) > \rightarrow Op(x)$, *which infer that before x, we need to read* $it_1, it_2, ..., it_n$. *If the rule generated from the read sequence has confidence greater than provided minimum confidence (conf), then the rule is added to the read rules set.*

Definition 8 (Write Rules(WR)). *For each write sequence* $\{Op(x), W(it_1), W(it_2), ..., W(it_n)\}$ *,generated from sequential pattern mining, we generate write rule with the format* $Op(x) \rightarrow < W(it_1), W(it_2), ..., W(it_n) >$, *which infer that after updating x, data items* $it_1, it_2, ..., it_n$ *should be updated during the transactions.*
If the rule generated from the write sequence has confidence greater than provided minimum confidence (conf), then the rule is added to the write rules set.

Cardinality of Table 4.1. displays the read and write items in a transaction. Each of the ten-transactions has been assigned with a unique Transaction ID.

4.3.1.4 Sequence pattern generator

Since the transaction have been pre-processed into an ordered set of query templates, the task of extracting meaningful rules from data consists primarily in mining frequent

TransactionID	Transaction
101	<r7,r1,r6,w5,r1,w4>
102	<r1,r5,w1,r4,r5,w4>
103	<r1,r6,r2,w4,r7,r3,w6,r1,r6, w2,r3,r5,r2,w5>
104	<r5,r6,w5,r5,w4>
105	<r2,w2,r4,r7,w3,r6,w5,r1,w4>
106	<r6,r5,w5,r3,r4,w4,r3,w7>
107	<r5,r3,r6,w7>
108	<r4,w6>
109	<r2,r5,w6>
110	<r6,r1,w3,r1,w6,r2,r7,r4,w2>

Table 4.1: Transactions for mining Sequential Patterns

Sequential Patterns
r6,w5,w4
r7,r6,r1
r7,r6,w4
r7,r6,w5

Table 4.2: Mind Sequential Patterns (Support threshold)

sequential patterns through successive algorithms. We used the PrefixSpan algorithm [74] to mine the entire set of frequent patterns. The procedure to specify all the subsequences of one's transactional patterns and subsequently computing the respective supports to generate frequent patterns has a high computational cost. The improved Prefix-Span (Prefix-projected sequential pattern mining) developed by Han et al. (2001) [73] runs this process with a lower computational cost compared to both Apriori-based GSP [81] and FreeSpan [82] algorithms. The output results in the generation of read write sequence patterns with the minimum support of 4.

Definition 9 (Prefix and postfix)**.** *Given two transactional sequences*
$a = \{Op_1(it_1), Op_2(it_2), \ldots, Op_n(it_n)\}$ *and*
$b = \{Op_1(it_1), Op_2(it_2), \ldots, Op_m(it_m)\}$ $(m < n)$,
sequence b is called a prefix of sequence a if and only if $Op_i(it_i) = Op_{i'}(it_{i'})$ $\forall$ $i \in$ {*1, 2, . . . ,m*}*. Then*
$\{p_{m+1}(it_{m+1}), \ldots, Op_n(it_n)\}$ *is the postfix of sequence a w. r. t. b if b is the prefix of a.*

Definition 10 (Projection) Sequences a and b are two transactional sequences such that

b is a subsequence of a. The projection of a onto its prefix b is sequence a‘ if and only if a‘ is the longest subsequence of a.
The patterns obtained are converted and stored as a set of read and write rules for classification of the incoming queries.
When all the frequent patterns are kept for rule generation, the issue of data redundancy arises. On the other hand, if the maximal patterns are kept where a sequence is considered to be maximal if all of its super sequences are infrequent , then it eliminates data redundancy to an extent by removing overlapping frequent patterns, however inducing bias in the system.
To solve the above-mentioned problems, we make a compromise between the two methods by retaining the frequent patterns having a length of at least three. In Sequence Pattern Generator, we assign the length of frequent sequential patterns a value of three because the patterns of length less than three carry redundant information [76].
Some frequent patterns with lengths longer than three were observed for carrying duplicate information, however, since we see patterns of length three frequently in practical scenarios, we chose to keep the said patterns.
Cardinality of Table 4.2 displays the number of frequent sequential patterns mined using Table 4.1. Sequences follow the operation of IPPS (Iteratively Pruned Prefix Span) Algorithm. Patterns with support value higher than the specified minimum support are added to the set of Sequential Patterns.

4.3.1.5 Rule generator

The algorithm 1 generates data dependency rules after the formation of frequent sequences. These sequences are transformed to the Read and Write rules. Read rules are of the form

$$< R(it_1), R(it_2), ..., R(it_n) > \rightarrow Op(x)$$

which provide all the attributes that are read before any operation is performed on attribute it_i. Write rules are of the form

$$Op(x) \rightarrow < W(it_1), W(it_2), .., W(it_n) >$$

which provide all the attributes that are updated after any operation is performed on attribute it_i. If the confidence of any rule generated from the sequence is greater than provided minimum confidence then the rule is added to the rules set.

Sequential Pattern	**Data Dependency Rules**	**Confidence**
r7,r6,r1	r7,r6 → r1	100%
r7,r6,w5	r7,w5 → r6	100%
r6,w5,w4	w5,w4 → r6	100%
r6,w5,w4	w4 → w5,r6	83%
r7,r6,r1	r1 → r7,r6	80%
r7,r6,w4	r7,r6 → w4	75%
r7,r6,w5	r7,r6 → w5	75%
r7,r6,w4	r6,w4 → r7	60%
r7,r6,w5	r6,w5 → r7	60%

Table 4.3: Read and Write Dependency Rules

Cardinality of Table 4.3. displays the number of data dependency rules generated in the set ∈ {Read, Write} corresponding to the sequential pattern generated from Table 4.2. Value of confidence for each rule is shown which is maintained above the specified minimum confidence. The Dependency Rule Miner Algorithm takes as input non-malicious queries fetched from the transactional logs as input and returns a set of read and write rules within a single dictionary.

Algorithm 1 Dependency rule miner.

```
Data   : DB : Initial Database, min_sup : Minimum support, min_conf: minimum
         confidence, min_len: minimum length of sequential patterns
Result : R: Set of Read and Write rules
begin
    Initialisation
    Dictionary D = Ø;
    Seq_Patt ← π;
    pos ← -1;
    Extract Frequent Sequential Patterns
    for frequent sequential patterns π ∈ Seq_Patt do
        IPPS(<>, (i,pos))
    end
    Procedure IPPS (β, DB|β)
    if length (β) > min_len then
        Seq_Patt ← append β
    end
    foreach sequence s with pos in DB|β do
        foreach item Ψ in s from pos to length (s) do
            if occurrence(Ψ) == Ø or occurrence(Ψ) != start then
                D ← generate and store all projections from Ψ
            end
        end
    end
    for projected sequences γ from D in DB|β do
        if length(γ) >= min_sup then
            IPPS (β + γ, DB|γ)
        end
    end
    Extract Dependency Rules
    for frequent sequential patterns π in Seq_Patt do
        foreach rule ∈ Rule_Generator(π) do
            if confidence(rule) >= min_conf then
                add rule to R
            end
        end
    end
end
```

Dependency Rule Miner Algorithm is then used to generate prefix span rules which are later used in detection phase to classify an incoming transaction. Step 3 initializes the dictionary storing the projections generated from the formatted sequence tokens. We chose the data structure as a dictionary since it provides us with a highly efficient method to access and store the read and write rules. Step 4 to 5 initialize the sequential pattern set and the start position of projections to be used. In step 7 to 9, we make a call to the subroutine that for all the sequences it employs the Iterative Pruned Prefix Span (IPPS) algorithm with the starting position of sequence.

In step 11 to 13, we check whether the length of the projected sequential pattern is greater than minimum length threshold (min_len). If it is greater then we append the projected pattern to the set of sequential patterns. From steps 14-19, for each read and write sequence projected with the starting position from the projected database, we generate all the possible projections and store the projections in the dictionary.

In steps 21 to 25, we retrieve the projections stored after processing from the dictionary. In the first call of subroutine, length-1 sequences are generated which works faster as the threshold of support is applied and saves execution time by pruning recursion of other sets. Recursive call is made for the sequence having higher support threshold than certain threshold (min_sup) to produce further projection with increasing length of sequence and appending previous projection of sequence.

From steps 26 to 34, sequential patterns are transformed into a set of read and write rules with the function Rule_Generator(). All the possible read and write rules generated having confidence greater than certain threshold (min_conf) are collected and other rules are discarded.

Later these R rules are used in Algorithm 3 to compute QLAI (See definition 13 for reference) of various queries within a transaction and classify it as malicious or non-malicious.

4.3.2 EM Clustering

EMSPM uses Expectation Maximization clustering with Gaussian Mixture Models which aims to group the given transactions in database logs into various role profiles. EM Clustering, used as a gaussian mixture model clustering technique, was proposed by Dempster et al. [78]. The scope of EM algorithm's applications and its widespread applicability

are evident in the book by McLachlan et al. [83]. Neal et al. [79] introduced other variants of the EM algorithm, like "incremental", "sparse" and "winner-take-all" versions which employed joint maximization of the function by other means which in turn led to maximization of the true likelihood. EM is a distance-based algorithm generally used for estimation of density. EM algorithm with Gaussian mixture model proves to be a robust technique for clustering, especially where the data is determined to be insufficient. Statistical models of the data can be generated using the EM algorithm with minimal computational time requirement (Mitra et al. 2003) [84]. This technique is used to estimate the parameters of probabilistic models to which the algorithm attempts to fit the given data. Starting with random initial parameters, it iteratively performs two main steps. In the first step, called the expectation step, expected cluster probabilities are computed while in the second step called maximization step, distribution parameters and their likelihood over given data is computed. The above two steps are repeated until the log-likelihood that measures quality of clustering reaches termination condition which specifies the convergence criteria.
EM clustering was used due to its low computation time and high accuracy. Moreover, other advantages of the EM algorithm as explained by Ordonez et al. [85] included guaranteed increase in likelihood with each iteration, strong statistical basis, linearity in database size, ability to stay robust to noisy data and capability to handle desired number of clusters (even of high dimensionality) as input. In addition to that, it provides fast convergence, given a good initialization.

Given a dataset $\{x\}_{i=1}^{N}$ our goal is to assign every cluster to an instance. We assumed that there are N data points in the dataset and that there are k clusters. We model the index of the cluster as a random variable z = j and a multinomial distribution satisfying j=1 is used to output the probability of the index of the cluster, such that

$$\pi_j = p(z = j), \forall j, j = 1, ..., k \tag{4.2}$$

p is a Gaussian distribution, I_j the identity matrix of order j. The parameters that are to be found, i.e., the mean l_j variance P_j = diag (r1; r2; rj) and the distribution function

pj are approximated.

$$\theta = \left\{ \mu_j, \sum_j \pi_j \right\}_{j=1}^{k} \tag{4.3}$$

$$p(x|\theta) = \sum_{z=1}^{k} (p(x|z,\theta)p(z|\theta)\pi_j) \tag{4.4}$$

where z is an unknown variable. The total log likelihood of all data is given by

$$l(\theta, D) = log \prod_{i=1}^{N} \sum_{j=1}^{k} \pi_j e^{-\frac{||x_i - \mu_j||^2}{2\sigma^2}} \tag{4.5}$$

We choose parameter values that maximize the likelihood function. Here D denotes the data. Some of the unknowns are assumed to be known, to simplify the optimization, while estimating the others variables and vice versa. For each class, the conditional expectation of z = j is given by

$$w_j = p(z = j|x, \theta) = \frac{p(x|z = j, \theta)}{p(x|\theta)} = \frac{\pi_j N(x_i|\mu_j, \sum_j)}{\sum_{i=1}^{k} \pi_j N(x_i|\mu_j, \sum_j)} \tag{4.6}$$

Since each point x contributes to w_j in some proportion for particular x_i we have

$$w_{ij} = \frac{\pi_j N(x_i|\mu_j, \sum_j)}{\sum_{i=1}^{k} \pi_j N(x_i|\mu_j, \sum_j)} \tag{4.7}$$

The optimization algorithm is called EM Algorithm and it groups incoming transactions into clusters with similar transactions representing a particular role on the basis of several attributes relating to user-access patterns within transactions.

EM clustering algorithm[Algorithm 2] is used to create transaction's role profiles which are later used in detection phase to assign a role profile to an incoming transaction. In steps 3 to 5 cluster means, covariances and mixing coefficients are randomly initialized for each of the K clusters. The aim of EM clustering is to cluster the transactions into K components so as to maximize the log likelihood of the given database computed in step 6. Then E-step and M-step is performed repeatedly until the convergence criteria is satisfied.

In E-step from steps 10 to 11, membership probabilities of each cluster are computed for the given dataset. Each point in the given dataset has a membership probability associ-

ated with every cluster, and lies between the values 0 and 1. Membership probabilities are then used to maximize the log likelihood of a given dataset under the current distribution.

From steps 14-17 cluster parameters — means, covariances and mixing coefficients are recalculated using updated membership probabilities.

Algorithm 2: Expectation Maximization Clustering

Data : X : user transactions logs , K : number of clusters

Result : $\mu(k)$: means of K clusters,

$\sum(k)$: covariances of K clusters,

$\pi(k)$: mixing coefficients of K clusters

1 **begin**

2 **Initialization**

3 $\mu(k) \leftarrow$ random K cluster means

4 $\sum(k) \leftarrow$ random K cluster covariances

5 $\pi(k) \leftarrow$ random K cluster mixing coefficients

6 $L_0 \leftarrow \sum_{n=1}^{N} lnp(x_n|\pi, \mu, \sum)$

7 **Learning Phase**

8 **repeat**

9 **Expectation Step**

10 **for** $k \leftarrow$ *1 to K* **do**

11 $\gamma_k \leftarrow \frac{\pi(k)p(X|k)}{\sum_{j=1}^{K} \pi_j p(X|j)}$

12 **end**

13 **maximization Step**

14 **for** $k \leftarrow$ *1 to K* **do**

15 $\mu_k \leftarrow \frac{\sum_{n=1}^{N} \gamma_k(x_n) x_n}{\sum_{n=1}^{N} \gamma_k(x_n)}$

16 $\sum_k \leftarrow \frac{\sum_{n=1}^{N} \gamma_k(x_n)(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma_k(x_n)}$

17 $\pi_k \leftarrow \frac{1}{N} \sum_{n=1}^{N} \gamma_k(x_n)$

18 **end**

19 $L_t \leftarrow \sum_{n=1}^{N} lnp(x_n|\pi, \mu, \sum)$

20 **until**;

21 $L_t - L_{t-1} < \epsilon$

22 **end**

In step 19, updated likelihood is calculated using updated cluster parameters. The above steps are repeated until the algorithm converges i.e., improvement of likelihood in current iteration is less than the specified threshold. After convergence the final cluster parameters represent the role profiles corresponding to the given database of transactions. The algorithm terminates when the criteria in step 21 is satisfied and improvement in log

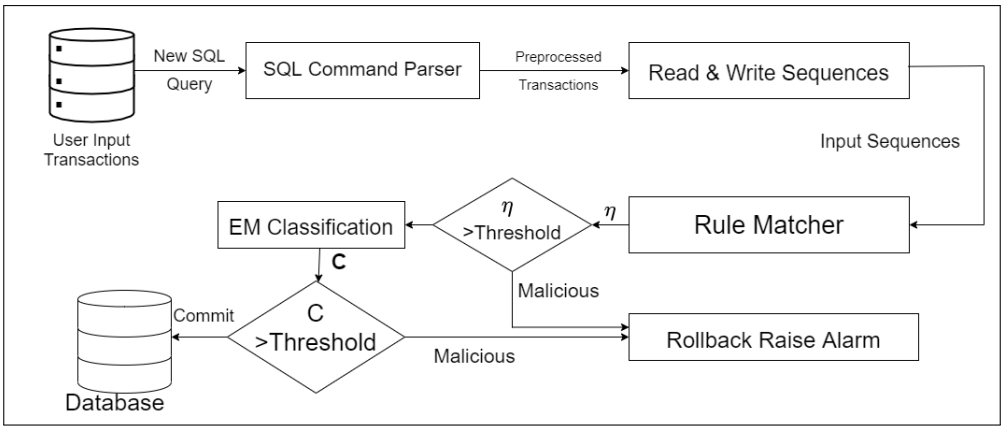


Figure 4.2: Detection Phase architecture of proposed EMSPM approach

likelihood is below terminating threshold ϵ.

4.4 Detection Phase

Once the learning phase of EMSPM is completed, we obtain a set of read and write rules as well as role profile clusters based on user access patterns. The complete detection phase architecture of EMSPM is described in Figure 4.2 which is used to classify incoming transactions as malicious or not.

The incoming SQL transaction will be interpreted and parsed against stored rules in the detection process.The degree of conformity to the mined rules, as well as the current user profile's membership in the aforementioned role profile clusters, are utilized to determine if the transaction is malicious or not. They treat compliance as the criterion for identification because in a real-world scenario there may be some divergence from the allocated/stored rules.

The detection phase consists of the following modules:

1. Expectation Maximization (EM) Algorithm
2. Rule matcher

4.4.1 Rule matcher

Firstly, new queries pass through the command parser followed by generation of read and write sequences. The Rule Matcher then compares the incoming transaction's read/write sequences to the data dependency rules defined during the learning phase for the attributes in the incoming query. In EMSPM, Rule Overlay Count and Rule Overlay Extent determines the extent to which the query sequence matches with the Rules.

Definition 11 (Rule Overlay Count) is generated by the ratio of number of rules that satisfy a query to the total number of rules generated. Given that $0 <= ROC <= 1$. We have shown the formal definition of ROC as: -

$$ROC = \frac{Matched(Q, R)}{Total(R)} \tag{4.8}$$

For example, a sequence given as, {R(a), R(b), W(a)}
These rules are matched,

$$< R(a), R(b), R(c) > \rightarrow W(a)$$

$$< R(a), R(c), R(d), R(b) > \rightarrow < R(e), W(a) >$$

Definition 12 (Rule Overlay Extent ROE) for a specific rule given for a sequence is defined as the degree of similarity with the input sequence and a set of rules. ROE for a rule and sequence is the maximum value generated by the weighted combination of Level- 0-Similarity and Level-1-Similarity in the ratio of 3:7 respectively. This ratio was found by tuning the parameters of the model, the maximum value was obtained when the weighted combination of Level-0-Similarity and Level-1-Similarity was in weight ratio 3:7. ROE can measure coherence between the rules and sequence in range [0,1] where higher value of ROE makes the rule high compliance with the query.

$$ROE = \frac{3 * L0S + 7 * L1S}{10} \tag{4.9}$$

For example, we show the case for the rule generated $\{R(a), R(b), R(c)\} \rightarrow W(a)$, and a sequence
{R(a), R(b), W(a)}

Level-0-Similarity L0S: The count of identical attributes between the rule and the sequence. We formally show the definition of L0S as

$$L0S = \frac{CountofIdenticalAttributes}{TotalAttributes} \tag{4.10}$$

For the above example,

$$L0S = \frac{2^2}{3^2} = \frac{4}{9}$$

Level-1-Similarity L1S: Ratio of the length of longest common subsequence multiplied by 2 to the total length of the rule and sequence. We have shown the formal definition of L1S.

$$L1S = \frac{2 * LCS(R, Seq)}{(len(R) + len(Seq))} \tag{4.11}$$

For the above example,

$$L1S = \frac{2.3}{4+3} = \frac{6}{7}$$

So, for this example, the ROE will be calculated as follows:

$$ROE = \frac{3 * \frac{4}{9} + 7 * \frac{6}{7}}{10} = \frac{11}{15}$$

Definition 13 (Query Log Affinity Index QLAI) QLAI is defined as the benchmark for determining whether the incoming transaction is malicious or not. QLAI takes into account the metrics, Rule Overlay Count (ROC) and Rule Overlay Extent (ROE). We formally show the definition of QLAI as:-

$$QLAI = \alpha.ROC + \beta.ROE \tag{4.12}$$

where α, β are weights assigned to the respective metrics.

Algorithm 3 Detection phase.

```
Data   : R: Rules Store
         RP: Role Profiles
Input  : TRN : Transaction for New Queries
         η : Malicious Rule Threshold
         (α, β) : Parameters for QLAI
Result : Classification of the new Query as malicious or not
begin
    Initialization Dictionary Attr ← ∅
    AttrCount ← ∅
    AttrTotal ← ∅
    Match ← ∅
    ROC ← ∅
    ROE ← ∅
    L0S ← { }
    L1S ← { }
    lcs ← ∅
    During Transaction
    foreach Query q in TRN do
        parseTree ← SQLParser(q)
        Sq ← SequenceGenerator(parseTree)
        foreach Rule Set rule in R do
            if f rule is matching Sq then then
                Match ← Match+1
            end
        end
        ROC ← Match / TotalRules(R)
        foreach Rule Set rule in R do
            foreach or each operation Op(a_i) in rule do
                if Attr has not a_i then
                    AttrTotal ← AttrTotal + 1
                    Attr ← update(a_i)
                end
            end
            foreach or each operation Op(e_i) in Sql do
                if Attr has e_i then
                    AttrCount ← AttrCount +1
                    Attr ← remove(e_i)
                end
            end
            L0S ← append ( (AttrCount)2 / (AttrTotal)2)
            lcs ← LCS(rule,Sq)
            L1S ← append (2*lcs / (len(rule) + len(Sq)))
        end
        Sq_ROE ← (3*L0S + 7*L1S) / 10
        ROE ← maximum(Sq_ROE)
        QLAI ← α * ROC + β * ROE
        if QLAI > η then
            Rollback – Raise Alarm
        end
    end
    C ← classify(TRN)
    if (C == malicious) then
        Rollback – Raise Alarm
    end
    else
        commit
    end
end
```

Steps 2 to 10 of the Detection Phase Algorithm (Algorithm 3) initializes the dictionary to store the data items required to match in the rule matching phase, AttrCount to count the unique attributes in a sequence obtained from the new query. AttrTotal gives us the total attributes inside a given rule. Match gives us the total number of rules matching a given sequence. ROC, ROE, LCS are initialized empty with L0S and L1S as empty lists. L0S and L1S are taken as list data types as we are appending the values and maximum of the values are taken into account.

In step 12, we use each of the queries present in the given transaction. Step 13 to 14, gives us the parse tree obtained through function SQLParser(). Generated parse tree is transformed into formatted sequence tokens using the function SequenceGenerator(). Steps 15 to 20 are responsible for calculating the number of matching rules with the sequence for each rule in the rule set. ROC metric is determined using the matched value.

Steps 21 to 27 use the dictionary to store the number of attributes that are new in the operation of read or write. Count of AttrTotal determines the total unique attributes in the present rule. In steps 28 to 33, a dictionary is used to retrieve the count for the data items present in the sequence which are either present in the rule or not.

In step 34, the value for L0S (Level-0-Similarity) is determined using the metrics AttrCount and AttrTotal.

Steps 35 to 37 use the longest common subsequence algorithm to determine the L1S metric. L1S is defined with the metrics of length of rule, length of the sequence and the lcs value. In steps 39 and 40, we determine the ROE (Rule Overlay Extent) as the maximum value obtained from the weighted combination of the two metrics, L0S (Level-0-Similarity) and L1S (Level-1-Similarity) with appropriate ratios. In step 41 we determine the extent of rule matching with the value of QLAI (Query Log Affinity Index) which takes the metrics of ROC (Rule Overlay Count) and ROE (Rule Overlay Extent).

From steps 42 to 45, QLAI is sent to the detection engine for determining whether the query is malicious or not. If the query is malicious then, we raise the alarm and rollback the transaction. Finally in steps 46 to 53, we classify the given transaction through EM Classifier to determine whether the query is malicious or not. If the query is malicious then, we raise the alarm and rollback the transaction else we commit the given transaction to the database.

4.4.2 EM Classification Algorithm

Incoming transactions which are determined to be malicious by IPPS rules are rolled back without committing to the database whereas those which satisfy data dependencies are further inspected to satisfy user-access patterns by using EM classification module. The algorithm iterates over all the clusters, and computes the membership probability of the incoming transaction associated with each of the transaction profile clusters. Then it computes the maximum of such membership probabilities and following the calculation of the Cluster Deviation Index classifies the transactions as malicious or non-malicious.

Definition 14 (Cluster Deviation Index CDI) Let k be the list of membership probabilities of given transaction X from all the K clusters. Then γ_{max} can be defined as the maximum membership probability of X among all clusters and the corresponding cluster defines the role profile of user executing X. Finally, Cluster Deviation Index (CDI) is formulated as:

$$CDI = 1 - \gamma_{max} \tag{4.13}$$

$$where, \gamma_{max} = max(\gamma_k) \forall k \in [1, K] \tag{4.14}$$

Algorithm 4: EM Classification Algorithm

Data : δ : dissimilarity threshold

K : number of clusters

$\mu^{(k)}$: means of K clusters

$\sum^{(k)}$: covariances of K clusters

$\pi^{(k)}$: mixing coefficients of K clusters

Input : X : transaction record

Output: C : Class (Malicious or Non-malicious)

1 **begin**

2 **Classification Phase**

3 **for** $k \leftarrow$ *1 to K* **do**

4 $\gamma_k \leftarrow \frac{\pi_k p(X/k)}{\sum_{j=1}^{k} \pi_j p(X/j)}$

5 **end**

6 $\gamma_{max} \leftarrow \max(\gamma_k)$

7 CDI $\leftarrow 1 - \gamma_{max}$

8 **if** $CDI > \delta$ **then**

9 C $\leftarrow$ Malicious

10 **end**

11 **else**

12 C $\leftarrow$ Non-malicious

13 **end**

14 **end**

In steps 3-4, using the means, covariances and mixing coefficients learnt by the EM clustering Algorithm (Algorithm 4), membership probabilities for a given transaction with each cluster is computed. In steps 6-7, the maximum of membership probabilities is used to compute Cluster Deviation Index (CDI) to measure the deviation of given transaction from all the clusters.

In steps 8-12, Class (C) is assigned to the given transaction based on CDI values obtained in step 7. Here δ is used as a dissimilarity threshold to identify malicious and nonmalicious transactions. If CDI $> \delta$, then the transaction is classified as malicious; otherwise, it is classified as non-malicious.

4.5 Implementation and performance of the proposed model

4.5.1 IDS Models for comparison

To demonstrate the efficiency of our proposed approach, several experiments were carried out on synthetically generated transactions targeted towards conventional banking databases adhering to TPC-C (Transaction Processing Performance Council 2002) benchmark [63].We created two independent modules for generating malicious and non-malicious transactions, due to the lack of available datasets that met our requirements. For the generation of malicious transactions, we created two sets of fraudulent transactions. The first set was randomly generated, simulating an attacker who is not aware of the usual database requirements. The second set was designed in a way that deviates from regular user behavior and indicates attempts to perform unauthorized actions. For the generation of normal (non-malicious) transactions, we established a set of SQL queries that adhere to the TPC-C benchmark, and used this to generate routine transactions. We also assigned specific traits to each user based on their access patterns and permissions, and constructed transactions based on those traits. To conduct the performance evaluation, 25,000 transactions were generated from the two logs. In this section, we first detail the transaction generation methodology, followed by the study of the performance of EMSPM on the synthetically generated datasets.

The keen observations were as follows -:

1. EMSPM took into account the fact that any dataset which conformed to our approach did not exist which prompted us to make effective use of synthetically generated datasets. This dataset contains two different modules for the two different types of transactions that we operated on — malicious transactions generation module and normal transactions generation module, the description and working of which has been given below.

2. To assess our approach on a wide gamut of attacks, we generated two different sets of malicious transactions. First set of transactions were generated randomly,

assuming that the attacker is unaware of normal database dependencies. This set is generated by replacing attributes randomly of legitimate queries which constitute the usual behavior and by replacing legitimate transactions of each. The second set of transactions were generated in such a way that they do not correspond to typical user activity and are indicators of users attempting to perform transactions beyond their permission levels.

3. Normal transactions are those that satisfy data dependencies and user-access patterns at the time of execution. To generate normal transactions, we define a fixed set of SQL queries which adhere to the TPC-C [63] benchmark and the schema used. Each user is assigned individual attributes that are within the scope of their access and permissions and the corresponding transactions are generated using those attributes thus satisfying the user-access patterns.

4. In the detection phase, a transaction is classified as malicious depending upon the value of the dissimilarity threshold. To select appropriate value of δ, we try to model our algorithm for different values of δ and compare the performance of our approach plotting graphs for various performance metrics like Precision, Recall, F1-score and Accuracy against the number of transactions.

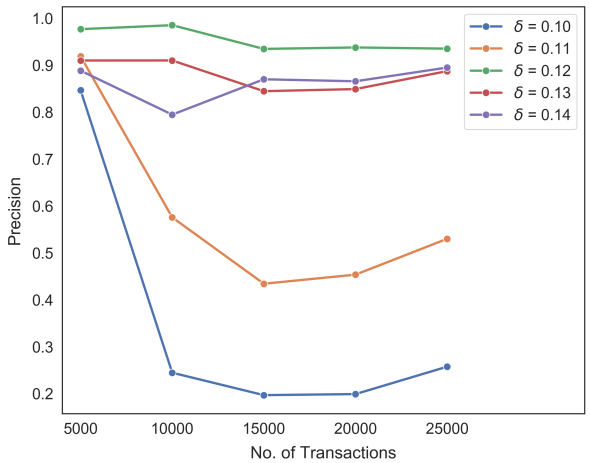


Figure 4.3: Variation in Precision with Number of Transactions for different dissimilarity threshold

δ	Number of Transactions				
	5000	10000	15000	20000	25000
0.10	0.8467	0.2448	0.197	0.1992	0.2577
0.11	0.9191	0.5758	0.4344	0.4538	0.5302
0.12	0.977	0.9855	0.9349	0.938	0.9354
0.13	0.9102	0.9102	0.8448	0.8493	0.8876
0.14	0.8886	0.7948	0.8702	0.8659	0.8951

Table 4.4: Precision for different dissimilarity threshold δ

From Figure 4.3 and Table 4.4 we can observe that for $\delta = 0.12$, the line plot for Precision remains well above all the other plots for different threshold values suggesting it to be the accurate threshold value. For δ less than 0.12, the number of false positives increases, resulting in low value of Precision. For δ greater than 0.12, the model fails to identify many true positives again resulting in low value of precision. Thus, if we consider only the Precision, $\delta = 0.12$ performs best for our given model.

In comparison of recall in Figure 4.4 and Table 4.5, though $\delta = 0.11$ comes very close to $\delta = 0.12$, but the best performance is attained by $\delta = 0.12$ when the model is evaluated over different numbers of transactions. This is because $\delta = 0.11$ fails to capture malicious transactions as effectively as $\delta = 0.12$.
Similar to the Precision and Recall, the optimum threshold for best F1-score observed from Figure 4.5 and Table 4.6 is also $\delta = 0.12$ which could be realized from the fact that F1-score is calculated from both Precision and Recall and thus the value of having equally better values of both Precision and Recall will achieve the highest F1-score. Since all the other threshold values had poor results in either Precision or Recall or both, F1-score of $\delta = 0.12$ is well above all other thresholds.

As evident in Figure 4.6, accuracy is high initially and there is a gradual decrease in accuracy for some initial number of training data transactions (approximately around 15,000). The model is overfitting to the small number of training examples. This is because the small number of training examples are not representative of the overall distribution of the data. However, as the number of training examples increases, it is observed

that the initial parameters of the model are unable to classify the new unseen data points accurately. Hence, the dip is seen around 15000 transactions. The model takes this into account and further tunes its hyperparameters to attain a better generalization of all the points which leads to a higher accuracy as the model learns more about the true distribution of the data. Accuracy of our approach from Figure 4.6 and Table 4.7 was highest in case of $\delta = 0.11$ for lesser number of transactions but as the number of transactions increased, the accuracy with $\delta = 0.11$ drops below 0.90. But for $\delta = 0.12$, the model consistently has an accuracy of over 0.93 with a maximum of 0.98373. Therefore, by comparing various performance metrics we reach to a conclusion that $\delta = 0.12$ is the optimal dissimilarity threshold to classify transactions in detection phase for our approach.

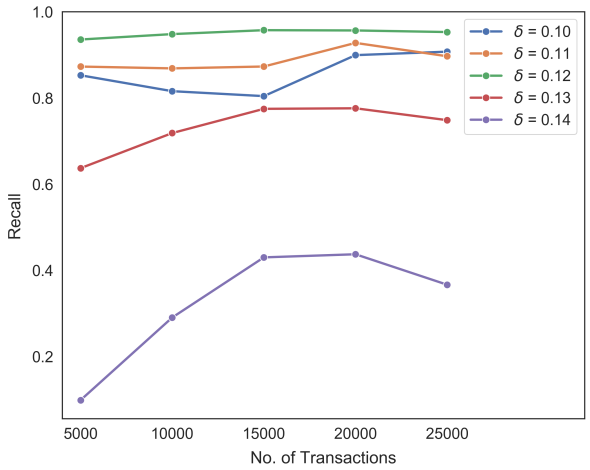


Figure 4.4: Variation in Recall with Number of Transactions for different dissimilarity threshold

δ	Number of Transactions				
	5000	10000	15000	20000	25000
0.10	0.853	0.8162	0.8047	0.8998	0.9079
0.11	0.9361	0.9487	0.9578	0.9571	0.9534
0.12	0.9361	0.9487	0.9578	0.9571	0.9534
0.13	0.6374	0.719	0.7752	0.7765	0.7489
0.14	0.099	0.2907	0.4304	0.4376	0.3669

Table 4.5: Recall for different dissimilarity threshold δ

δ	Number of Transactions				
	5000	10000	15000	20000	25000
0.10	0.7223	0.1998	0.1586	0.1793	0.2339
0.11	0.8026	0.5005	0.3794	0.4212	0.4757
0.12	0.9561	0.9667	0.9462	0.9475	0.9444
0.13	0.5802	0.6544	0.6549	0.6594	0.6647
0.14	0.088	0.2311	0.3745	0.3789	0.3284

Table 4.6: F1-Score for different dissimilarity threshold δ

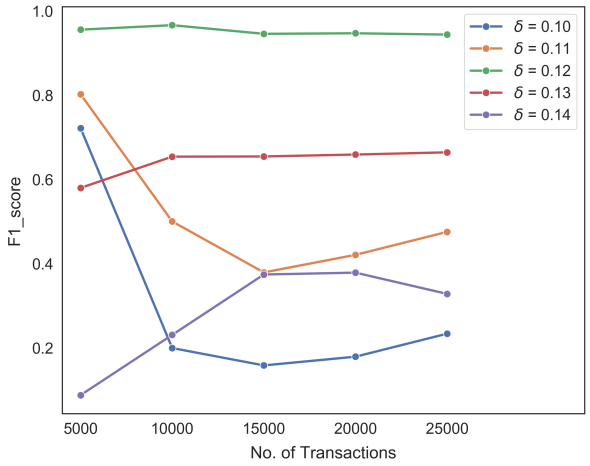


Figure 4.5: Variation in F1-Score with Number of Transactions for different dissimilarity threshold

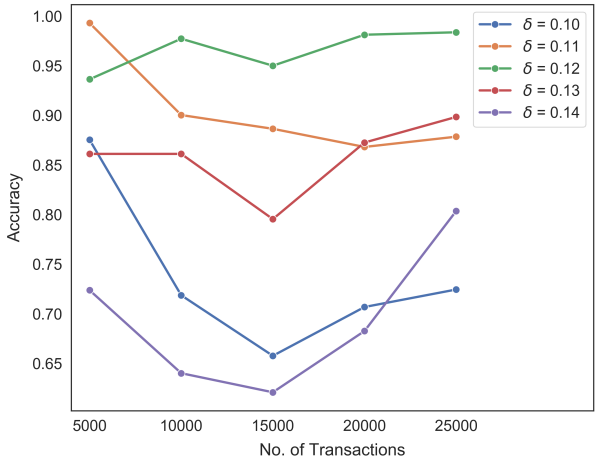


Figure 4.6: Variation in Accuracy with Number of Transactions for different dissimilarity threshold

δ	Number of Transactions				
	5000	10000	15000	20000	25000
0.10	0.8753	0.7187	0.6577	0.7069	0.7245
0.11	.9931	0.9003	0.8865	0.8682	0.8785
0.12	0.9364	0.9772	0.95	0.9813	0.9837
0.13	0.8612	0.8612	0.7954	0.8726	0.8984
0.14	0.7238	0.6401	0.6209	0.6827	0.8035

Table 4.7: Accuracy for different dissimilarity threshold δ

4.5.2 Performance evaluation of our Approach

The performance metrics used to carry out the assessment were precision, recall and F1-score. Precision is defined as the ratio of correctly identified malicious transactions known as True Positives (TP) to the total number of transactions classified as malicious from the database logs, a combination of False Positives (FP) and True Positives (TP). Recall is defined as the ratio of correctly identified malicious transactions known as True Positives (TP) to the total number of malicious transactions existing in the database logs, a combination of False Negatives (FN) and True Positives (TP). In case of imbalanced dataset, high precision and low recall as well as low precision and high recall describe a poor model. Thus F1-score is used which takes both precision and recall into account for

performance evaluation. F1-score is defined as the harmonic mean of precision and recall.

Figures 4.7,4.8,4.9 depict the variation among Precision, Recall and F1-score with the number of transactions. It can be observed from the graph that Precision first increases from 0.97698 to 0.98546 and then decreases to 0.93544 with further increase in the number of transactions.

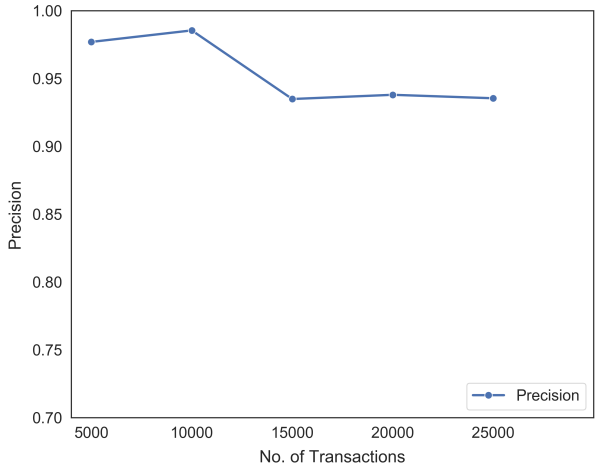


Figure 4.7: Variation in Precision, Recall and F1-score with Number of Transactions for dissimilarity threshold = 0.12

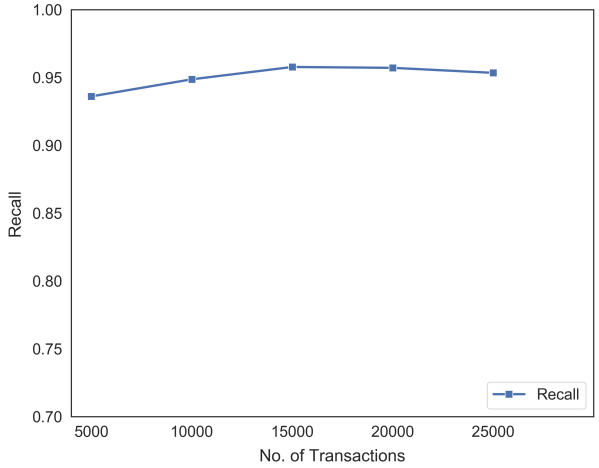


Figure 4.8: Variation in Recall with Number of Transactions for dissimilarity threshold = 0.12

It can be understood from the fact that as the number of transactions increases, the

number of rules generated for each transaction increases which also results in an increased number of False Positives. Though the value of Recall is low for a lesser number of transactions starting from 0.93608, it increases upto 0.95776 as the number of transactions are increased and then attains a stable value.

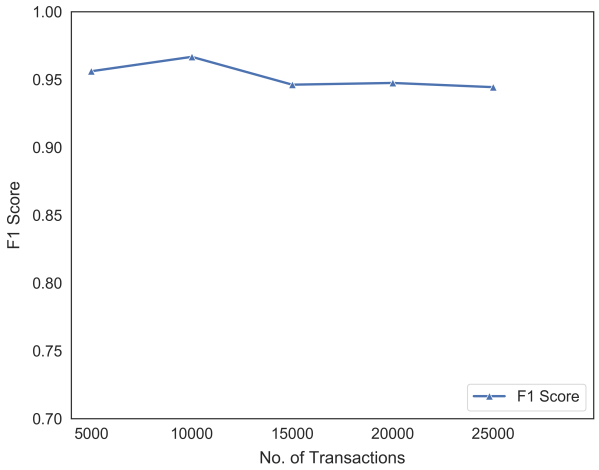


Figure 4.9: Variation in F1-score with Number of Transactions for dissimilarity threshold = 0.12

The initial low value of recall can be understood from the fact that with lesser number of transactions, the model fails to accurately identify all the malicious transactions leading to higher False Negatives, hence reducing the value of recall.

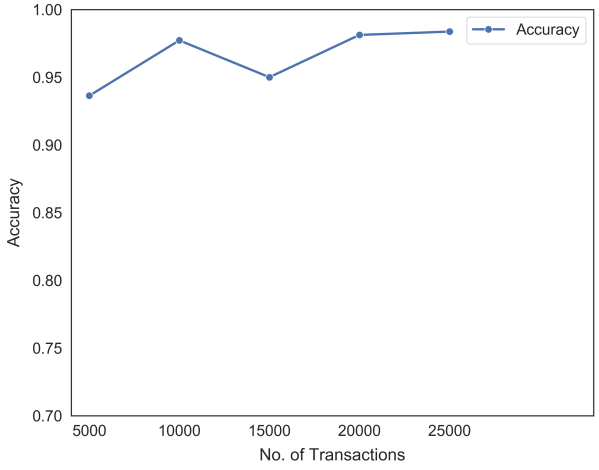


Figure 4.10: Variation in Accuracy with Number of Transactions for dissimilarity threshold = 0.12

Since F1-score is a harmonic mean of Precision and Recall, it depicts a fair balance between precision and recall and stays always between the two values maintaining a value between 0.94435 and 0.96671.

Figure 4.10 illustrates the variation between accuracy and number of transactions with a dissimilarity threshold = 0.12.

Accuracy is defined as the ratio of correctly identified transactions i.e. True Positives and True Negatives (TP+TN) to total number of transactions (TP+FP+TN+FN). Initially with a lesser number of transactions, accuracy observed was 0.93635. But as the number of transactions increased, the accuracy in- creased gradually upto 0.98373. This can be justified from the fact that with increased number of transactions resulting in larger numbers of rules and well- defined clusters, the model classified both types of transactions pretty accurately. This trend is also justified by the fact that with an increase in the number of transactions available, the model can mine increasingly consistent rules, making it more robust.

Therefore, on a complete dataset our EMSPM approach performs with an accuracy of 0.98373 for 25000 transactions.

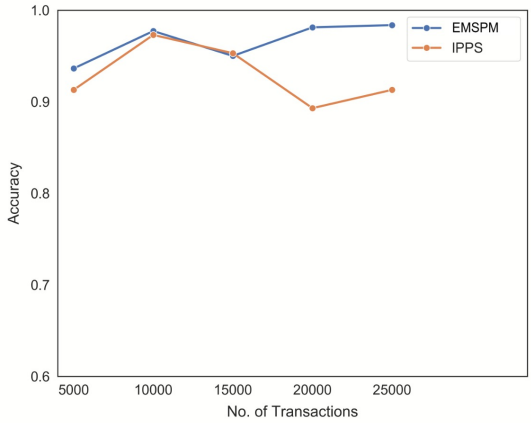


Figure 4.11: Comparison of IPPS and EMSPM for variation in Accuracy with Number of Transactions

4.5.3 Comparison of IPPS and EMSPM

Our EMSPM algorithm uses two techniques to classify an incoming transaction as malicious or non-malicious namely IPPS (Iteratively Pruned Prefix Span) algorithm and EM (Expectation Maximization) clustering.

In our EMSPM algorithm, the first step is to check whether a transaction adheres to rules generated by IPPS, if not an alarm is raised and it is classified as malicious, otherwise if IPPS rules fail to identify the transaction as malicious, then EMSPM uses EM classification algorithm to evaluate the transaction for user access patterns and detect the transaction as malicious or not.

Figure 4.11 demonstrates the effectiveness of the EMSPM algorithm over the IPPS module. It can be observed that the EM algorithm improves the classification capabilities of the IPPS algorithm significantly. If the classification is done purely with IPPS algorithms the overall accuracy for 25000 transactions is 0.913. This is due to the fact that in the IPPS algorithm, if a transaction that complies with the data dependency rules can be classified as non-malicious without checking for user access patterns that govern a particular role profile.

EMSPM combines both IPPS generated data dependency rules as well as role profiles clusters generated by the usage of an EM clustering algorithm that can determine whether the user executing the transaction adheres to the access privileges allocated or performs transactions outside the scope of their roles. This combined approach increases the accuracy of classification from 0.913 of the IPPS to 0.9837 of EMSPM for the complete dataset.

Table 4.8 contrasts various techniques based on the algorithm used, the ability to avoid intrusion, and the performance metrics - precision and recall. The performance of each technique was evaluated using different support values, and maximum precision and recall values were used. The results demonstrates that our algorithm outperforms competing methods. Since we considered the relative adherence to data dependencies, these improvements may be attributed to the low sensitivity of our algorithm to changes in user pattern. This is expressed in the transactions in the real world where the transactions

Author Names	Technique	Command Syntax	Scalability	RBAC Used	Anomaly Detection	Approach for query evaluation	Intrusion Detection Capabilities	Performance
Hu et al. [25]	Integrated dependency with sequence alignment analysis	✓	✗	✗	✗	Syntax Centric	Partial	Recall = 0.90 Precision = 0.64
Srivastava et al. [86]	Integrated dependency with sequence alignment analysis	✓	✗	✗	✗	Syntax Centric	Partial	Recall = 0.77 Precision = 0. 80
Panigrahi et al [53]	User Behavior Mining	✓	✓	✓	✗	Syntax Centric	Partial	Recall = 0.93 Precision = 0.91
Hashemi et al [87]	Temporal Mining	✓	✗	✗	✗	Syntax Centric	Yes	Recall = 0.90 Precision = 0.75
Kamra et al [31]	Dependency And Relation Analysis	✓	✓	✓	✓	Syntax Centric	Yes	Recall = 0.63 Precision = 0.77
Sohrabi et al. [42]	Mining dependencies	✓	✗	✗	✗	Data Centric	Yes	Recall = 0.65 Precision = 0.77
Doroudian et al. [61]	Mining dependencies	✓	✗	✗	✗	Syntax Centric	Yes	Recall = 0.89 Precision = 0.91
Ronao and Cho [55]	Weighted Random Forest	✓	✓	✓	✓	Syntax Centric	Yes	Recall = 0.95 Precision = 0.89
Sallam et al [48]	Anamoly Detection using Bayesian Classifier	✓	✗	✓	✓	Data and Syntax Centric	Yes	Recall = 0.97 Precision = 0.90
Seok-Jun et al. [88]	Convolutional Neural-based Learning Classifier	✓	✓	✓	✗	Data Centric	Yes	Recall = 0.93 Precision = 0.92
Sharmila Subudhi et al. [32]	OPTICS Clustering with Ensemble Learning	✓	✗	✗	✓	Data Centric	Yes	Recall = 0.92 Precision = 0.95
EMSPM	Mining Dependencies and transaction role profiling with EM Clustering	✓	✓	✓	✓	Data and Syntax Centric	Yes	Recall = 0.95 Precision = 0.97

Table 4.8: Performance comparison of EMSPM with other techniques

never fully comply with the data dependencies. Our algorithm combines the advantage of statistical-based detection and anomaly-based detection methods, allowing us to reduce both the false positive and false negative errors.

4.6 Summary of the chapter

In this chapter, we presented a DIDS which can safeguard a database from insider as well as external threats, and in general prevent it from attacks by users that are unacquainted with the normal data dependencies between the data items and the intricate syntactic features of legitimate transactions. Our intrusion detection system incorporates a Rule Mining module and an Expectation Maximization (EM) Clustering module. The Rule Mining module mines user information access patterns using modified PrefixSpan and the

Expectation Maximization (EM) Clustering module creates unique profiles of intrusion-free transactions by clustering the user activity parameters from information logs. The incoming transactions are assessed against the two levels and the extent of conformity to the mined rules together with membership of the present user profile within the role profile clusters work to classify the transaction as either malicious or non-malicious.

Considering only those frequent patterns whose lengths were at least three to remove redundant information resulted in a significant decrease in the number of false positives and increase in the overall performance. Furthermore, the use of EM clustering, which is widely popular due to its low computation time and high accuracy increased the overall efficiency of our approach. So, the use of EM clustering makes the model not only time efficient but also cost efficient. The cost efficiency makes the model accessible to be open sourced and also available to all strata requiring this intrusion detection technique.

Chapter 5

Frequent Pattern Mining and Metaheuristic hybrid Clustering

In the previous chapter a novel approach towards database intrusion detection systems (DIDS) based on Expectation Maximization Clustering and Sequential Pattern was introduced, whereas in this chapter, we will focus on the combination of BIDE and modified Particle Swarm Optimization Clustering. With growing dependence of organisations on database systems for management of information, there is huge level of risk associated in cases of security breaches leading to database being compromised. The amount of business data collected by all organisations globally is thought to double in 1.2 years [89]. Hence Intrusion Detection Systems (IDS) is supposed to meet the three basic criteria of data security which are Confidentiality, Integrity and Availability collectively known as CIA triad [11].

In this chapter, the objective is to detect and prevent all types of malicious attacks on the database. A majority of intrusion detection systems can identify only external attacks, while in this chapter our 2 approaches manages to prevent privilege abuse along with external attacks.In first approach, we proposed a BIDE and Particle Swarm Optimization clustering based malicious query detection (BPSOMQD), combining association rules and cluster analysis. In this approach, data dependency rules are generated by using BIDE algorithm to mine user transactions database containing legitimate user transactions. Cluster analysis is done to obtain role profiles by means of modified Particle Swarm Optimization algorithm to detect anomalous user patterns. All incoming transactions are validated through this detection mechanism. An alarm is generated if the transaction is found to be malicious, terminating the transaction for ensuring protection of sensitive information residing in the database.

In the second approach, we proposed a Frequent Sequential Pattern mining and a modified metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm (FPGWWO). It determines malicious transactions in RBAC and non RBAC supervised databases. The proposed method uses CM-SPADE mining algorithm for extracting data dependency rules to detect outsider threats. The modified metaheuristic clustering is then used to detect insider threats by assigning role profiles to the users based on prior user activities. By matching the user's role profile and comparing the adherence of the transaction pattern to the generated dependency rules, thus identifying whether incoming transactions are malicious.

The chapter is organized as follows. Section 5.1 illustrates the proposed BPSOMQD model as well as the architecture and algorithm of our proposed approach. In Section 5.2, the architecture and algorithm of the proposed FPGWWO model is illustrated. Finally, the conclusion of the proposed approaches is drawn in Section 5.3.

5.1 First Approach: Proposed BPSOMQD Model

In this chapter, two-phase DIDS approach is proposed that incorporates association rule mining and role profile clustering in sequential phases in the training phase to generate association rules and role profiles respectively. The main objective behind this approach is to use these two techniques in conjunction to effectively capture the intrusive activities and produce noteworthy results by combining the effectiveness of signature as well as anomaly-based intrusion detection system.

The two phases incorporated in this approach are the learning phase and detection phase. The learning phase begins with extracting transactions from transaction logs and preprocessing them into desired format to apply frequent closed sequential pattern mining algorithm and build association rules for detection phase using BIDE algorithm [90]. Association rules satisfying minimum length and confidence values are stored to be later used to detect whether transactions are malicious. In second part of learning phase,

database logs are used to generate role profiles pertaining to existing users of the organisations using the modified Particle Swarm Optimization algorithm. In this phase, it is assumed that only legitimate transaction and database logs are considered in this phase.

Secondly, in detection phase, all incoming transactions are processed and compared with the rules generated in the learning phase to calculate the Multilevel Rule Similarity Score(MRSS) of the transaction with existing rules. This MRSS score along with Cluster Similarity Index to role profiles is used to conclude whether the transaction is malicious.

5.1.1 System Architecture

In this approach the main aim is to devise a DIDS which is decoupled from the existing database management system so as to function independently in various environments without affecting the internal structure of the system. It aims to identify intrusion attacks and prevent execution of malicious transactions which could compromise the integrity and security of database.

5.1.2 Learning Phase

Figure 5.1 depicts the total learning phase of the proposed approach. In this phase, transaction logs are utilized to extract data dependency rules which are used to calculate Multilevel Rule Similarity Score in detection phase. Apart from this, Database logs are used to capture role profiles of existing authorised users capable of performing transactions in order to identify deviation from normal user behaviour. In the detection phase, data dependency rules and role profiles are collectively used to determine whether the incoming transaction is malicious.

5.1.2.1 Transaction Pre-processor

Transaction pre-processing is an important part of learning phase which transforms all the transactions stored in the audit logs into suitable format which can later be fed into subsequent modules of learning phase for generating association rules and role profiles.

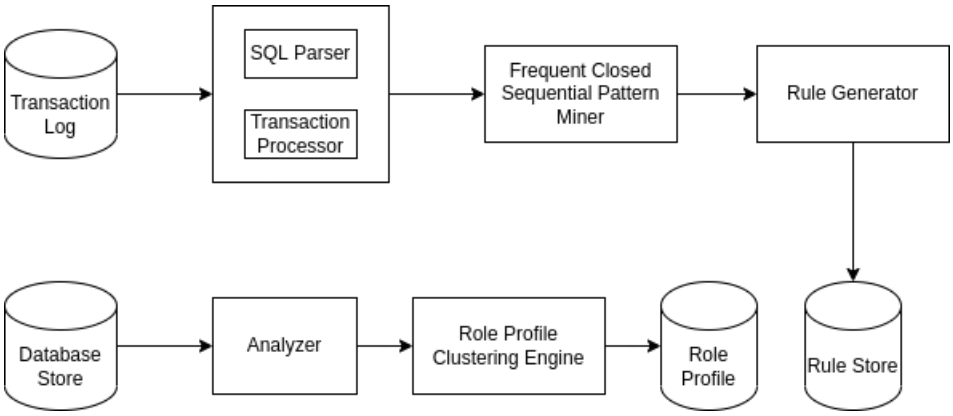


Figure 5.1: Learning Phase architecture of proposed approach

Definition 1. Query (Q) - Each Query Q is interpreted as a sequence of read and write operations on item sets present in the database.

Definition 2. Frequent Sequential Pattern (FSP): - A pattern which appears in at least minsup number of sequences present in the database is called as frequent sequential pattern.

Definition 3. Frequent Closed Sequential Pattern (FCSP): - A frequent sequential pattern that is not featured in another sequential pattern with the same support is called as frequent closed sequential pattern.

5.1.2.2 Frequent Closed Sequential Rule Miner

In this module, we have used BIDE algorithm [90] to mine frequent closed sequential patterns from existing transaction logs to generate data dependency rules.

Table 5.1 describes a sample set of sequences to understand the functioning of frequent closed sequential rule miner module.

Post the generation of frequent closed sequential patterns, those sequences are transformed into read and write rules among which those satisfying the minimum length, minimum support and minimum confidence criteria are added to the final rules set.

Table 5.2 displays the association rules generated from sequences listed in Table 5.1.

Algorithm 1 generates data dependency rules using frequent closed sequential patterns obtained from BIDE algorithm. These sets of data dependency rules are later used in

Algorithm 1: Frequent Closed Sequential Rule Miner

Data : DB: Initial Database,
min_sup: minimum support,
min_conf: minimum confidence,
min_len: minimum length of sequential pattern

Result : R: Rule Set

```
1  begin
2      Initialization
3      Sequential Pattern Base: π ← ∅
4      Rule Set: R ← ∅
5      Extract Frequent Sequential Patters
6      for sequences in DB from i ← 1 to n do
7          BFCSPG (<>, (i, −1))
8      end
9      Procedure BFCSPG (f, DB|f)
10     sup = len(DB|f)
11     if length(f >= min_len) then
12         if sup < min_sup then
13             return
14         end
15         if isClosed(DB, f, DB|f) then
16             π = π ∪ (f, sup)
17         end
18     end
19     for next_item, rem_DB in nextEntries(DB, DB|f) do
20         new_f = f ∪ next_item
21         if len(DB|f) == len(rem_DB)and(f, sup) in π then
22             π = π − (f, sup)
23         end
24         if cannot_prune(DB, new_f, rem_DB then
25             BFCSPG(new_f, rem_DB)
26         end
27     end
28     Extract Dependency Rules
29     for frequent sequential pattern fsp in π do
30         for each rule in ∈ Rule_Generator(fsp) do
31             if (confidence(rule) >= min_conf) then
32                 R = R ∪ rule
33             end
34         end
35     end
36 end
```

TID	Sequence
1	R12, W00, W20, W22, W21, R12, R23 W13, W15
2	R03, W14, W25, W24, W14, R03, R04, R04
3	R03, W11, W25, R14, W14, W04, R04
4	R12, W20, W21, R10, R23, W21, W15, R02
5	R12, R24, W20, W14, W21, W01, R23, W12, W15, R02
6	R03, W25, W14, R04, R26
7	R12, W04, W20, W22, W21, W25, R23, W20, W15, R02
8	R03, R20, W25, W14, R26, R04
9	R03, R22, W25, R03, W14, R04, R04
10	R03, R23, W25, W20, W14, W24, R04, R04

Table 5.1: Sequences for Mining Sequential Patterns

Association Rule	Support	Confidence
{R03, W25, W14} → {R04}	0.6	1.000000
{R03, W25, R04} → {W14}	0.6	1.000000
{R03, W14, R04} → {W25}	0.6	1.000000
{R04, W25, W14} → {R03}	0.6	1.000000
{R03, W25} → {R04, W14}	0.6	1.000000
{R03, W14} → {R04, W25}	0.6	1.000000
{R03, R04} → {W25, W14}	0.6	1.000000
{W25, W14} → {R03, R04}	0.6	1.000000
{R04, W25} → {R03, W14}	0.6	1.000000
{R04, W14} → {R03, W25}	0.6	1.000000
{R03} → {R04, W25, W14}	0.6	1.000000
{W25} → {R03, W14, R04}	0.6	0.857143
{W14} → {R03, W25, R04}	0.6	0.857143
{R04} → {R03, W25, W14}	0.6	1.000000

Table 5.2: Mined Data Dependency Rules

detection phase for identification of malicious transactions. Step 1-2 initializes sequential pattern base and rules set to be used. In step 3-5 a subroutine BIDE Frequent Closed Sequential Pattern Generator (BFCSPG) is called for all the sequences in the database. From steps 6-14, if a sequence is closed in the initial database, satisfies minimum length criteria as well as has a length of its projected database satisfying the minimum support criteria, then it is added to the sequential pattern base. In steps 15-22, the evaluating sequence is projected using the projected database of that sequence to ensure whether a longer super sequence of that sequence is closed in the database. If such a sequence is found, then the current sequence is removed from sequential pattern base and the super sequence is added to it. This method ensures that only frequent closed sequential patterns exist in the database which reduces the number of association rules generated.

Later in steps 23-29, the frequent closed sequential patterns are used to generate the data dependency rules which satisfies the minimum confidence criteria along with minimum support and minimum length criteria satisfied by the patterns. If all the criteria are satisfied, then such a rule is added to the final rule set.

5.1.2.3 Modified PSO Clustering Algorithm

PSO is a widely used evolutionary algorithm for performing clustering of data [91]. In our approach, we have created a variant of traditional PSO algorithm to cluster the data. In this variation, the algorithm primarily focuses on obtaining global best score as the clustering process commence while during termination stages the focuses shifts on the personal best scores. The objective behind this variation is to quickly allow the algorithm to converge by searching for global best score while relying on the personal best scores during the last few iterations to search for solution near the personal best centroid.

Algorithm 2 highlights the modified PSO algorithm developed in this study. Step 1-8 initializes various parameters used in modified PSO algorithm. According to line 9, the complete algorithm is run for maximum specified number of transactions. From step 10-20, the fitness of each particle is evaluated and if the fitness is greater than personal best score of current particles then the personal best score and position is updated. The particle position with maximum personal best score is then selected as global best position and global best score is set to be the maximum personal best score. For every iteration the personal best and global best values get updated. From steps 21-29 the position and velocity of every particle gets updated in accordance with steps 25 and 27. After completion of maximum number of iterations, the final cluster centroids Gpos represent the role profiles corresponding to the given database of transactions which are later used in Detection phase to measure similarity of incoming transaction with existing role profiles of database logs.

Algorithm 2: Modified PSO Clustering

Data : X: database logs,
K: Number of Clusters

Result : $C_k : Cluster\ Centroids = G\_pos$

```
begin
    Initialization
    N ← number of particles
    P ← list of personal best scores initialized −inf
    G ← global best scores initialized −inf
    T ← maximum number of iterations
    P_pos ← list of personal best scores initialized as ϕ
    G_pos ← global best scores initialized as ϕ
    x ← randomly initialised list of particles
    v ← randomly initialised velocity of particles in every dimension
    Learning Phase
    for t ← 1 to T do
        for i ← 1 to N do
            f = fitness_function(x_i)w
            if f > P_i then
                P_i = f
                P_pos_i = x_i
            end
            if P_i > G_i then
                G = P_i
                G_pos = P_pos_i
            end
        end
        for i ← 1 to N do
            for k ← 1 to K do
                G_inc = c * (1 − 1/T) * rand1 * (x_i − G_pos)
                P_inc = c * (1/T) * rand1 * (x_i − P_pos_i)
                v_ik = w * v_ik + G_inc + P_inc
            end
            x_i = x_i + v_i
        end
    end
end
```

5.1.3 Detection Phase

Detection phase utilizes the association rules and role profiles generated in the Learning Phase to identify malicious transactions and prevent such transactions from executing in the database system environment. In this phase, each query of incoming transaction is parsed, pre-processed and transformed to generated sequence of operations. These sequences of operations are then equated against rules generated in previous phase to obtain a Multilevel Rule Similarity Score. Along with that, the current user profile is compared with existing role profiles to calculate a Cluster similarity index(CSI). Finally a transaction is classified as malicious using a combination of MRSS and CSI. If it is determined that the transaction is malicious, then an alarm is raised, transaction is aborted and rolled back to restore the original state of database before commencement of the transaction. The complete detection phase architecture of the proposed approach is depicted in the Figure 5.2.

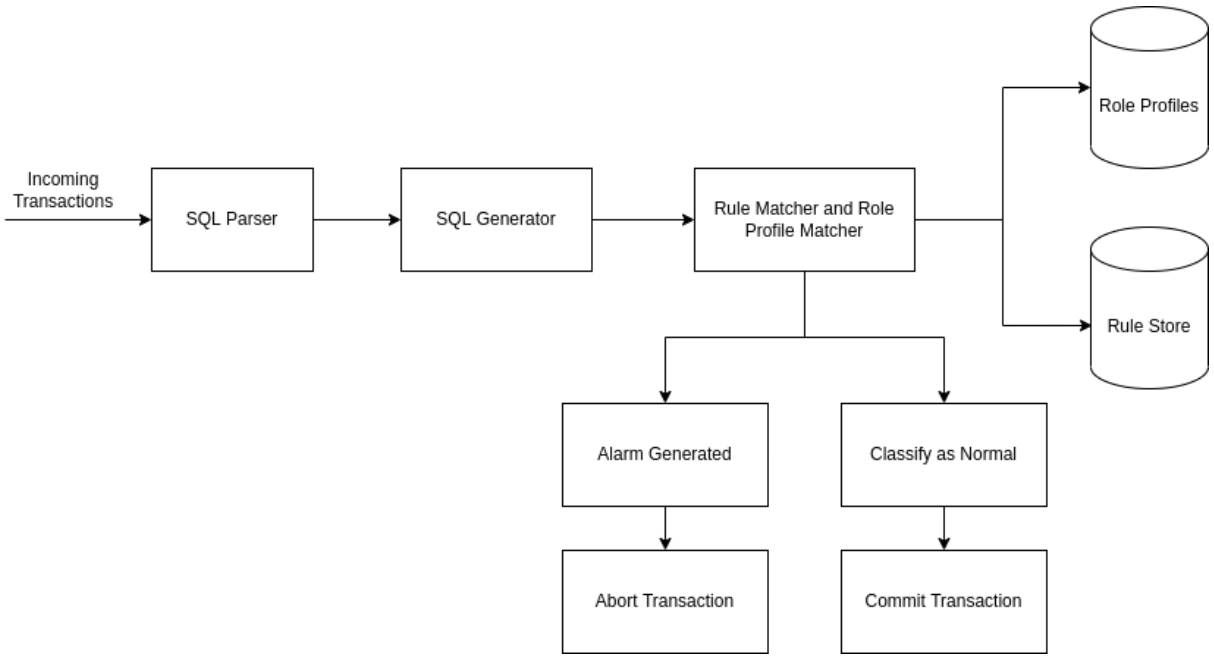


Figure 5.2: Detection phase architecture of proposed approach

5.1.3.1 Rule Matcher

In the first step of detection phase, the read and write sequences from incoming transaction queries are compared with data dependency rules generated from the frequent closed

sequential pattern miner of the learning phase.

Definition 4. Multilevel Rule Similarity Score (MRSS): -
Multilevel Rule Similarity Score for a given sequence measures the degree of similarity between the sequence from query and the data dependency rules generated from transaction logs. It is the maximum similarity score obtained from the sequence against all data dependency rules by using a weighted combination of 4 individual similarity measure namely SS1, SS2, SS3, SS4.
Where w1, w2, w3, w4 are user defined similarity level weights.

$$MRSS = \frac{w1 * SS1 + w2 * SS2 + w3 * SS3 + w4 * SS4}{w1 + w2 + w3 + w4}$$

Similarity Score 1 (SS1): It is the ratio of number of common attributes present in data dependency rule and the sequence to that of attributes present in the sequence.

$$SS1 = \frac{Count\,of\,common\,attributes}{Total\,number\,of\,attributes\,in\,sequence}$$

Similarity Score 2 (SS2): It incorporates edit distance between rule and sequence into account and is defined as

$$SS2 = 1 - \frac{edit\,distance}{max(len(rule), len(sequence))}$$

where edit distance is defined as the minimum number of insert, delete, or update operations necessary to transform sequence into rule.
Similarity Score 3 (SS3): It incorporates the length of longest common subsequence between rule and sequence into account and is defined as

$$SS3 = \frac{Length\,of\,the\,longest\,common\,subsequence}{Total\,number\,of attributes\,in\,sequence}$$

Similarity Score 4 (SS4): It is the ratio of sum of common attributes count present in data dependency rule and the sequence to product of L2 norm of vectors of data dependency rule and sequence.

$$SS4 = \frac{Sum\,of\,count\,of\,common\,attributes}{||Rule|| \times ||Sequence||}$$

where ||x|| is the L2 norm of x.

5.1.3.2 Classification Algorithm

Definition 5. Cluster Similarity Index (CSI):-
Let P be the list of membership probabilities of given trans- action X from all the K clusters. Cluster Similarity Index (CSI) is calculated as the maximum membership probability of given transaction from all the K clusters.

$$CSI = max(Pk) \forall k \in [1, K]$$

In the second step of detection phase Multilevel Rule Similarity Score of previous steps is combined with Cluster Similarity Index to give the overall prediction whether a transaction is malicious.

Algorithm 3 describes the overall procedure of detection phase. In step 1, we use each of the queries present in the given transaction to determine whether the transaction is malicious. Step 2-3 generates parse tree which is later used to produce sequence.
From Step 5-11, the generated sequence is compared with all the data dependency rules generated to compute maximum MRSS using SS1, SS2, SS3 and SS4. From Step 12-14, membership of sequence is checked with role profile clusters generated using modified PSO clustering algorithm.

In step 15-16, Cluster Similarity Index is calculated and combined with MRSS to classify the incoming transaction. If the overall score exceeds the dissimilarity threshold, then the transaction is classified as malicious, otherwise non-malicious as described in Step 17-21. Finally, in Step 22-26 we take action according to the class of transaction. When a transaction is found to be malicious, it is rolled back and an alarm is raised else it is successfully committed in the database.

5.1.4 Implementation and performance of the proposed model

To examine the performance of our proposed approach, many experiments were run on a conventional banking dataset adhering to TPC-C benchmark [63]. The entire dataset of

Algorithm 3: Detection Phase Algorithm

Data : R: Rule Set,
K: Number of Clusters,
δ : dissimilarity threshold,
T: Transaction,
w1,w2,w3,w4: similarity level weights

Result : $C : Class(malicious\ or\ non-malicious)$

begin
  During Transaction
  for *each of the Query q in TRN* **do**
    $parseTree \leftarrow SQLParser(q)$
    $Seq \leftarrow SequenceGenerator(parseTree)$
    $MRSS = 0$
    for *each rule in R* **do**
      $SS1 = similarity\_score1(rule, Seq)$
      $SS2 = similarity\_score2(rule, Seq)$
      $SS3 = similarity\_score3(rule, Seq)$
      $SS4 = similarity\_score4(rule, Seq)$
      $MRSS = max(0, \frac{w1*SS1+w2*SS2+w3*SS3+w4*SS4}{w1+w2+w3+w4})$
    end
    for $k \leftarrow 1$ *to* K **do**
      $P_k = membership(C_k, Seq)$
    end
    $CSI = max(P_k)$
    $C\_score = \frac{2\times MRSS\times CSI}{MRSS+CSI}$
    if $C\_score > \delta$ **then**
      $C \leftarrow malicious$
    end
    if $C\_score <= \delta$ **then**
      $C \leftarrow non-malicious$
    end
    if $C == malicious$ **then**
      Rollback-Raise Alarm
    end
    if $C\ != malicious$ **then**
      commit
    end
  end
end

transactions is broadly divided into two sets of transactions — one comprising of malicious transactions carried out by unauthorized users while other being normal transactions complying with data dependency rules and performed by authorized users. Using both sets, a total of 25000 transactions were generated and utilized to evaluate the performance of the proposed approach.

To measure the performance of proposed approach, the three metrics that were taken are precision, recall and F1- score.

Transaction Count	Precision	Recall	F1-score	Accuracy
1000	0.892	0.892	0.892	0.978
2000	0.877	0.907	0.892	0.977
3000	0.898	0.927	0.912	0.982
4000	0.908	0.930	0.919	0.983
5000	0.910	0.938	0.924	0.985

Table 5.3: Comparison of Precision, Recall, F1-score and Accuracy with No. of Transactions

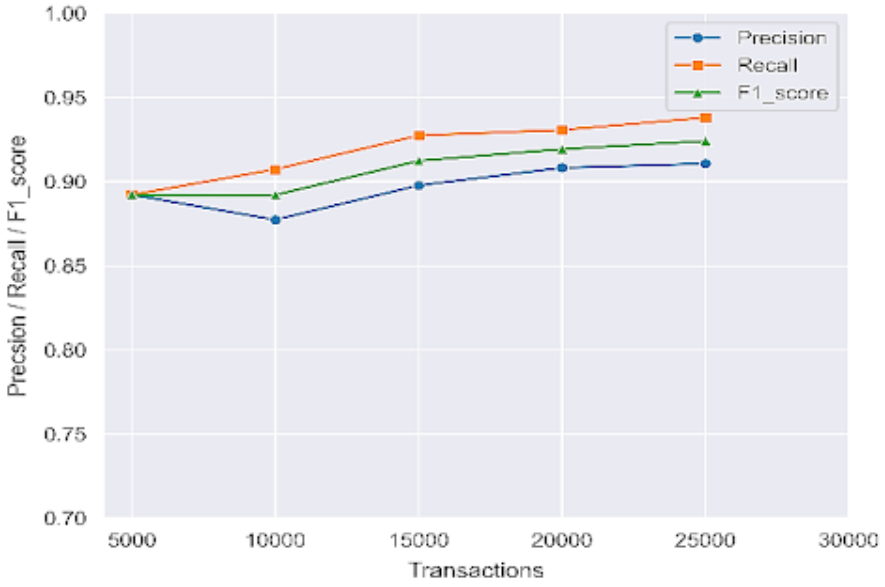


Figure 5.3: Variation of Precision, Recall, F1-Score with No. of Transactions

The comparison of Precision, Recall and F1-score with number of transactions is depicted in Figure 5.3 and Table 5.3 to highlight the effect of number of transactions on various

evaluation metrics. From the graph it can be inferred that as a general trend the results of evaluation metrics increase as the number of transactions increases which is accomplished due to increase in number of association rules and well defined user profiles obtained using modified PSO clustering algorithm. It can be seen that the value of precision range goes from 0.892 to 0.91 as the number of transactions in- crease, while that of recall ranges from 0.892 to 0.938. Since F1-score is harmonic mean of these two, its value will always lie in between the two and ranges from 0.892 to 0.924.

Figure 5.4 and Table 5.3 depicts the accuracy variation with increasing number of transactions in the database. It can be observed that as the transactions increases from 5000 to 250000, the accuracy increases from 0.978 to 0.985. With increased number of transactions, the ability of the model to correctly distinguish both malicious and normal transactions results accounts for increased accuracy.

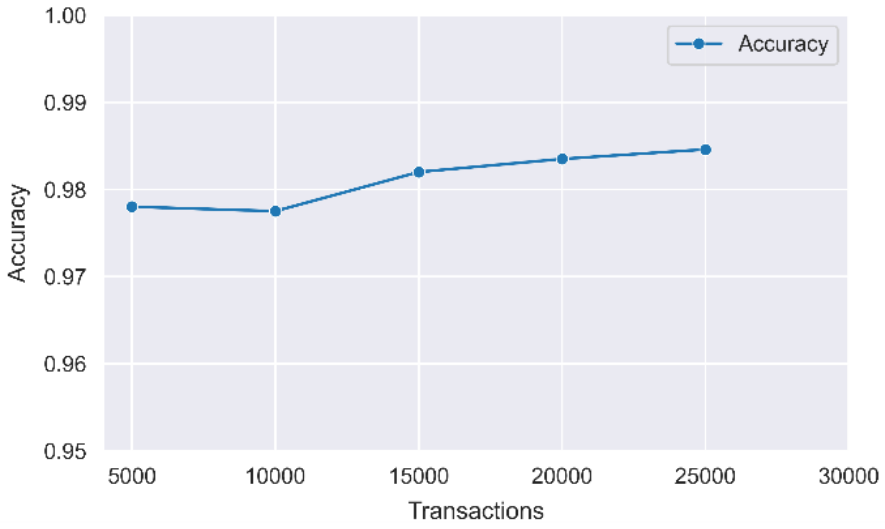


Figure 5.4: Variation of Accuracy with No. of Transactions

Table 5.4 provides a detailed comparison of our approach with various state of art approaches based on evaluation metrics like precision and recall as well. This table highlights that our approach clearly outperformed the existing approaches in terms of performance which can be attributed to the large number of association rules as well as high quality clusters generated representing user profiles using modified PSO clustering algorithm.

Reference	Technique	Command Syntax	Anomaly Detection	Intrusion Prevention Capabilities	Performance
Hu et al. [25]	Integrated dependency with sequence alignment analysis	✓	✗	Partial	Recall = 0.90 Precision = 0.64 Accuracy = 74.80%
Srivastava et al. [36]	Integrated dependency with sequence alignment analysis	✓	✗	Partial	Recall = 0.77 Precision = 0.80 Accuracy = 78.47%
Hashemi et al. [87]	Temporal Mining	✓	✗	Yes	Recall = 0.90 Precision = 0.75 Accuracy = 81.81%
Doroudian et al. [61]	Mining Dependencies	✓	✗	Yes	Recall = 0.89 Precision = 0.91 Accuracy = 89.98%
Asmaa Sallam et al. [92]	Anomaly Detection technique	✓	✓	Yes	Recall = 0.88 Precision = 0.96 Accuracy = 91.90%
Keyvanpour et al. [93]	Clustering based intrusion detection	✓	✓	Yes	Recall = 0.95 Precision = 0.87 Accuracy = 90.80%
Our Approach	BPSOMQD	✓	✓	Yes	Recall = 0.910 Precision = 0.938 Accuracy = 92.37%

Table 5.4: Performance comparison of the proposed system

5.2 Second Approach: Proposed FPGWWO model

In this section, in the initial training phase we propose a DIDS that employs rule mining and clustering followed by the classification in the detection phase. The proposed approach FPGWWO mines the read and write dependency rule subsequently followed by the modified Hybrid of Grey Wolf Optimizer(GWO) with Whale Optimisation Algorithm (mhGW-WOA) clustering using the current database logs containing transactional details. The pertinent patterns generated by the algorithm prevents undesirable rules from being created. In the first part of the learning phase, we make use of the CM-SPADE [94] algorithm to generate sequential patterns succeeded by generating data dependencies that occur from the pre-processed transaction logs. Classification rules are generated for the mined data dependencies that exceeds predefined confidence values. After analysing database logs using (mhGW-WOA) clustering algorithm, role profiles(clusters) are generated from defined parameters in the second stage of the learning phase. In this phase, only legitimate transaction and database logs are assumed to be provided.

5.2.1 System Architecture

The architecture of the proposed system (FPGWWO) consists of two modules:

- Learning Phase
- Detection and Classification Phase

5.2.2 Learning Phase

The architecture of the learning phase is represented in figure 5.5. In this phase, the rule generator uses transaction logs to extract data dependencies, and then stores them for resemblance calculations in subsequent stages. While assessing resemblance, database records are also used to analyze user access patterns, and then these patterns are grouped into role profiles. These role profile groups and generated rules are then passed to the detection phase. The rules and role clusters formed are then used in the detection phase. The learning module has two vital units:

- Rule mining
- mhGW-WOA Clustering

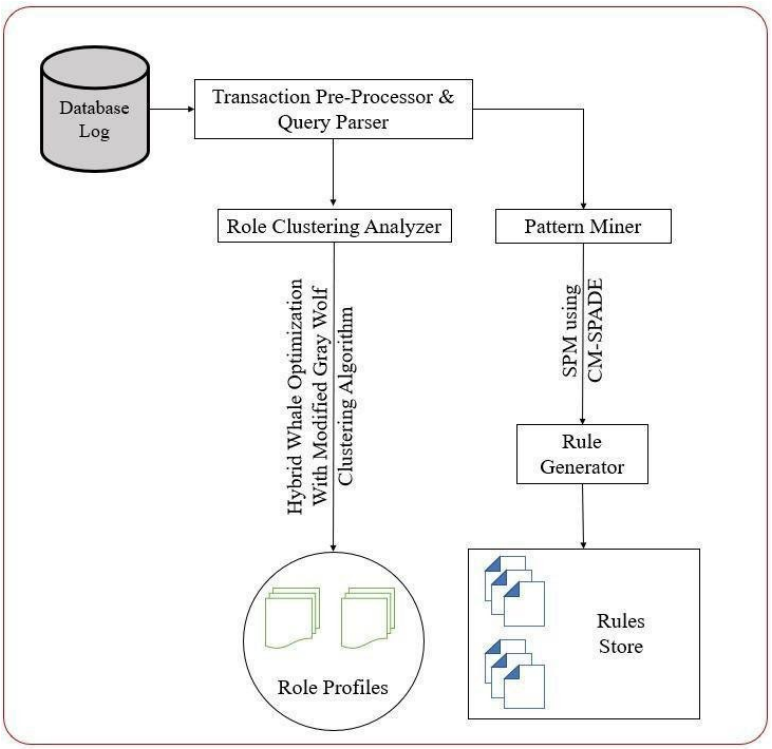


Figure 5.5: Learning Phase Architecture

5.2.2.1 Rule Mining

Rule mining refers to the discovery of patterns that indicate the basic and important characteristics of the database, then preprocess the appropriate pattern derived from the algorithm in a way that excludes the generation of undesired rules. In the first part of the learning phase, we make use of the CM-Spade[94] algorithm to generate sequential patterns succeeded by generating data dependencies that occur from the preprocessed transaction logs. The classification rules that are higher than the defined confidence threshold are generated with respect to the mined data dependencies.

Definition 6. (Sensitive elements): They are characterised as elements that are modified during the transaction or are a part of restricted access groups. An element's sensitivity is established by its weight in the transaction.

Transaction ID	Transaction
20	<r10,r15,r11,w13,r16,r12,w15,r10,r15,w11,r12,r14,r11,w14>
21	<r15,r10,w12,r10,w15,r11,r16,r13,w11>
22	<r16,r10,r15,w14,r10,w13>
23	<r11,w11,r13,r16,w12,r15,w14,r10,w13>
24	<r11,r14,w15>
25	<r10,r14,w10,r13,r14,w13>
26	<r13,w15>
27	<r15,r14,w14,r12,r13,w13,r12,w16>
28	<r14,r12,r15,w16>
29	<r14,r15,w14,r14,w13>

Table 5.5: Transactions for Mining Sequential Patterns

Table 5.5 displays the read and write items in a transaction. Each of these transactions are assigned a unique transaction ID.

Rule Generator The process of extracting relevant rules from the data mainly involves the extraction of frequent sequence patterns through successive algorithms, because the transactions have previously been processed in an ordered set of queries.

Definition 7. (Sequential Pattern Mining):
Consider SDB a sequence database and minsup as thres-hold. A sequence s is considered a sequential pattern iff supSDB(s)$\geq$ minsup.
Table 5.6 displays the number of frequent sequential patterns mined using transaction table (Table 5.5). Patterns with support value higher than the specified minimum support are added to the set of Sequential Patterns. We have taken minsup = 5.

Sequential Patterns
$r15, w14, w13$
$r16, r15, r10$
$r16, r15, w13$
$r16, r15, w14$

Table 5.6: Mined Sequential patterns

Definition 8. (Vertical Database Format): Vertical Format Sequence Database(SDB) is a database in which each entry represents an item and provides sequence list and location in which the item appears. [95]. Table 5.7 shows the vertical representations of the database of Table 5.5.

$r10$		$r11$		$r12$		$r13$		$r14$		$r15$		$r16$	
TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets
20	1,8	20	3,13	20	6,11	20		20	12	20	2,9	20	5
21	2,4	21	6	21		21	8	21		21	1	21	7
22	2,5	22		22		22		22		22	3	22	1
23	8	23	1	23		23	3	23		23	6	23	4
24		24	1	24		24		24	2	24		24	
25	1	25		25		25	4	25	2,5	25		25	
26		26		26		26	1	26		26		26	
27		27		27	4,7	27		27	2	27	1	27	
28		28		28	2	28	5	28	1	28	3	28	
29		29		29		29		29	1,4	29	2	29	

(a) Read Rules

$w10$		$w11$		$w12$		$w13$		$w14$		$w15$		$w16$	
TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets	TID	Item sets
20		20	10	20		20	4	20	14	20	7	20	
21		21	9	21	3	21		21		21	5	21	
22		22		22		22	6	22	4	22		22	
23		23	2	23	5	23	9	23	7	23		23	
24		24		24		24		24		24	3	24	
25	3	25		25		25	6	25		25		25	
26		26		26		26		26		26	2	26	
27		27		27		27	6	27	3	27		27	8
28		28		28		28		28		28		28	4
29		29		29		29	5	29	3	29		29	

(b) Write Rules

Table 5.7: Vertical Representation of Table 1

Definition 9. (i-extension): An item q is said to succeed by i-extension to an item j in a sequence $I_1, I_2, \cdots, I_n$ iff j, q $\in I_x$ for an integer x such that 1$\leq$ x $\leq$ p and q $>$ lex j.

Definition 10. (s-extension): An item q is said to succeed by s-extension to an item j in a sequence $I_1, I_2, \cdots, I_n$ iff j$\in I_n$ and q$\in I_m$ for some integers n and m such that 1 $\leq$ n$<$m $\leq$p.

Definition 11. (CMAP): A Co-occurrence MAP (CMAP) is a structure mapping each item q∈ I to a set of items succeeding it. We define two $CMAP_s$ named $CMAP_i$ and $CMAP_s$. $CMAP_i$ maps each item q to the set cmi(q) of all items j ∈ I succeeding q by i-extension in no less than minsup sequences of SDB. $CMAP_s$ maps each item q to the set $cm_s(q)$ of all items j ∈ I succeedings q by s-extension in no less than minsup sequences of SDB.

The algorithm 4 generates data dependency rules after the generation of frequent sequences which are transformed to read and write sequence. Table 5.8 shows the number of data dependency rules generated in the set ∈ Read, Write in accordance to the sequential pattern generated from Table 5.6. Confidence value for each rule is shown which is maintained above the specified minimum confidence.

In line 2 of the algorithm 4 the initial database is scanned to create a vertical representation of the database and identify the list of frequent items F. From line 3 the enumerate function is initialized. From line 3 the loop iterates over each element A_j from frequent list F. In step 6, the T_i is initialized to an empty set. From line 7 a nested for loop iterates over each element A_i from frequent list F with j greater than equal to i. Inside the nested loop pattern, A_i and A_j are merged and saved to variable R in line 8. From line 9 to line 18, the for loop iterates over the merged patterns R and checks if the pattern is either i-ext or s-ext and its support is greater than or equal to the minimum support and adds the respective pattern to T_i. In line 19 the Enumerate function is called recursively with frequent list T_i.

5.2.2.2 Role Clustering Analyzer

We use a role clustering method to detect the insider threats which would be otherwise left undetected. In our model (FPGWWO) we use a hybrid meta heuristic clustering algorithm to achieve the same. This hybrid module has two components namely Grey Wolf Optimizer (GWO) and Whale Optimizer (WO).
In algorithm 5, steps 2 and 3 consist of the following tasks: initialising search agent variables. The next five steps entail searching for the most qualified potential agent available. In step 9, the variables are updated to reflect the new values that have been

Algorithm 4: Dependency Rule Miner

Input : DB: Initial Database
minsup: Minimum Support
Result : R: Set of Sequential Patterns

```
1  Begin:
2      Scan DB to create Vertical(DB) and identify F
3      the list of frequent items
       Enumerate(F) for each pattern A_i ∈ F do
4          Output A_i.
5          T_i ← φ for each pattern A_j ∈ F, with j ≥ i do
6              R ← MergePatterns(A_i, A_j)
7              for each pattern r ∈ R do
8                  a ← last element in r
9                  if r is i-ext then
10                     if sup(CMAP_i ≥ minsup) then
11                         T_i ← T_i ∪ {R};
12                 else if r is s-ext then
13                     if sup(CMAP_s ≥ minsup then
14                         T_i ← T_i ∪ {R};
15                     else if ∄x : x ∈ cm_i(a) AND
16                     ∄x : x ∈ cm_s(a) then
17                         continue;
18                     end
19                 end
20             end
21         end
22     end
23 end
24 End
```

Sequential Pattern	Data Dependency Rule	Confidence
$r16, r15, w14$	$r15, w14 \rightarrow r16$	60%
$r16, r15, w13$	$r15, w13 \rightarrow r16$	60%
$r16, r15, w14$	$r16, r15 \rightarrow w14$	75%
$r16, r15, w13$	$r16, r15 \rightarrow w13$	75%
$r16, r15, r10$	$r10 \rightarrow r16, r15$	80%
$r15, w14, w13$	$w13 \rightarrow w14, r15$	83%
$r15, w14, w13$	$w14, w13 \rightarrow r15$	100%
$r16, r15, w14$	$r16, w14 \rightarrow r15$	100%
$r16, r15, r10$	$r16, r15 \rightarrow r10$	100%

Table 5.8: Data dependency Rules

Algorithm 5: mhGW-WOA Clustering

Input : Load Database Logs
Result : Cluster centroids

Begin:
  Initialize number of search agents
  Initialize each search agent to contain K randomly cluster centers
  while $t <$ *no. of total iterations* **do**
    for *search agent* **do**
      Calculate fitness value for the agent
      Update best search agent
    end
    for *search agent* **do**
      Update a, A, r, c, l, p, b
    end
    if $p<0.5$ **then**
      if $|a| <1$ **then**
        $\vec{D} = |\vec{C}.\vec{X^*}(t) - \vec{X}(t)|$
        $\vec{X}(t+1) = \vec{X^*}(t) - \vec{A}.\vec{D}$
      else
        $\vec{D} = |\vec{C}.\vec{X}_{rand} - \vec{X}|$
        $\vec{X}(t+1) = \vec{X}_{rand} - \vec{A}.\vec{D}$
      end
    else
      if $|a| <1$ **then**
        $\vec{D'} = |\vec{X^*}(t) - \vec{X}(t)|$
        $\vec{X}(t+1) = \vec{D'}.e^{bl}.\cos(2\pi l) + \vec{X^*}(t)$
      else
        Get best 3 positions
        x[i] = pscores[0][1] + pscores[1][1] + pscores[2][1]
        x[i]/ = 3
      end
    end
  end
End

determined. Using the Whale Optimization Algorithm, we will next update our agents at stages 10 through 20 of the process. Then, in steps 21 to 24, we use the Modified Grey Wolf Optimization Algorithm to update our agents' information. When compared to previous techniques, this step broadens the scope of the algorithms' learning capabilities. The algorithm produces the best cluster centres, which will be utilised as the basis for role profiles in the following section. Our clustering method is optimised via hybrid of the Whale Optimization Algorithm and the Grey Wolf Optimization Algorithm, shown in Figure 5.6.

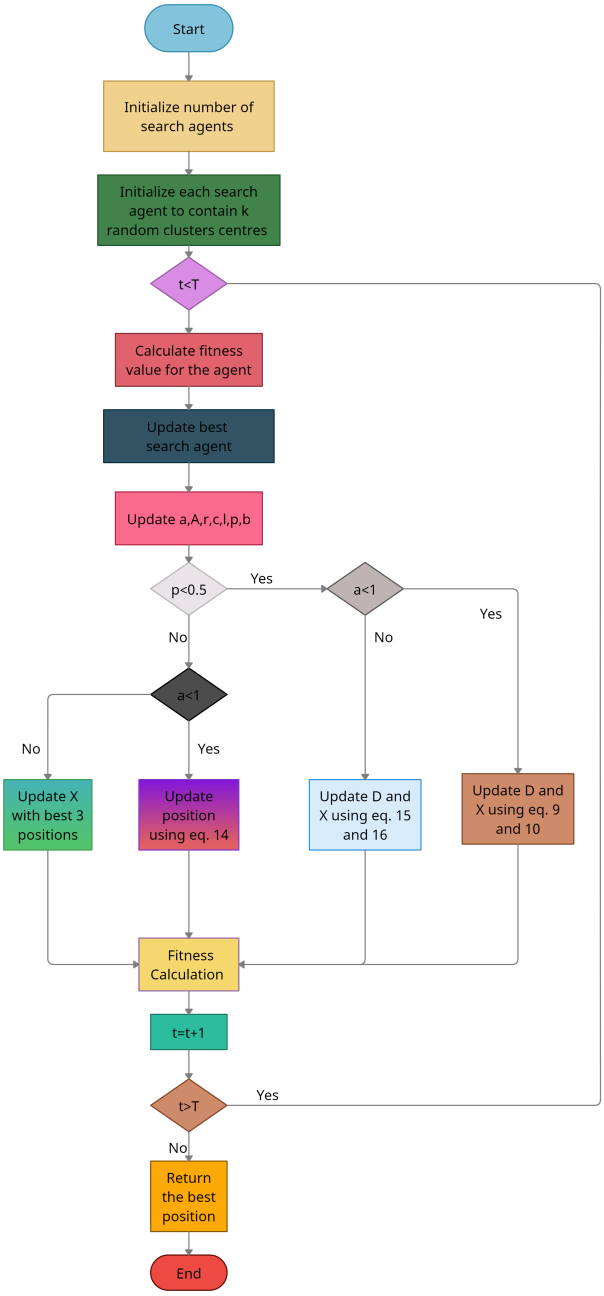


Figure 5.6: Flowchart of the mhGW-WOA clustering

5.2.3 Detection and Classification Phase

The learning phase of the algorithm provides the read rules, write rules and the role profiles. Now, these rules and role profile clusters are used to detect any anomalous transaction via the detection phase architecture. Figure 5.7 depicts the detection phase architecture as shown below.

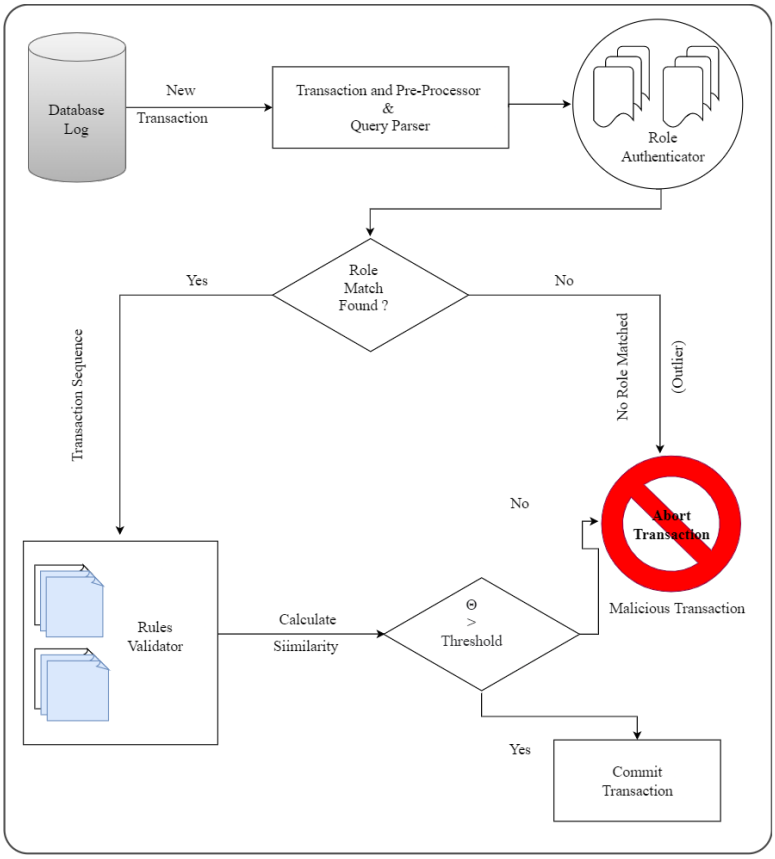


Figure 5.7: Detection Phase architecture

In this detection phase, the current user transaction is first passed through the transaction preprocessor and query parser. The anomalies in the current user transactions are then detected for each transaction by comparing them with user profiles via the role authenticator and with the sequential rules generated in the learning phase segment via the rule validator. An alarm is raised if a query is determined to be malicious. Otherwise the transaction is committed to the database. The architecture of the detection phase consists of 2 levels named Role Authenticator and Rule Validator.

5.2.3.1 Role Authenticator

The role profile clusters generated by the clustering algorithm in the learning phase are matched against the incoming transaction in the database. The role authenticator examines each query separately to evaluate if it is justified by the role of the user or not, making the intrusion detection model resistant to insider attacks as well. If the role profile of an incoming transaction is not matched to the existing profile clusters, the transaction is declared as malicious and aborted, instantaneously. However, if the role profile match is found, the transaction sequence is further checked by the rule validator for classification.

5.2.3.2 Rule Validator

The rule validator determines whether queries in a single configuration file are executed in the same order as previously validated queries. The incoming transaction query sequence is compared with read and write rules that are derived from data dependency mining using the CMSPADE algorithm in the learning phase. The data-dependent rules generated for each sensitive element in the list are compared to the run time input sequence list.Therefore, compliance with each rule is calculated. Now, we need to quantify the rule-based similarity of the two sequence datasets to draw conclusions about the credibility of the incoming transactions. This is done by calculating the similarity.

5.2.3.3 Calculating Similarity

A 100 point similarity system has been introduced to measure the extent of adherence between transactions and a set of rules mined using the rule mining algorithm. We have four grades of similarity called resemblance (R). Each grade has been allocated points. We try to contain the values of overlap measurement between 0.1 and 0.9 for minimum and maximum adherence respectively. We define:
l=0.1 and h=0.9

I Grade D resemblance: (10 Points assigned out of 100)

Grade D resemblance denoted by R4 is the measure of the number of mined rules in our database that have the same element accused in the transaction as in the rules.
Let,

a: the number of rules that follow this condition.

b: the number of rules that do not follow this condition.

$$R_4 = \left[1 + \frac{(a-1).1b}{(a+b)} * (h-1)\right] * 10$$

(Higher weight is assigned to rules that do not follow the required conditions)

II Grade C resemblance: (20 Points assigned out of 100)

This stage considers a subset of rules that belong to set 'a' in stage i.e the rules that have common elements as incoming transactions. Let,

x: denotes the number of rules that follow the same order as in transactions.

Eg : TR : R(a)R(b)W(c)

Rule: R(d)R(a)R(b)W(a)W(c)

y: denotes the number of rules that do not follow that condition.

$$R_3 = \left[1 + \frac{(x-1).2y}{(x+y)} * (h-1)\right] * 20$$

III Grade B resemblance: (30 Points assigned out of 100)

This stage considers subsets of stage 2.The set of rules is used i.e those rules that follow the same order as transactions. Let,

p : the number of rules that contain transactions as a subset i.e all common elements are together as in the transaction.

q : the number of elements that do not follow this rule.

$$R_2 = \left[1 + \frac{(p-1).25q}{(p+q)} * (h-1)\right] * 30$$

IV Intra-Transaction rules:

These rules consider a group of transactions that are executed together in a sequential manner. The algorithm used will be similar to the mentioned above. The rule transaction will be considered sequentially in the batches of k ($2 \leq K \leq 5$). Let,

s : number of rules that adhere to the transaction set in exact order.

t : number of rules that do not adhere to the transaction set.

(a) Grade A_2 resemblance (12 Points assigned out of 100)

This follows the above methods given in the above sections. The batch of size 2 is assigned a high weight and the order should look like:

k=2

$$R_{12} = \left[1 + \frac{(s-1).2t}{(s+t)} * (h-1)\right] * 12$$

(b) Grade A_3 resemblance (12 Points assigned out of 100)

The batch size of 3 will follow the same order as the above subset. It is assigned a lower weight than the A_2 resemblance. But it will have the same multiple of A_2 resemblance as the points assigned to both these resemblances are the same.

k=3

$$R_{13} = \left[1 + \frac{(s-1).1t}{(s+t)} * (h-1)\right] * 12$$

(c) Grade A_4 resemblance (6 Points assigned out of 100)

A_4 resemblance represents a batch with size 4. The order for the A_4 resemblance will be exactly half of that of A_3 resemblance as the points given to the A_4 resemblance is half of it and also the weight assigned to it is the same as that of the previous subset.

k=4

$$R_{14} = \left[1 + \frac{(s-1).1t}{(s+t)} * (h-1)\right] * 6$$

(d) Grade A_5 resemblance (6 Points assigned out of 100)

The A_5 resemblance is for the final batch of size 5. As the A_5 resemblance contains the same points as that of the previous subset, i.e. A_4 resemblance so it will have the same multiple as that of A_4. But the two equations should differ as the weights assigned to A_5 resemblance is the lowest of all the Grade A resemblances.

k=5

$$R_{15} = \left[1 + \frac{(s-t)}{(s+t)} * (h-1)\right] * 6$$

Definition 12 (Congruity Index) A Congruity Index (θ) is a similarity threshold defined as the sum of all the grades of similarity (resemblances).

$$\theta = \sum_{i=1}^{n} R_i$$

where n is the number of grades. In our case we have four grades namely A,B,C,D with four subgrades for $A(A_1, A_2, A_3, A_4)$ Therefore,

$$\theta = R_{12} + R_{13} + R_{14} + R_{15} + R_2 + R_3 + R_4$$

5.2.3.4 Detection Algorithm

In our proposed algorithm 6 we take in the generated rule set, number of clusters, similarity threshold (δ) and the transactions itself. The algorithm starts with converting the transaction into sequence using the sequence generator. Then in steps 4- 9 we calculate the role similarity of each cluster and store the highest role similarity cluster along with its similarity. We compare that similarity with that of the threshold, if it's less than the threshold we classify the transaction as malicious and raise an alarm. The respective rule is passed into the rule authenticator module if the similarity is greater than the threshold. This novel module simply calculates the different resemblances (B, C, D, A1, A2, A3, A4) from step 13- 20. We define congruity index (θ) which would be used to classify the transaction. In steps 22-25 we compare the value of θ with the δ . If *theta* is less than δ then the transaction is declared malicious and if θ is greater than δ then the transaction is declared non-malicious. Also following this in steps 27 - 30 if the transaction was malicious, the database executes a rollback along with raising an alarm and if the transaction was non malicious, the database commits all the data.

5.2.4 Implementation and performance of the proposed model

To analyze the proposed methodology (FPGWWO) performance, multiple assessments were performed on a synthetically generated banking databases adhering to the TPC-C benchmark[63]. We created two independent modules for generating malicious and non-malicious transactions, due to the lack of available datasets that met our requirements. For the generation of malicious transactions, we created two sets of fraudulent transactions. The first set was randomly generated, simulating an attacker who is not aware of the usual database requirements. The second set was designed in a way that deviates from regular

Algorithm 6: Detection Phase

Data : R: Rule Set
K: Number of Clusters
δ: Similarity Threshold
T: Transaction

Result : C:Class (malicious or non-malicious)

During Transaction:
  for *each of the Query q in TRN* **do**
    Seq ← SequenceGenerator(q)
    Max_sim ← ϕ
    for all $C_k \forall$ Cluster_Centres
    Temp ← cal_role_sim(q,C_k)
    if *Temp > Max_sim* **then**
      Max_sim ← Temp
      Max_cluster ← C_k
    End If
  end
  if($Max\_sim < \delta$)
  Raise Alarm
  for *each* ***rule*** *in R* **do**
    R_4 = GradeD_resemblance(rule, Seq)
    R_2 = GradeC_resemblance(rule, Seq)
    R_3 = GradeB_resemblance(rule, Seq)
    R_{12} = GradeA_2_resemblance(rule, Seq)
    R_{13} = GradeA_3_resemblance(rule, Seq)
    R_{14} = GradeA_4_resemblance(rule, Seq)
    R_{15} = GradeA_5_resemblance(rule, Seq)
    $\Theta = R_4 + R_3 + R_2 + R_{12} + R_{13} + R_{14} + R_{15}$
    if $\Theta < \delta$ **then**
      C← malicious
    else
      C← non-malicious
    end
    if *(C== malicious)* **then**
      Rollback – Raise Alarm
    else
      commit
    end
  end
End

user behavior and indicates attempts to perform unauthorized actions.For the generation of normal (non-malicious) transactions, we established a set of SQL queries that adhere to the TPC-C benchmark, and used this to generate routine transactions. We also assigned specific traits to each user based on their access patterns and permissions, and constructed transactions based on those traits. The performance of our systems was evaluated using these two types of database logs. 5000 transactions in total were generated.

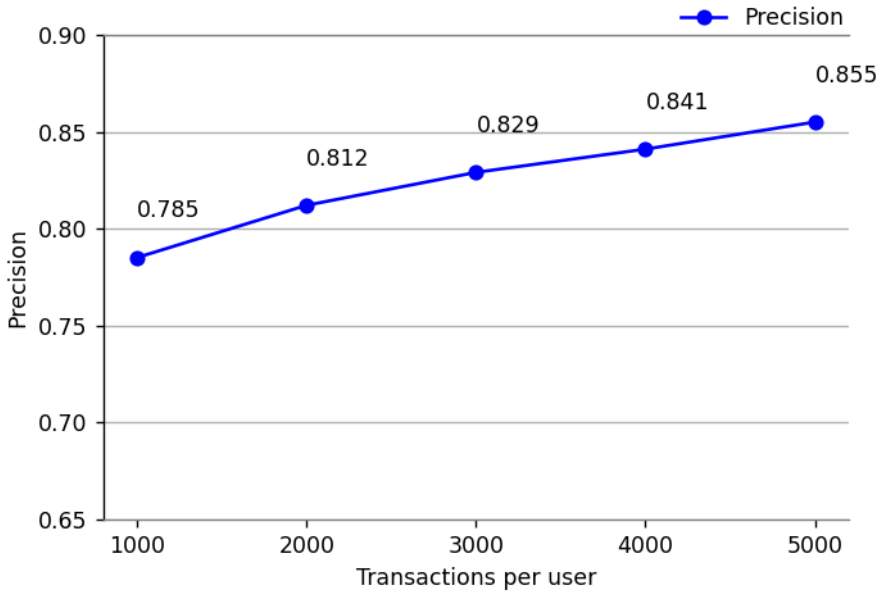


Figure 5.8: Variation of Precision with number of transactions per user at Threshold 50

The Figures 5.8-5.10 depict the variation in precision, recall and F1 Score of the system for threshold value of 50. With change in the number of transactions in the database. It can be inferred from the graph that the value of Precision first increases from 0.785 to 0.855. With an increase in the number of transactions, the value of Recall rises from 0.912 to 0.950. The number of rules created for each transaction grows as the number of transactions grows. As a result, rules will be more matched to the database, and role clustering will be more refined. As a result, the number of false negatives decreases while the number of true positives rises, resulting in an increase in the recall value. Because the F1-score is a harmonic mean of Precision and Recall, it reflects a good balance of precision and recall by continuing to rise between the two values, ranging from 0.845 to 0.882 (Table 5.9).

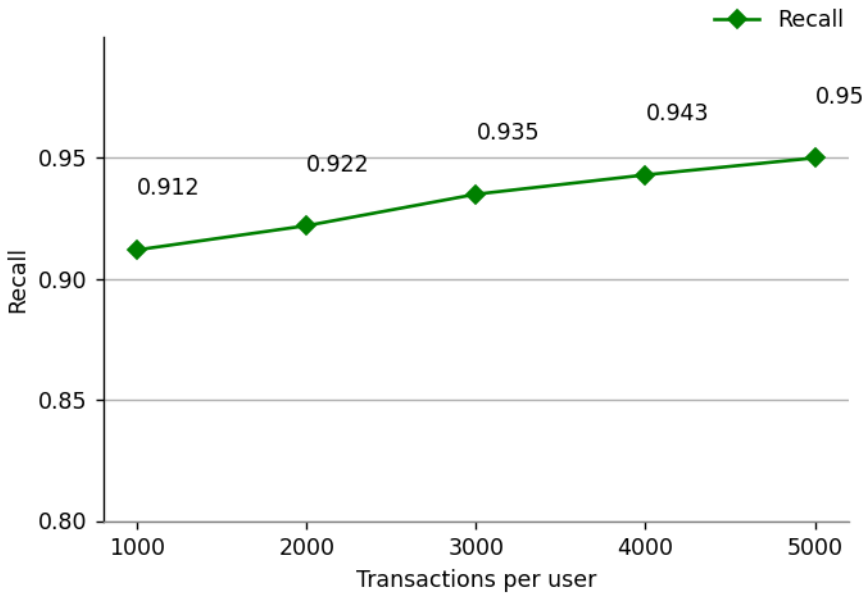


Figure 5.9: Variation of Recall with number of transactions per user at Threshold 50

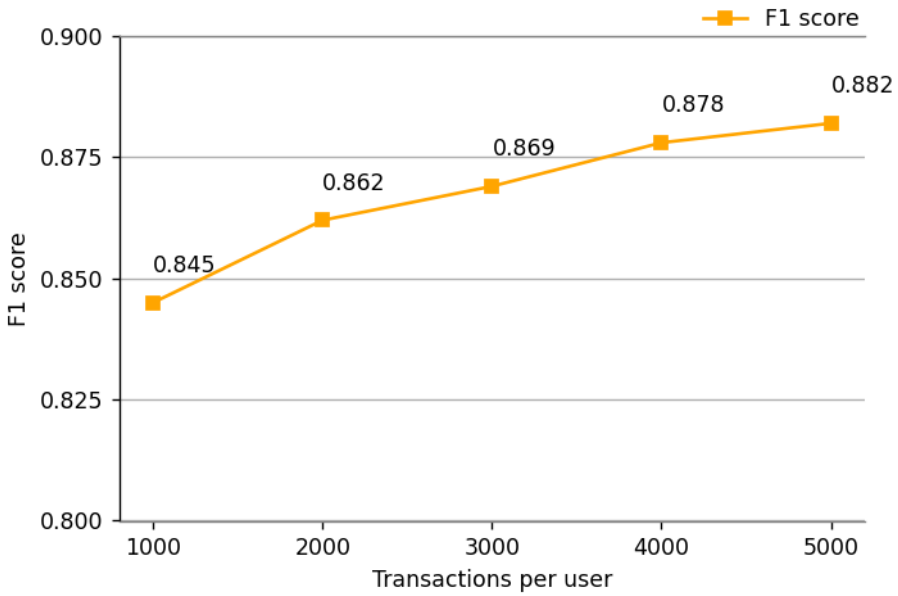


Figure 5.10: Variation of F1 score with number of transactions at Threshold 50

Threshold = 50					
Parameter	Number of Transactions				
	1000	2000	3000	4000	5000
Precision	0.785	0.812	0.829	0.841	0.855
Recall	0.912	0.922	0.935	0.943	0.95
F1 score	0.845	0.862	0.869	0.878	0.882

Table 5.9: Precision, Recall, F1-score at Threshold 50

Figures 5.11-5.13 illustrate the variation in the system's precision,recall and F1 score with the number of database transactions using threshold value of 60. As shown in the graph, the value of Precision begins to rise from 0.772 to 0.818 at the beginning of the study period. Recall's value increases from 0.921 to 0.938 as a result of increasing number of transactions. Increasing the number of transactions leads to an increase in the number of rules that are created for each transaction. As a result, the rules are more matched to the database, and the role clustering is more refined as a result of these improvements. Thus, the number of false negatives decreases and the number of true positives increases, which leads to an increase in the recall value due to the decrease in false negatives. Here also, the value of F1 Score shows a slight increase from 0.831 to 0.871.

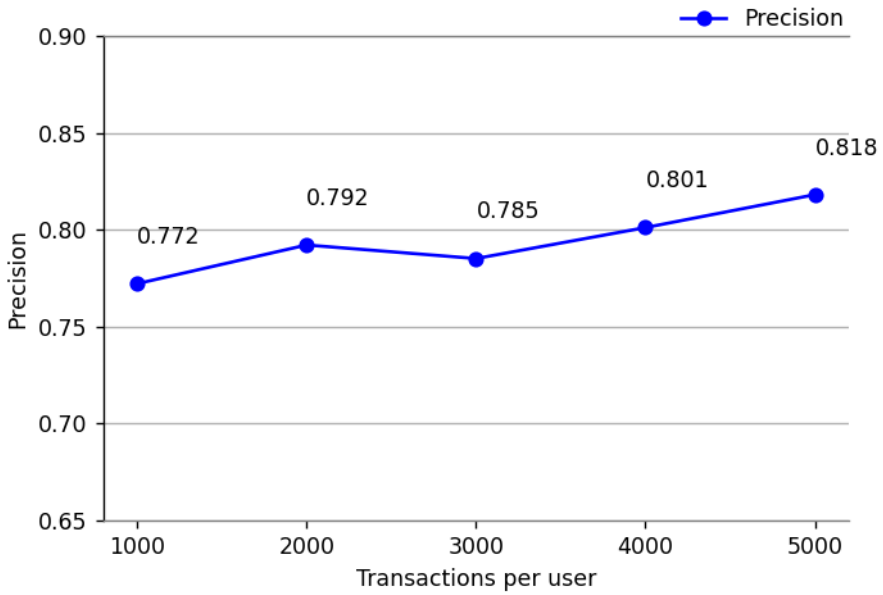


Figure 5.11: Variation of Precision with number of transactions per user at Threshold 60

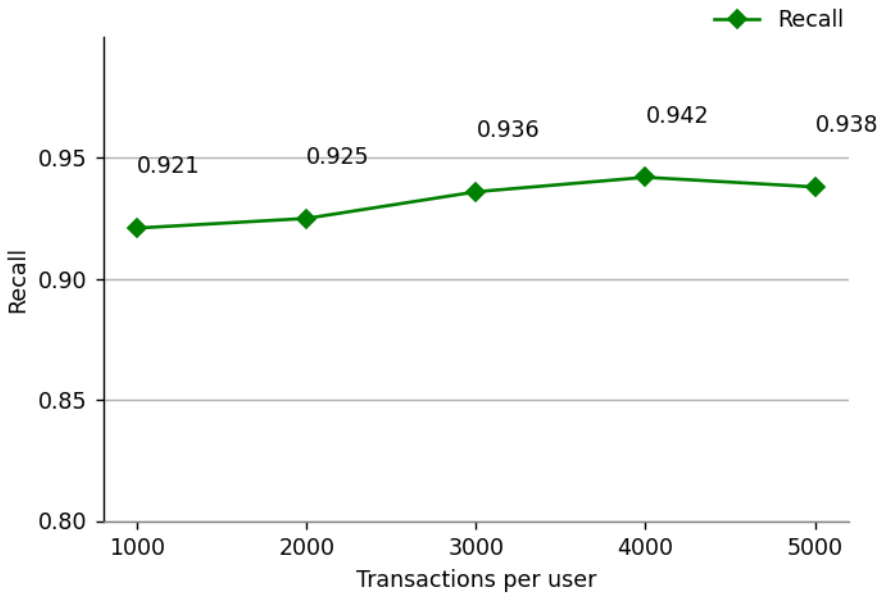


Figure 5.12: Variation of Recall with number of transactions per user at Threshold 60

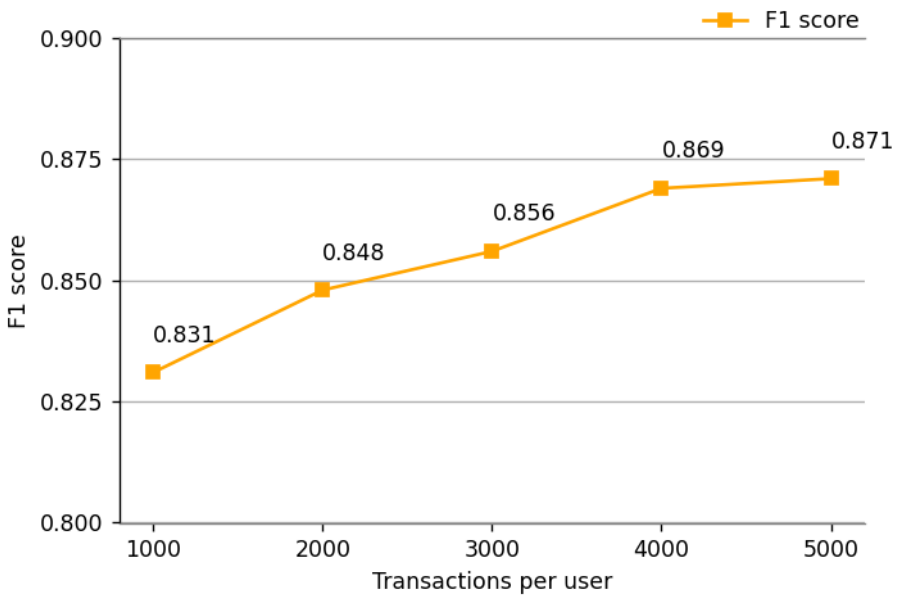


Figure 5.13: Variation of F1 score with number of transactions at Threshold 60

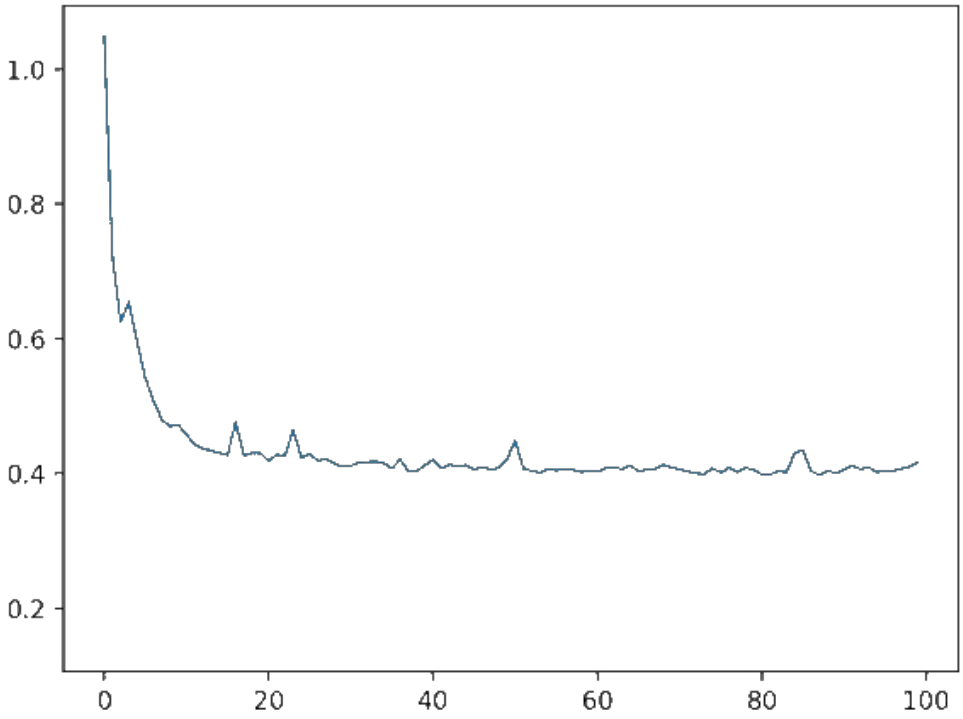


Figure 5.14: Loss of mhGW-WOA Clustering

Threshold = 60					
Parameter	Number of Transactions				
	1000	2000	3000	4000	5000
Precision	0.772	0.792	0.785	0.801	0.818
Recall	0.921	0.925	0.936	0.942	0.938
F1 score	0.831	0.848	0.856	0.869	0.871

Table 5.10: Precision, Recall, F1-score at Threshold 60

We also assess how well our role clustering algorithm performs. Figure 5.14 shows the variation of the loss of the clustering algorithm. As observed from the graph, the loss steadily decreases initially and then reaches a stable value and sustains it. Loss value becomes less than half of its initial value which is an impressive improvement and shows the adaptability of the algorithm on the dataset. This behaviour validates that the algorithm is performing well for the study.

Table 5.11 contrasts the various techniques based on the algorithm used, including but not limited to its ability to avoid intrusion and higher performance on metrics such as precision and recall. On comparing the metrics across different support values for each

technique, the maximum values were considered. We observed better performance for our algorithm compared to the alternatives. The critical factor behind the improved performance is the sensitivity of the algorithm to change in user patterns since we consider the relative adherence to data dependencies. This behaviour is expressed by real-world transactions since they never completely comply with data dependencies. Since our algorithm combines the benefits of anomaly-based and statistical detection methods, we are able to reduce the number of false negatives and false positive errors.

Author Name	Proposed Technique	Use of RBAC	Intrusion Detection Capabilities	Approach for Query evaluation	Detection Level	Detection of Threats	Detection Paradigm	Performance
Hu et al. [25]	Integrated dependency with sequence alignment analysis	No	Partial	Syntax Centric	Transaction level	External and user threats only	Anomaly	Recall = 0.90 Precision = 0.64
Srivastava et al. [86]	Integrated dependency with sequence alignment analysis	No	Partial	Syntax Centric	Transaction level	External and user threats only	Anomaly	Recall = 0.77 Precision = 0. 80
Kamra et al [31]	Dependency And Relation Analysis	Yes	Yes	Syntax Centric	Transaction level	Insider using RBAC, External using clustering	Anomaly	Recall = 0.63 Precision = 0.77
Hashemi et al [87]	Temporal Mining	No	Yes	Syntax Centric	Transaction Level	Both external and insider threats	Anomaly analysis of the time series	Recall = 0.90 Precision = 0.75
Kundu et al. [40]	Sequence alignment analysis	No	Partial	Syntax Centric	Transaction level	External and user threats	Anomaly	Precision = 0.945
Panigrahi et al [53]	User Behavior Mining	Yes	Partial	Syntax Centric	Transaction level	Both External and Insider threats	Both Anomaly detection and misuse detection	Recall = 0.93 Precision = 0.91
Doroudian et al. [61]	Mining dependencies	No	Yes	Syntax Centric	Transaction level	External and user threats	Anomaly and specification based	Recall = 0.89 Precision = 0.91
Ronao and Cho [55]	Weighted Random Forest	Yes	Yes	Syntax Centric	Root level	External and user threats	Anomaly	Recall = 0.95 Precision = 0.89
Sallam et al [48]	Anomaly Detection using Bayesian Classifier	Yes	Yes	Data and Syntax Centric	RDBMS level	Insider threats	Anomaly	Recall = 0.97 Precision = 0.90
Kim and Cho [52]	CNN - LSTM Neural Network	Yes	Yes	Data Centric	RDBMS level	User threats and Insider threats	Anomaly	Recall = 0.887 Accuracy of 0.933
Seok-Jun et al. [88]	Convolutional Neural-based Learning Classifier	Yes	Yes	Data-Centric	RDBMS level	Insider Threats	Anomaly	Recall = 0.93 Precision = 0.92
FPGWWO	Frequent Sequential Pattern Mining and metaheuristic clustering	Yes	Yes	Data and Syntax centric	Transaction level	External and insider threats	Anomaly	Recall = 0.938 Precision = 0.91

Table 5.11: Performance Comparison of FPGWWO with state-of-the-art techniques

5.3 Summary of the Chapter

This chapter presents two approaches., Firstly a novel intrusion detection and prevention technique that accommodates the behaviour of users at individual as well role level. Along with that, this approach relies on extraction of data dependency rules from transaction databases and combines it with user profiles generated by means of modified PSO clustering algorithm to classify the transaction as malicious or non-malicious. Combining- both dependency rules as well as role profiles helps to identify both previously seen attacks as well as novel attacks that deviate from authorized user profile. Secondly, a Database Intrusion Detection System (DIDS) to prevent unauthorised access from changing critical user details. Our method (FPGWWO) can identify threats from both within and out-

side the system by incorporating a frequent sequential pattern mining algorithm and a hybrid metaheuristic clustering using modified GWO and WOA Optimization Algorithm (mhGW-WOA). Frequent Sequential Pattern Mining Algorithm i.e CM-SPADE is used by the pattern miner module to mine database logs and constructing the read and write rules which are based on legitimate access patterns. Through the activities in the database logs, the hybrid swarm clustering algorithm (mhGW-WOA) clusters out users based on their role profiles. A new transaction is being evaluated by using the above two modules for ensuring that each user's role is matched in the current role profiles and the congruity index is determined which are based on the mined read and write rules for categorizing the queries as malicious or non-malicious. Our future research will focus on improvising ways for integrating user behaviour and other elements of transaction pattern mining utilising more efficient mining algorithms. It would also take into account the limits of the current technique in terms of adding new features and calculating adherence to improve the model's performance.

Chapter 6

Trust Factor and Outlier Based Intrusion Detection

In the previous chapter a novel approach towards database intrusion detection systems (DIDS) based on Frequent Closed Sequential Pattern Mining and Modified PSO Clustering and a frequent sequential pattern mining and a modified metaheuristic hybrid clustering of GWO and WOA optimization algorithm was introduced, whereas in this chapter, Trust factor based user behavior analysis using sequential pattern mining for DIDS called as TFUBID and Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining called OIUWSPM are introduced. Malicious updates to the database along with a breach in confidentiality may lead to loss of reputation, litigations and may lead to severe financial ramifications for organizations, and thus protecting data from malicious insiders is pivotal to their smooth functioning.

The CERT (Computer Emergency Response Team) report on insider threats [96] defined malicious insiders as employees and business partners who have authorized access to company database, but wilfully misuse their privileges to adversely impact the confidentiality, integrity and availability of an information system. Since insiders either have full or at least partial access to the organizational database, insider threats are harder to detect than threats from outsiders [97] . Traditional mechanisms like access control and authentication is inadequate in meeting the security needs of modern organizations. Furthermore, most traditional security mechanisms are preventive in nature, however many insiders can easily corrupt the data stored in the database. Since most attacks by insiders and outsiders who have assumed the identity of legitimate insiders cannot be prevented, more advanced and effective data-driven mechanisms are needed.

In this chapter, we firstly develop an effective database IDS called TFUBID to prevent

misuse of access privileges by insiders of an organization. We cluster vectors illustrative of the behaviour of each user in order to model general behavioural patterns rather than user specific patterns only. This is based on the assumption that groups of users access the organizational database for similar purposes. We cluster the user vectors using fuzzy clustering wherein each data point can be a member of more than one cluster. Fuzzy clustering is used since employees may belong to different groups in the organisation, and therefore their behavioural vectors may belong to more than one cluster. For each user we maintain a score on the basis of his access pattern in the past. We update this score with every transaction and use this for quantifying the maliciousness of the user.
To summarize our paper has the following major contributions:

1. We present a novel technique to dynamically determine the legitimacy of transactions using historical access patterns of users.

2. We have designed a new architecture that aims to prevent access privilege misuse, unauthorised access and modification of both critical and non-critical attributes that are under direct relationship with critical attributes.

3. We introduce the notion of an "incredulity score" which quantifies the degree of anomalous behaviour exhibited by each user based on his previous transactions.

Secondly, we developed Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining (OIUWSPM) which is a novel method for detection of malicious transactions through a sequential flow from outlier detection followed by different behavioural checks at the role induced rule mining component and user level feature extraction.The Outlier Detection module generates clusters based on the syntactic characteristics of transactions. Role level analysis is based upon dynamic usage of attributes local to every role domain. User profiling is structured by stockpiling legitimate transactions committed by the user. Security checks are made at every level to prevent further analysis of a transaction to reduce false positive rate and achieve a higher degree of optimisation.
Our main contributions are summarized as follows:

1. We develop a DIDS dependent on the postulates of RBAC for detecting the presence of insider threats which involves a hierarchical increase in granularity of analysed

data based upon the results of outlier, role and user level detection to classify the transactions.

2. We utilize the idea of Dynamic Sensitivity distinctively at role level which complements the access counts of each attribute.

3. We introduce the notion of Coherence Count calculated by the application of Longest Common Subsequence(LCS) and the utilization of Levenshtein distance for calculating Divergence between the user level Relation access paths.

4. We present novel approach for analysing user behaviour by registering user level Relation access path reinforced on user transactions.

The remaining chapter is organized as follows. Section 1 presents TFUBID model , while Section 2 presents our novel approach OIUWSPM model and we finally conclude the chapter by summarizing our finding and exploring avenues of future work in Section 3.

6.1 First Approach:Proposed TFUBID Model

We propose a two-phase intrusion detection and prevention model TFUBID (Trust factor based user behaviour analysis for database intrusion detection) that clusters users based on similarity of their data access patterns and the types of queries submitted by them, i.e. our model tracks the access patterns of all users and classifies them as normal or malicious. The superiority of our model lies in its ability to prevent unauthorised retrieving and modification of most sensitive data elements referred to as IDAs or Integral Data Attribute. Our model also makes sure that the query pattern for access of IDAs is specific and fixed for a particular user to avoid data breaches, i.e. the user associates himself with his regular access behaviour. Any deviation from the usual may lead to a reduction in the user's confidence and may act as representative of user's malicious intent. Some attributes can be used to indirectly infer IDAs and are therefore critical to the functioning of the organisation. For instance, account number of a user may be used to access the signatures. Such attributes are referred as DCA (Directly Correlated Attributes).
The proposed architectures are given in Figure 6.1 and Figure 6.2.

6.1.1 Basic Notations

In this chapter, the following major terminologies are used:

Definition 1. (Transaction) A set of user executed queries. Each transaction is represented by a unique transaction ID and also carries the user ID. Hence $< U_{id}, T_{id} >$ act as unique identification key for each set of query patterns. Each Transaction T is denoted as :
$< U_{id}, T_{id}, < q_1, q_2, ...q_n >>$ where, q_i denotes the i_{th} query, $i \in [1...n]$
For example, suppose a user has ID 1001 and executes the following set of SQL queries:

```
q1: SELECT a,b,c FROM R1,R2 WHERE R1.A > R2.B
q2: SELECT P FROM R5 WHERE R5.P==10
```

Then this can be represented as a transaction of the form:
$t =< 1001, 67, < q1, q2 >>$

Definition 2. (Integral Data Attributes (IDA)) These data attributes are of prime significance and have direct correlation with the integrity of the system. In a vertically hierarchical organisation, these are the attributes accessed only by the top level management, and the access by lower levels of hierarchy is strictly protected (refer to Table 6.1). User access pattern for IDAs is used by our model as an indicative of user's behaviour which helps in detection of malicious activities in the system.

Type of Attribute	**Sensitivity Level**
Integral Data Attributes	Highest
Directly Correlated Attributes	Medium
Normal Attributes	Low

Table 6.1: Sensitivity level of Different Attributes

Definition 3. (Critical Rules (CR)) A set of rules that contain a Integral Data Attributes

in its antecedent or consequent.

$$\begin{aligned} CR = \zeta \mid (\zeta \in RR \vee \zeta WR) \\ \cap (x \in IDA \cap (R(x1), R(x2)... \\ \Rightarrow O(x) \cup O(x) \Rightarrow W(x1), W(x2)...)) \end{aligned} \tag{6.1}$$

We propose a method of user Access Pattern Recognition using the Critical Rules, which represent the user query pattern associated to the Integral Data Attributes.

Definition 4. (Directly Correlated Attributes (DCA)) are attributes except IDAs, which are either part of antecedents or consequents of Critical Rules.
$DCA = \{\mu_i \mid \mu_i \in CR \cap \mu_i \notin IDA\}$.
For example, if $R(b) \rightarrow R(a)$
$R(b), R(c) \rightarrow R()$ and if a is a IDA, then the set {b, c} represents DCAs.
The query patterns as perceived by our model TFUBID are explored using DCAs that represent the first level of access of the IDAs. The first level of access means that these elements are the connecting elements and are accessed before IDAs or IDAs are accessed via these elements. A user's behaviour is characterized by a series of first-order statements (derived from queries) called attribute hierarchy embedded in first-order logic, which defines abstraction, decomposition and functional correlations among different access arrangements. The unit-transactions accessing IDAs are decomposed into attribute hierarchy comprising of DCAs, which further represents the user's most sensitive retrieval pattern.

Definition 5. (Incredulity Score(ϕ)) Incredulity Score is indicative of the amount of deviation of the user's access pattern from his designated role. This score summarizes the user's historic malicious access attempts and attempts to quantify the extent of vulnerability that the organisation faces because of a particular user. Incredulity Score combined with the deviation of user's present query from his normal behaviour pattern, yields the output of the proposed IDS. For our paper:
$0 <= \phi <= 1$.
Higher the Incredulity Score, more is the evidence of user deviating from the assigned role and the higher is the malicious intent.

Definition 6.(Trust factor) Trust factor is derived from the incredulity score and

quantifies how credible a user is. It is calculated as

$$TF = 1 - \phi$$

Based on the values of the Trust factor (refer to Table 6.2), we have proposed a division of Users among different levels, which suggests how likely a user, is going to execute a malicious query. A low trust level suggests that a user is more likely to execute a malicious query while a user with Medium trust level is less probable to do so as compared to the ones belonging to Low trust level. High Trust Level denotes the class of genuine users which will execute malicious queries with very less probability and thus can be trusted. It must be noted that the Trust Factor ranges are subjective and may differ from one organisation to the other.

TF Range	**Trust Level**
0 - 0.3	Low
0.3 - 0.6	Medium
0.6 - 1	High

Table 6.2: Levels of trust factor

Definition 7. (Incredulity Table) In our approach, we also maintain incredulity score of each user.The initial Incredulity scores for all users are set to 1. The initial incredulity score is shown in Table 6.3 as ϕ_i . Initially a user is assumed to be malicious due to no previous record but with each query or update in future we determine whether the user is credible or not. Thus scores of all users are set to 1. The Incredulity table is updated each time a user executes query.

Consider a user having U_{id} 1001 who has a deviation from normal query equal to 0.81. Then the updated Incredulity table is as shown in Table 3 as $\sqrt{ICS * \phi_i}$. In the table ICS represents the deviation from normal query and ϕ_i denotes the Initial Incredulity score.

U_{id}	ϕ_i	$\sqrt{ICS * \phi_i}$
1001	1	0.9
1002	1	1
1003	1	1
1004	1	1
1005	1	1

Table 6.3: Initial incredulity score

6.1.2 Learning Phase

Our model aims to generate user-profiles from transaction-logs, and quantifies deviation from normal behaviour. In particular, this phase aims to recognise and characterise the user activity pattern on the basis of their queries arrangement. Figure 6.1 illustrates various components of the architecture of the proposed model.

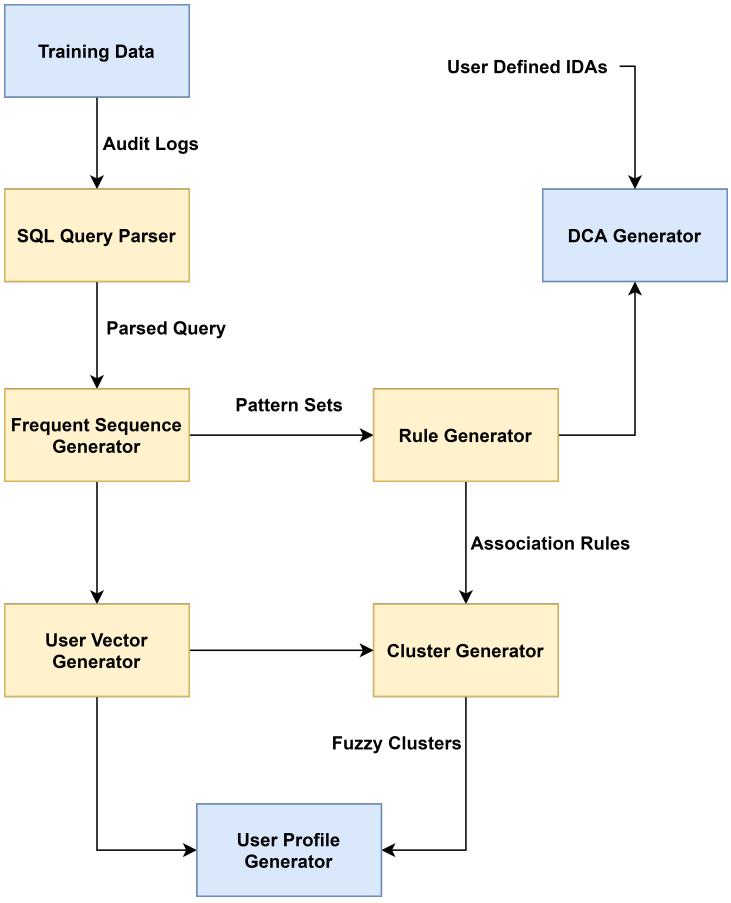


Figure 6.1: Learning Phase Architecture

6.1.2.1 Training Data

Our initial input to the learning phase algorithm is the transaction log comprising of only authorised and consistent transactions. This data is free of any unauthorised activity and is used to form user and role profiles based on normal user transactions. The logs are scanned to extract SQL queries and the User ID of the users who executed them.

6.1.2.2 SQL QUERY PARSER

This module takes SQL queries as input, parses them and produces read and write sequences as output. The query parser also assigns a unique Transaction ID. The final output consists of: Transaction (T_{ID}), User ID(U_{ID}) and read-write sequences generated using parsing algorithm. As an example consider the 10 transactions shown in Table 6.4. For each transaction, the table comprises of the ID of the user executing the transaction, its transaction ID and its corresponding read-write sequence.

User Id	**Transaction Id**	**Transaction**
2001	101	$< r(n), r(a), r(m), w(f), r(a), w(e) >$
2002	102	$< r(a), r(f), w(a), r(e), r(f), w(e) >$
2001	103	$< r(n), r(m), r(b), w(e), r(n), r(c), w(m),$ $r(a), r(m), w(b).r(c), r(f), r(b), w(f) >$
2003	104	$< r(f), r(m), w(f), r(f), w(e) >$
2003	105	$< r(b), w(b), r(e), r(n), w(c), r(m), w(f), r(a), w(e) >$
2001	106	$< r(m), r(f).w(f), r(c), r(e), w(e), r(c), w(n) >$
2002	107	$< r(f), r(c), r(m), w(n) >$
2004	108	$< r(e), w(m) >$
2003	109	$< r(b), r(f), w(m) >$
2004	110	$< r(m), r(a), w(c), r(a), w(m), r(b), r(n), r(e), w(b) >$

Table 6.4: User Access Sequences

Frequent Sequences are generated on the basis of minimum support as described in the next subsection.

6.1.2.3 Frequent sequences generation

The read and write sequences generated by the SQL query parser are given as inputs to the frequent sequences generator. These sequences are first pre-processed where they are assigned weights according to type of data items, for instance the IDAs are given greater weight as compared to DCAs and other normal attributes. This module uses the PrefixSpan algorithm [75] to generate frequent sequences out of the input sequences corresponding to each U_{ID}. PrefixSpan uses pattern growth method to extract frequent sequences , avoiding a huge candidate sequence generation leading to better memory utilization and execution time. Table 6.5 shows thirteen sequential patterns which fulfill the support constraint. For example, sequential pattern $< r(m), w(f), w(e) >$ is supported by transactions 1, 4, 3, and 6. The sequence $< r(n), w(a), r(m) >$, which is solely supported by transaction $< 1 >$, is an illustration of a sequence that fails to comply to minimal support.

Sequential Patterns
r(c)
w(m)
r(a), w(e)
r(b), w(b)
r(b), w(n)
r(e), w(e)
r(f), w(e)
r(f), w(f)
r(m), r(f)
r(m), w(f), w(e)
r(n), r(m), r(a)
r(n), r(m), w(e)
r(n), r(m), w(f)

Table 6.5: Sequential patterns where support exceeds threshold value

Read Rule
$r(a) \rightarrow w(e)$
$r(b) \rightarrow w(b)$
$r(b) \rightarrow w(n)$
$r(e) \rightarrow w(e)$
$r(f) \rightarrow w(e)$
$r(f) \rightarrow w(f)$
$r(m) \rightarrow r(f)$
$r(m) \rightarrow w(f)$
$r(n), r(m) \rightarrow r(a)$
$r(n), r(m) \rightarrow w(e)$
$r(n), r(m) \rightarrow w(f)$
Write Rule
$W(f) \rightarrow w(e)$

Table 6.6: Read and write rules

6.1.2.4 Rule generation

The frequent sequences are given as inputs to the rule generator module which uses association rule mining to generate read rules and write rules from these frequent sequences. Table 6.6 consists of read and write rules returned by the rule generation module.
By analysing the sequential patterns which are mined in Table 6.5, a few examples can be utilized for finding data dependencies while others ought not be considered. To begin with, the sequential patterns like $< r(c) >$ only contains the operation on a datum and have no data dependency, thus they shouldn't be taken into consideration. Furthermore, certain sequential patterns that are mined only included read operations. We focus on the transactions containing write operations as these are related with malignant data item modifications by user transactions. As a result, patterns with only read operations are also discarded.

6.1.2.5 DCA Generation

In our approach, we semantically define a class of data items known as Integral Data Attributes or IDAs. These IDAs and rules are given as input to our DCA (Directly Correlated Attributes) generator which specifies all those elements as DCA which are present in either the antecedent or the consequent of those rules that involve at least one of the IDAs. As an example, Table 6.7 shows the distribution of data items into the aforementioned classes. Once the Integral Data Attributes are known, the model uses them to generate critical rules (Definition 6). Table 6.8 shows the critical rules mined using access pattern recognition.

IDA	**DCA**	**Normal**
n,e	a,b,f,m	c

Table 6.7: Classification of Attributes

Critical Rules
$r(a) \rightarrow w(e)$
$r(b) \rightarrow w(n)$
$r(e) \rightarrow w(e)$
$r(f) \rightarrow w(e)$
r(n), $r(m) \rightarrow r(a)$
r(n), $r(m) \rightarrow w(e)$
r(n), $r(m) \rightarrow w(f)$

Table 6.8: Critical Rules

Algorithm 1: DCA Generator

Data : IDA,
Set DCA = {},
RR = Set of Read Rules,
WR = Set of Write Rules

Result : The set of Directly Correlated Attributes DCA

Function: DCA Generator (IDA, RR, WR)

```
1  begin
2      Initialization
3      for Ω ∈ RR ∪ WR do
4          for α ∈ Ω do
5              if α ∈ IDA then
6                  while β ∈ Ω do
7                      DCA{} ← β
8                  end
9              end
10         end
11     end
12 end
```

6.1.2.6 User vector generation

Using the frequent sequences for the given audit period this module generates user vectors which are representative of a user's activity.

A user vector is of the form

$$B_{ID} =< U_{ID}, w_1, w_2, w_3, ...w_n >; where, w_i = |O(a_i)| \tag{6.2}$$

$|O(a_i)|$ represents the total number of times user with the given U_{ID} performs operation ($O \in \{R, W\}$) on the attribute a_i in the pre-defined audit period. An audit period τ is defined as time period with a time window $\tau = [t_1, t_2]$. Each of these w_i would represent

how frequently a user performs the operation on the particular data item

$$UV_{ID} =< U_{ID}, < p(a_1), p(a_2), ..., p(a_n) >; \\ where, p(a_k) = \frac{w_k}{\sum_{w_j \in B_i} w_j} \tag{6.3}$$

$p(a_k)$ is defined as the probability of accessing the attribute a_k.
Value of $p(a_k)$ close to 1 would mean that the user accesses the given attribute frequently. As an example, consider Table 6.9 which gives a count of operations performed on data item by a specific user. This table is then used as input to calculate the probability of access which in turn helps in generating each user's profile vector as shown in Table 6.10.

User Id	w_a	w_b	w_c	w_e	w_f	w_m	w_n
2001	3	3	4	4	5	5	3
2002	2	0	1	2	3	1	1
2003	1	3	1	3	5	3	1
2004	2	2	1	2	0	3	1

Table 6.9: User vector generator

$< 2001, < 0.11, 0.11, 0.15, 0.15, 0.185, 0.185, 0.11 >>$
$< 2002, < 0.2, 0, 0.1, 0.2, 0.3, 0.1, 0.1 >>$
$< 2003, < 0.058, 0.176, 0.068, 0.176, 0.29, 0.176, 0.058 >>$
$< 2004, < 0.2, 0.2, 0.1, 0.2, 0, 0.3, 0.1 >>$

Table 6.10: User Profile Vector

6.1.2.7 Cluster Generation

The Cluster generator takes user vectors and rules as input and generates fuzzy clusters. Users are clustered into different fuzzy clusters based on the similarity of their user vectors using the Fuzzy c-means [98, 99]. clustering algorithm A cluster profile would include $C_i =< C_{ID}, \{R\} >$ where, C_{ID} represents the cluster centroid, and {R} is a set of rules which is formed by taking the union of all the rules that the members of the given fuzzy cluster follow. Each user belongs to a cluster to a certain degree w_{ij}, where w_{ij} represents

the membership coefficient of the i^{th} user (u_i) with the j^{th} cluster. The centre of a cluster (α) is the mean of all points, weighted by their membership coefficients. Mathematically,

$$\begin{aligned} w_{ij} &= \frac{1}{\sum_{k=1}^{C} \left\| \frac{u_i - \alpha_j}{u_i - \alpha_k} \right\|^{\frac{2}{i-1}}}; \\ where, \alpha_k &= \frac{\sum_u w(u)^m u}{\sum_u w(u)^m} \end{aligned} \tag{6.4}$$

The objective function that is minimized to create clusters is: [100]

$$argmin = \sum_{i=1}^{n} \sum_{j=1}^{C} w_{ij}^{m} \left\| u_i - a_j \right\|^2 \tag{6.5}$$

where n is the total number of users, C is the number of clusters, and m is the fuzzifier. The dissimilarity or distance function used in the formation of fuzzy clusters is the modified Jenson Shannon distance [101] which is given as:
Given two user vectors,

$$UV_x =< U_x, < p_x(a_1), p_x(a_2), p_x(a_3), ...p_x(a_n) >> \tag{6.6}$$

and

$$UV_y =< U_y, < p_y(a_1), p_y(a_2), p_y(a_3), ...p_y(a_n) >> \tag{6.7}$$

of equal length n, the modified Jensen Shannon distance is computed as in equation (6.7),

$$A = (1 + p_x(a_i) * w(a_i)) \log_2 \frac{1 + p_x(a_i) * w(a_i)}{1 + p_y(a_i) * w(a_i)} \tag{6.8}$$

$$B = (1 + p_y(a_i) * w(a_i)) \log_2 \frac{1 + p_y(a_i) * w(a_i)}{1 + p_x(a_i) * w(a_i)} \tag{6.9}$$

$$D(UV_p \| UV_q) = \sum_{i=1}^{n} \frac{A + B}{2} \tag{6.10}$$

where, $w(a_i)$ is the semantic weight associated with the $a_{i^{th}}$ attribute.

6.1.2.8 User profile generator

The User profile generation module takes user vectors and the cluster profiles as input and generates user profiles of the form:

$$U_i =< U_{ID}, < p(a_1), p(a_2), p(a_3)...p(a_k) >, < c_1, c_2, ...c_C >>$$

Where U_{ID} is a unique ID given to each user, $< p(a_1), p(a_2), p(a_3), ...(a_n) >$ is a vector containing the probability of the user accessing a particular attribute, and $< c_1, c_2, ...c_C >$ is a vector representing the membership coefficients of the given user for C different clusters. For instance, consider a system with 4 fuzzy clusters and 4 attributes, the given table 6.11 illustrates the profile of user U_{1001}.

Input					Output
c1	c2	c3	c4	User Vector	User Profile
0.2	0.2	0.2	0.4	$< U\_\{1001\}, 0.2, 0.109, 0.9.0.6 >$	$< U\_\{1001\}, < 0.2, 0.1, 0.9, 0.6 >,$ $< 0.2, 0.2, 0.2, 0.4 >>$

Table 6.11: Profile of user U1001

6.1.3 Testing Phase

In this Testing phase, we test the trained model's efficacy for detecting malicious queries. A user query is obtained as input and it is compared with the user's historical access patterns to see if the test transaction is malicious. The model first checks whether the user is trying to access a IDA. If yes, the transaction is allowed only if the given user has accessed that IDA before. Next, it checks if any DCA is being accessed. A user can perform write operation on a DCA only if it was previously written by the same user, otherwise the transaction is termed as malicious. Next, we check if the transaction abides by the rules that are generally followed by similar users. Figure 6.2 demonstrates the testing phase architecture.

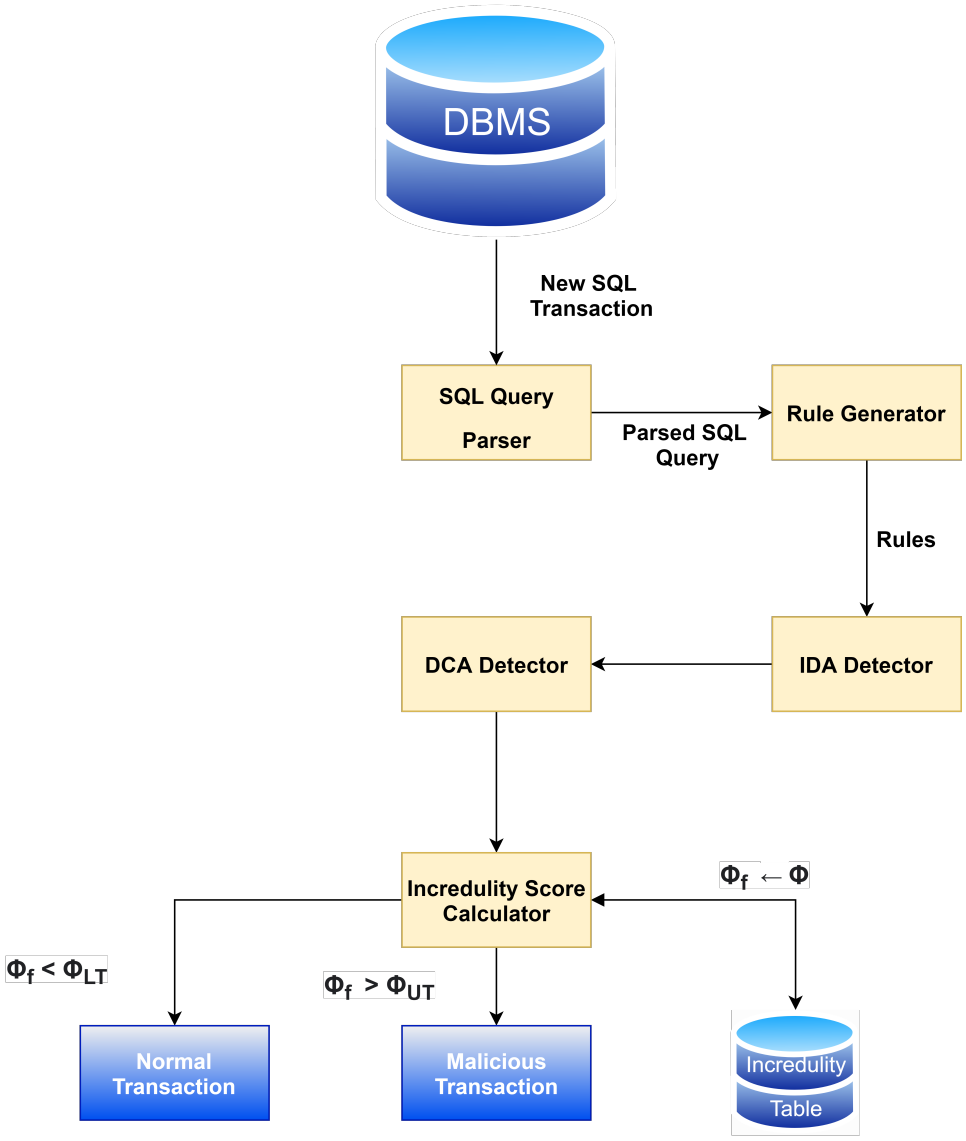


Figure 6.2: Architecture of Testing phase

6.1.3.1 Rule Generator

This module takes the read-write sequences generated by the SQL query parser and outputs rules followed by the input transaction. It uses PrefixSpan algorithm [75] for sequential pattern mining to generate rules. This can be a read rule or a write rule and indicates the operations done by the user, data attributes accessed by the user and the order in which they are accessed. Now this rule is used to check for deviation from the regular behaviour.

6.1.3.2 IDA Detector

The semantically critical elements referred to in our approach as IDAs are detected in this module. The read-write rule corresponding to the incoming transaction is checked for the presence of IDAs.

Algorithm 2: IDA Detector

Data : Set of rules (γ) from test transaction,
Set χ_{IDA}, U_{ID},
User Profile(Θ)

Result : Checks whether the test transaction is malicious or normal with respect to IDA

```
1  begin
2      Initialization
3      for ρ ∈ γ do
4          for λ ∈ ρ do
5              if λ ∈ χ_IDA then
6                  if w(λ) ∈ ρ & Θ[U_ID][w(λ)] == 0 then
7                      Raise Alarm;
8                  end
9                  if r(λ) ∈ ρ & Θ[U_ID][r(λ)] == 0 & Θ[U_ID][w(λ)] == 0 then
10                     Raise Alarm;
11                 end
12             end
13         end
14     end
15 end
```

If the rule being checked for intrusion contains a IDA, then it is dealt with using the following policy:

1. If read operation has been performed on any IDA, i.e. r(IDA) is present in the rule and UV[i][r(IDA)] = 0 and UV[i][w(IDA)] = 0 for the given user, then the transaction is termed as malicious.

2. If write operation has been performed on any IDA i.e. w(IDA) is encountered and V[i][w(IDA)] = 0 for the given user, then the transaction is termed as malicious.

6.1.3.3 DCA Detector

This module addresses the issue of inference attacks on IDAs. As discussed earlier, certain data elements can be used to access IDAs, i.e. first order inference. This module uses the rules mined in the learning phase to determine which elements can be used to directly infer the IDAs. Our system seeks to prevent inference attacks by closely monitoring write operations on DCAs. If write operation has been performed on any DCAs i.e. w(DCA) is present in the rule to be checked and UV[i][w(DCA)] = 0 for the given user, then the transaction is termed as malicious.

Algorithm 3: DCA Detector

Data : Set of rules (γ) from test transaction,
Set χ_{DCA}, U_{ID},
User Profile(Θ)

Result : Checks whether the test transaction is malicious or normal with respect to DCA

```
1  begin
2      Initialisation
3      for ρ ∈ γ do
4          for λ ∈ ρ do
5              if λ ∈ χ_DCA then
6                  if w(λ) ∈ ρ & Θ[U_ID][w(λ)] == 0 then
7                      Raise Alarm;
8                  end
9              end
10         end
11     end
12 end
```

6.1.3.4 Incredulity Score Calculator and Analyser

If the transaction has not been found malicious in the previous two modules, then we check whether the transaction is malicious based on the user's previous history and the behaviour pattern of all similar users. To do so, we maintain a record of action of all users using their Incredulity Scores. The deviation of a user's new transaction with his normal

access pattern is referred to as incredulity, and the corresponding measure of incredulity is the incredulity score. A user who is a potential threat tends to have a high incredulity score. We use ICS to keep a track of the maximum similarity of the given rule. We combine ICS with ϕ_i (Initial incredulity score) to get the final measure of incredulity score ϕ_f for the given user. We define 2 thresholds ϕ_{LT} and ϕ_{UT} . ϕ_{UT} represents the upper limit for the incredulity score of a non-malicious user whereas ϕ_{LT} denotes the lower limit. This means that if ϕ_f for a user comes out to be greater than ϕ_{UT} , the user is malicious. On the other hand, ϕ_f value less than ϕ_{LT} denotes a benign user. In particular, if the incoming rule (R_1) is a write rule, then the consequent of the incoming rule is matched with the corresponding rules in the user's cluster. A user is said to be the part of the i^{th} cluster if $\mu_i > \delta$.

Where, μ_i is the fuzzy membership coefficient of the given user for the i^{th} cluster and δ is a user defined weight. Otherwise, if the incoming rule (R_1) is a read rule, then the antecedent of the incoming rule is matched with the corresponding rules in the cluster of which a user is as part.

Algorithm 4: Weighted Jaccard Distance

Data : Rules R_1, R_2, δ_1, δ_2,
Set χ_{R_1}, χ_{R_2}

Result : Distance between the two rules (τ)

Function: jcDistance (R_1, R_2)

begin
  for $\rho \in \gamma$ **do**
    for $\Omega \in R_1$ **do**
      $\chi_{R_1} \leftarrow \Omega$
    end
    for $\Omega' \in R_2$ **do**
      $\chi_{R_2} \leftarrow \Omega'$
    end
    $\tau = 1 - \frac{\delta_1(\chi_{R_1} \cap \chi_{R_2}) - \delta_2(\chi_{R_1} \cup \chi_{R_2} - \chi_{R_1} \cap \chi_{R_2})}{\chi_{R_1} \cup \chi_{R_2}}$
    return τ;
  end
end

In order to quantitatively measure the similarity between two rules, we use weighted Jaccard distance:

$$JD = 1 - \frac{\delta_1(R_1 \cap R_2) - \delta_2(R_1 \cup R_2 - R_1 \cap R_2)}{R_1 \cup R_2} \tag{6.11}$$

$\forall R_2, \mu_i > \delta$ and $i \in [1, k]$. The minimum value of JD is regarded as ICS. ϕ_i is fetched directly from incredulity table. Final incredulity score for the given user is calculated as:

$$\phi_f = \sqrt{ICS * \phi_i} \tag{6.12}$$

1. If $\phi_f < \phi_{LT}$, the transaction is termed as non-malicious. In this case, the current incredulity score in the incredulity table for the given user is reduced by a factor known as "amelioration factor ($\mathring{A}$)". Thus, ϕ_i is updated as $\phi_i = \mathring{A}\phi_i$

2. If $\phi_{UT} < \phi_f \geq \phi_{LT}$, the transaction is termed as non-malicious and the incredulity table entry for the given user is updated with ϕ_f.

3. If $\phi_f \geq \phi_{UT}$ the transaction is termed as malicious.

As a viable example let us consider the initial incredulity score values of various users in the organization as shown in Table 6.12.

Table 6.12: Initial incredulity Score of each table

U_{id}	ϕ_i
1001	0.9
1002	0.8
1003	0.2
1004	0.6
1005	0.7

In addition, the minimum value of ICS for each user is shown in Table 6.13.Then the incredulity score can be calculated using Equation number 9 as shown in Table 6.14.

If we consider ϕ_{LT} as 0.3 and ϕ_{UT} as 0.6, the nature of each transaction by the users and the updated incredulity scores ϕ_f are summarized in Table 6.15.

Table 6.13: Minimum incredulity score corresponding to each user

U_{id}	ICS
1001	0.2
1002	0.3
1003	0.2
1004	0.6
1005	0.3

Table 6.14: incredulity score for users

U_{id}	$\phi_f = \sqrt{ICS * \phi_i}$
1001	0.42
1002	0.49
1003	0.2
1004	0.6
1005	0.46

6.1.4 Results and experiments

This section describes testing methodology for the proposed algorithm. Firstly, the datasets used for evaluation in the experiments is discussed. Secondly, we evaluate the performance of our approach on different accuracy metrics and compare our performance with the pertinent works.

6.1.4.1 Description of dataset

The proposed approach deals with detecting malicious user behaviors. An innate requirement for performance analysis is availability of an ideal dataset with concrete job functions. But actually, every organization or company refrains from sharing this data due to confidentiality and sensitivity constraints

6.1.4.2 Synthetic Dataset

To assess our approach on a wide gamut of attacks, we generated two different sets of malicious transactions. First set of transactions were generated randomly with the assumption that the attacker may be unaware of normal database dependencies. This set is created by randomly altering the characteristics of valid queries, which make up the norm, and by replacing the valid transactions of each query. The second set of transactions were generated such that they do not conform to the average user activity and are indicators

Table 6.15: Final Classification corresponding to each user

U_{id}	ϕ_f	**Nature of transaction**	**Updated** ϕ_f
1001	0.42	Non-malicious	0.42
1002	0.49	Non-malicious	0.49
1003	0.2	Non-malicious	0.198
1004	0.6	Malicious	0.6
1005	0.46	Non-malicious	0.46

of users trying to perform transactions outside of their scope and permissions.

6.1.4.3 Normal Transactions

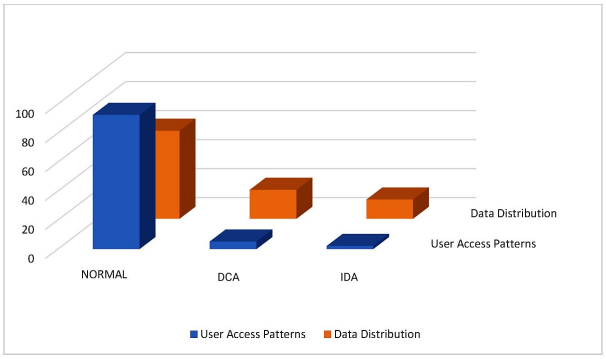
The performance of the algorithm is inspected by conducting out several tests on a credit card company dataset adhering to the TPC-C benchmark [63]. We use two audit logs: training log, testing log. The training log comprises of legitimate user transactions and testing log consists of a mixture of legitimate as well as malicious user transactions. To mimic the presence of unusal records in the real-world dataset, we have also injected some anomalies for detection. The injected anomalies are altered in a particular manner from legitimate transactions making them non-trivial to be detected.In conclusion, about 20,000 transactions were used where, about 99% of data was non-malicious while less than 1% of data was malicious. The reason for this imbalance can be accounted to the fact that we are trying to replicate the original database in a banking organization. The percentage of malicious transaction in a real-time database is very small when compared to the non-malicious transactions. Figure 6.3a highlights the distribution of the dataset used for analysis. The details of CDEs, DAEs and Normal data items has already been given in Section 3 and examples have been discussed in Section 5. The access pattern data hereby shows that CDEs are seldom examined, that too only by a few user roles and hence, protection of CDEs from malicious access is of major importance as compared to DAEs and Normal data elements.

6.1.4.4 Cluster Analysis

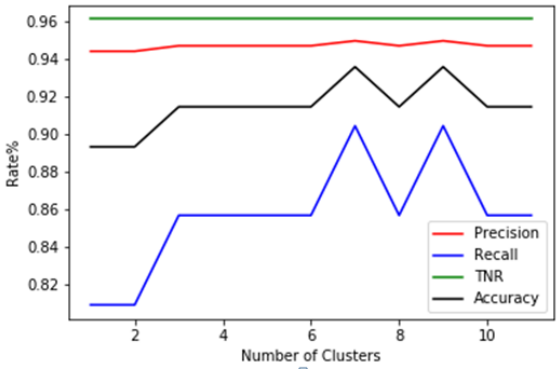
When the number of users/user roles go beyond a given threshold, it becomes infeasible for an IDS to keep track of individual user access patterns for detecting anomalies. Therefore, To improve the accuracy of the IDS while minimizing the performance loss, we have used clustering based approach. We have selected Fuzzy clustering over non-fuzzy clustering because there data is distributed into distinct clusters and each data point belongs to exactly one cluster. Membership grades are allocated to each of the data point in fuzzy clustering. These membership grades denote the extent to which data points belong to each cluster. While evaluating the performance by varying the number of clusters formed, it is observed that a certain number of clusters results stands out in performance. Moreover, both decreasing and increasing the number of clusters (with respect to this number) leads to loss in performance. Figure 6.3b depicts variation in precision, recall, TNR, accuracy with change in number of clusters. It can be observed that with respect to number of clusters formed TNR and Precision remain almost constant and invariant. On the other hand recall and accuracy have similar trends with minute change in the magnitudes. It can be seen that there is an overall increasing trend in recall and accuracy as number of clusters increases. Both recall and accuracy attain their optimal values for number of clusters 7 and 9. This is because the model is able to accurately capture two distinct groupings of data in the dataset. By partitioning the data into clusters in a way that reflects the underlying structure of the data, the model is able to accurately distinguish between these two groupings.

6.1.4.5 Distances and Thresholds

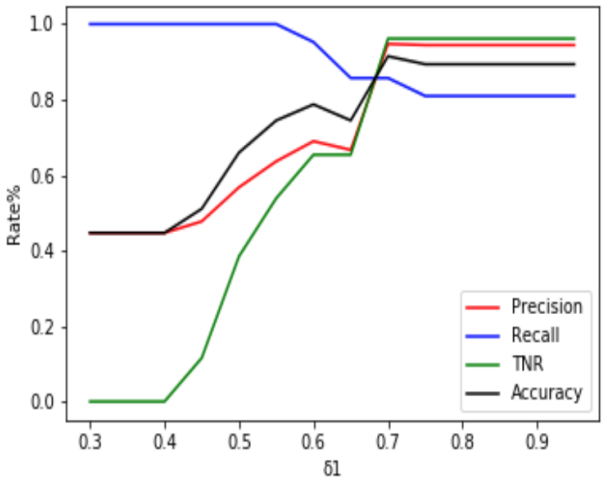
We have described Modified Jensen-Shannon distance as a measure to calculate distance between two user vectors of same length. The Jensen–Shannon divergence is a technique for estimating the similarity of two distributions. We preferred to use modified Jensen-Shannon distance to give weights to data attributes and avoid the curse of dimensionality. The variation of modified Jensen-Shannon distance with Euclidean distance is shown in the Figure 6.4c. we have defined modified Jaccard distance to quantitatively measure the similarity between two rules. The Jaccard index is a statistical measure used to compare sample set diversity and similarity. The Jaccard coefficient which is calculated as the ratio of size of intersection to the size of union of the sample sets, measures the similarity



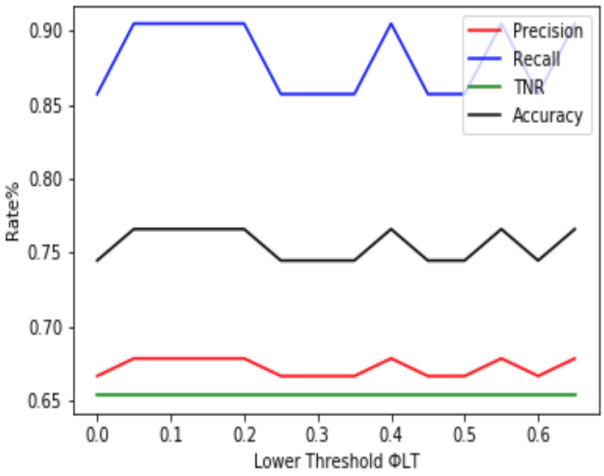
(a) Frequency of data items and their access frequency



(b) Variation of performance with number of clusters



(c) Variation of performance measures with δ_1

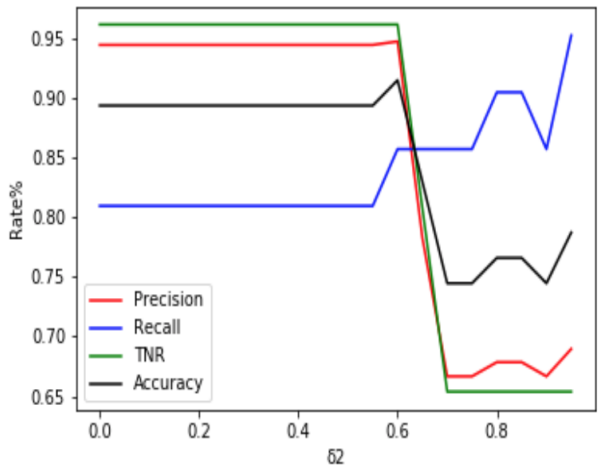


(d) Variation of performance measures with ϕ_{LT}

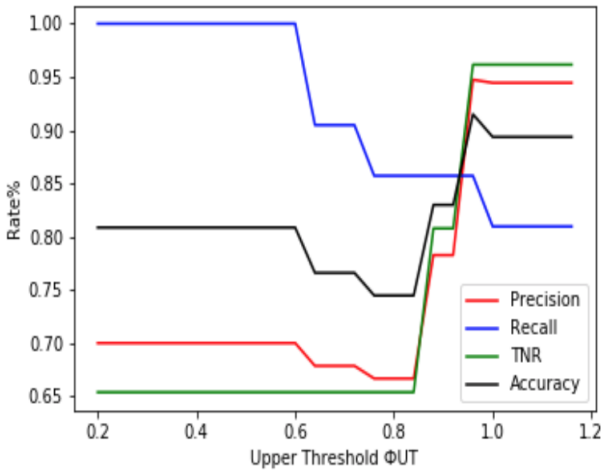
Figure 6.3: Variation of performance measures with different parameters

between finite sample sets. The variation of modified Jaccard index with Jaccard index is shown in Figure 6.4d.

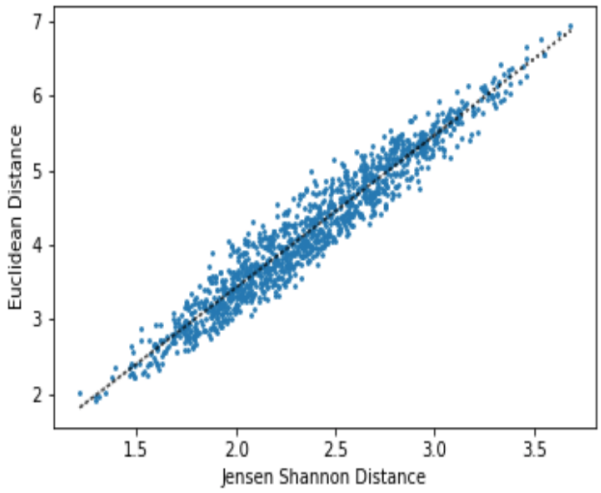
The variation of precision, recall, TNR, accuracy with the various thresholds, namely δ_1, δ_2, ϕ_{UT} , ϕ_{LT} . Figure 6.3c represents the variation of Precision, recall, TNR, accuracy with δ_1. It can be observed from the graph that Precision, TNR and Accuracy increase with the increase in value of δ_1, while the value of Recall decreases with increase in value of δ_1. The variation of Precision, recall, TNR, accuracy with δ_2 has been outlined in Figure 6.4a. It can be inferred from the graph that the value of Precision, TNR and Accuracy starts decreasing when the value of δ_2 increases beyond a certain value. Recall, on the other hand, increases for higher values of δ_2. Figure 6.4b further highlights the variation of Precision, recall, TNR, accuracy with ϕ_{UT}. It can be concluded that the value of Precision first decreases and then exponentially increases with the increase in value of ϕ_{UT} . An identical trend is followed by Accuracy. Somewhat similar trend is followed



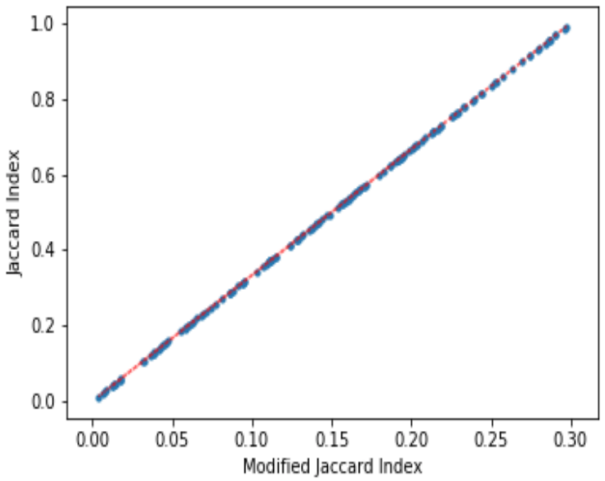
(a) Variation of performance measures with δ_2



(b) Variation of performance measures with ϕ_{UT}



(c) Euclidean distance vs Jensen Shannon Distance



(d) Jaccard Index vs Weighted Jaccard Index

Figure 6.4: Variation of Distances and Thresholds

by TNR except that it does not decrease initially. On the contrary, the value of Recall decreases with the increase in value of ϕ_{UT} . Figure 6.3d shows the variation of Precision, recall, TNR, accuracy with ϕ_{LT} . It can be observed from the graph that the values of all the parameters fluctuate a little but remain more or less constant with the increase in value of ϕ_{LT}.

With regards to the dataset we have used, following inferences can be drawn from the graphs:

1. Value of δ_1 should be close to 0.65 for optimum performance.
2. Value of δ_2 should be close to 0.55 for optimum performance.
3. Value of ϕ_{UT} should be close to 0.59 for optimum performance.

Table 6.16: Comparison of our approaches with related works

Sensitivity Measures	**TPR**	**ACC**	**F1 Score**
Approach 1	0.96	0.94	0.94
Approach 2	0.73	0.80	0.85
Approach 3	0.74	0.83	0.85
Hu et al. [25]	0.73	0.81	0.79
Cho et al. [52]	0.88	0.93	0.90
Hashemi et al. [87]	0.71	0.84	0.82
Majumdar et al. [36]	0.70	0.80	0.78
Bertino et al. [44]	0.91	0.93	0.92
Subudhi et. al. [32]	0.92	0.92	0.93
Seok-Jun et. al. [88]	0.93	0.92	0.92
Kim et. al. [102]	0.93	0.94	0.94
Bu et. al. [103]	0.85	0.88	0.86

4. Value of ϕ_{LT} should be close to 0.2 for optimum performance.

6.1.4.6 Comparison with related methods

Using performance measures such as Precision, Recall, F1-score and Accuracy, we also compare our approaches with other related works. Our different approaches are as follows:

Approach 1. Our approach using modified Jenson-Shanon distance and weighted Jaccard index.

Approach 2. Using unweighted Jaccard index with Jenson-Shanon distance.

Approach 3. Using Euclidean distance with unweighted Jaccard index.

Table 6.16 shows the comparison of our proposed approaches in terms of Precision (TPR), Accuracy(ACC) and F1 score. It can be observed that approach 1 outperforms other approaches in all the three aspects. The main reason behind the effectiveness of Approach 1 lies in its fundamental idea of using modified Jenson-Shanon Distance and weighted Jaccard Index both of which are able to capture fine characteristics of the user behaviour. It also provides a comparative analysis of our approach with prior methodologies based on the True Positive Rate, Accuracy and F1 Score. The performance of all three approaches was assessed using the aforementioned metrics. The evaluation results show that our algorithm outperforms all the compared methodologies. This can be ascribed to the low sensitivity of our algorithm to changes in user patterns since we incorporate user behaviour analysis in our approach.

It is interesting to note that Approach 3, although slightly, is able to perform better

as compared to Approach 2. Approach 3 employed Euclidean distance whereas Approach 2 used Jenson-Shanon Distance which is known to deliver better results over Euclidean distance yet the former performed better. This shows that the modification introduced in Jenson-Shanon Distance is imperative and has enhanced the overall performance of the model.

Figure 6.5 shows a detailed bar graph representing a comparison of three proposed approached on the basis of the magnitudes of various performance measures generated during the testing phase where PPV-Positive Predictive value, TPR- True Positive Rate, ACC-Accuracy, F1 Score, NPV- Negative Predictive value, FDR- False Discovery Rate, FOR-False Omission Rate , BM- Bookmaker informedness , FPR- False Positive Rate, TNR-True Negative Rate, FNR- False Negative Rate, MK- Markedness , MCC- Matthew's Correlation Coefficient.

Our approach consolidates the advantages of statistical Integral Data Attribute analysis and user behaviour analysis based anomaly-based detection methods, resulting in a reduction in both false positive and false negative errors.

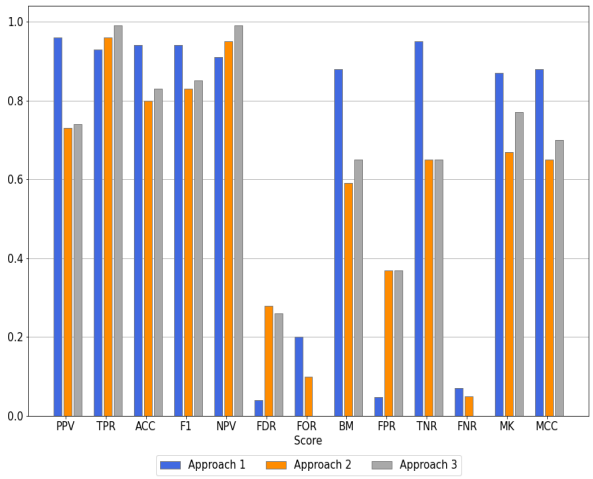


Figure 6.5: Comparison of Approaches on basis of various sensitivity metrics

6.2 Second Approach: OIUWSPM model

The presented IDS framework functions as an independent unit and amplifies the system safety without hindering the normal working of the database management system (DBMS).Our methodology evolves as per development of the following divisions:

1. Learning Phase
2. Detection Phase

6.2.1 Learning Phase

During the learning phase, intrusion-free transactions issued by database users are collected from the Query Log Database. These transactions are parsed and then processed to fetch fuzzy clusters, association rules and user behaviour Relation access paths from them which will be utilized in the detection phase. We develop learning phase at three specification levels. With every level, the eccentricity of the approach increases. Learning phase comprises of following components:

1. Input: Transaction Logs
2. Pre-processing Modules:
    (a) Query Template Generator
    (b) Sequential Pattern Miner
    (c) Sensitivity Calculator
3. Specification Levels :
    (a) Training for Outlier Analysis(Level 1)
    (b) Role Induced Learning using ARM(Level 2)
    (c) User Behaviour Learning by Feature Extraction(Level 3)

The proposed architecture for learning phase is shown in the Figure 6.6. For analyzing our architecture, the following definitions can be taken into account:

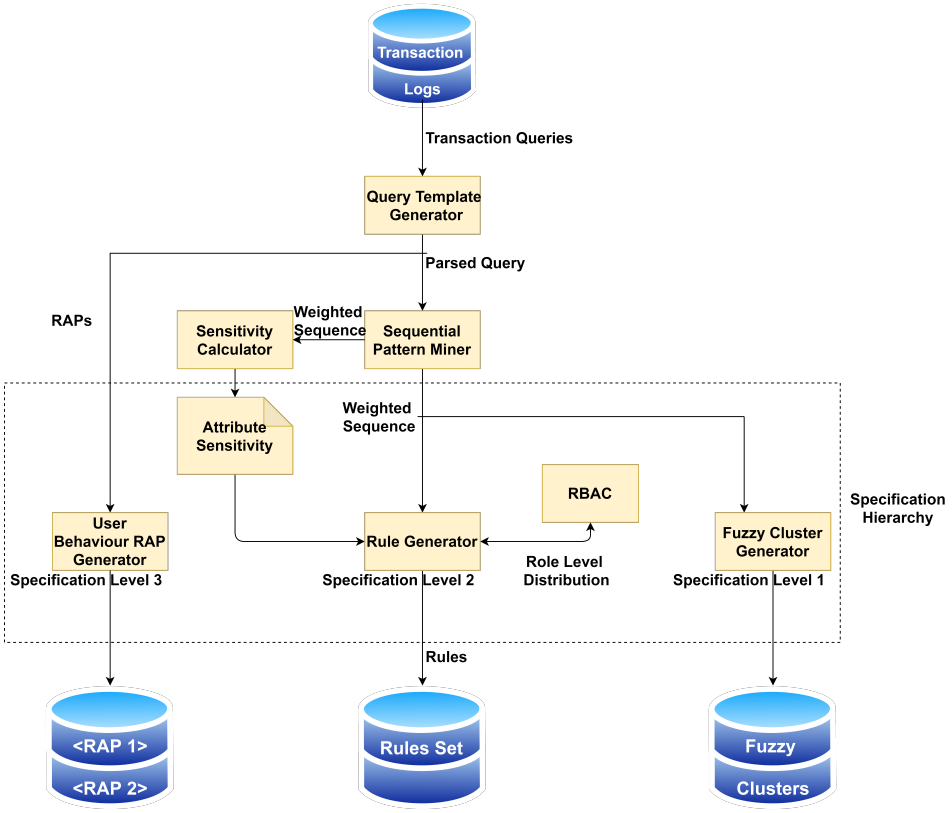


Figure 6.6: Learning Phase Architecture

Definition 8 (Transaction): A transaction is an ordered sequence of operations done on a database instance, with user operations marked by $O(atr)\ \epsilon\ \{Read, Write\}$. The processes must be carried out in a distinct manner in order for the transaction to be completed. Transactions are assigned $Query\_Id$ as identifier along with $User\_Id$ and $Role\_Id$.

Definition 9 (Read rules): **Read rules** are generated from Read Sequences. A read rule is defined as $\{r(a_1), r(a_2), ..., r(a_n)\} \rightarrow O(x)$, where x is any attribute and each attribute $a_1,\ a_2, ...,\ a_n$ should be unique. From all the rules generated by Sequential pattern mining on Read Sequence, only those rules which have the confidence value greater than threshold are considered to be fit for read rules. Sample generated read rules are shown in Table 6.17. To simplify representation, we have utilised Upper Case alphabets to denote a read operation on a sample database attribute *as shown in table 6.19* and lower case alphabets are used to denote write action on an attribute.

Definition 10 (Write Rules): **Write Rules** are generated from the Write Sequences. A write rule is defined as $O(x) \rightarrow \{w(a_1), w(a_2), ..., w(a_n)\}$, where x is any attribute and

S.No	Rules
1	GFH → I
2	ABC → D
3	AE → g
4	H → I
5	A → b
6	A → c
7	G → f
8	G → D
9	GCB → h
10	AEH → I

Table 6.17: Read rules

each attribute $a_1,\ a_2, ...,\ a_n$ should be unique. From all the rules generated by Sequential pattern mining on Write Sequence, only those rules which have the confidence value greater than threshold are considered to be fit for write rules. Sample generated write rules are shown in Table 6.18.

S.No	Rules
1	f → g
2	a → bcd
3	b → cd
4	c → d
5	f → e
6	h → i
7	a → bcdefghi
8	b → cdefghi
9	g → hi
10	f → ghi

Table 6.18: Write Rules

Definition 11 (Key Operation): The antecedent in the write rule and the consequent in the read rule is called the **Key Operation**.

For example: For read rule $AE \rightarrow g$, u is the consequent and thus the key operation. For write rule $b \rightarrow cd$, I is the antecedent and thus the key operation.

6.2.1.1 Transaction Logs

Database containing all the previous legitimate transactions committed by the users over time is called transaction logs and it consists of transactions with the following format:

$\langle' Query\_Id',' User\_Id',' Role\_Id',' Input\ Query'\rangle$

6.2.1.2 Pre-processing Modules

(a) Query Template Generator(QTG)

The training phase uses the transaction logs which contain past legitimate queries of multiple users, their user ID and the user role trying to execute it. The queries are parsed using the QTG succeeded by generation of query templates of the form:

$\langle `QueryID', `User\_Id', `Role\_Id', `Parsed\ Query', RAP\rangle$

The QTG extracts read/write sequence and an appropriate Relation access path(RAP).

Definition 12(Relation access path(RAP): Relation access path(RAP) represents the order in which all the relations in the addressed schema are accessed by the user.

Consider the Employee Database Schema given in Table 6.19. The attributes have been denoted by using the Alphabetical Notation i.e for every read operation (A-Z) are utilized and for every write operation (a-z) are utilized. Hence the symbol 'A' means that attribute 'A' i.e. EmpNo has been read from Employee relation and symbol 'n' means that attribute 'n' i.e. Salary has been updated in Salaries relation. The following transaction query is taken and parsed through the QTG:

Table ID	Relation	Attributes
1	Employee	Empno(A), FirstName(B), LastName(C), BirthDate(D) HireDate(E), Gender(F)
2	Department	DeptNo(G), DeptName(H)
3	EmpDept	EmpNo(I), DeptNo(J), StartDate(K), EndDate(L)
4	Salaries	EmpNo(M), Salary(N), StartDate(O), EndDate(P)
5	Titles	EmpNo(Q), Title(R), StartDate(S), EndDate(T)
6	DeptManager	DeptNo(U), EmpNo(V), StartDate(W), EndDate(X)

Table 6.19: Sample Database Schema

Input Query

Update Salaries as Sal, Employee as Emp

Set Sal.Salary = Sal.Salary + 10000

WHERE

Sal.EmpNo = Emp.EmpNo

AND

Emp.Gender = 'Female'

The complete transaction is input to the QTG as follows:

$\langle$'$Q1'$,'$U1'$,'$R1'$, $InputQuery\rangle$

The read/write sequence and the Relation access path fetched from the above query by the query template generator is as shown below.

Parsed Sequence: 'FAMn', where F denotes the reading of 'Gender' from Employee relation(1), A and M symbolize the equality check of the employee numbers 'EmpNo' in both the relations Employee(1) and Salaries(4), finally 'n' denotes the update operation performed on the attribute 'Salary' in Salaries relation(4). Consequently, the Relation access path is '1144', where '1' and '4' represents the access of Table 6.19 and Table 6.22 respectively.

The final Query Parsed is:

$\langle$'$Q1'$,'$U1'$,'$R1'$,'$FAMn'$,'1144'$\rangle$

Similarly, various other queries can be parsed:

$\langle$'$Q2'$,'$U2'$,'$R2'$,'$BCAQSTr'$,'1115555'$\rangle$

$\langle$'$Q3'$,'$U3'$,'$R3'$,'$BCAVu'$,'11166'$\rangle$

$\langle$'$Q4'$,'$U4'$,'$R4'$,'$JIAbc'$,'33111'$\rangle$

$\langle$'$Q5'$,'$U5'$,'$R5'$,'$ABCDEF'$,'111111'$\rangle$

(b) Sequential Pattern Miner

In our approach, the query templates generated by the previous module are used as input to the Sequential Pattern Miner(SPM) along with, the weights of all the attributes and minimum weighted support threshold (β_{wsup}). SPM produces weighted sequential patterns for each role by using IUA algorithm [104] which is illustrated in the following definitions [104]:

Definition 13(Weight of an attribute): The **weight of an attribute** 'a', represented by w_a is a measure of it's relevance in the database which will be determined by the Database Administrators(DBAs) and organisation level experts. It varies from 0 to 1.

Table 6.20 illustrates the weights of certain attributes taken from the database schema.

Attribute	Weight	Attribute	Weight
A	0.75	M	0.75
B	0.21	N	0.9
C	0.26	Q	0.75
D	0.43	S	0.76
E	0.77	T	0.76
F	0.81	R	0.85
I	0.75	U	0.39
J	0.39	V	0.75

Table 6.20: The Weights assigned to the attributes

Definition 14(Weight of a sequence): The **weight of a Sequence** S, denoted by w_S is the sum total of weights of all the attributes in S over the number of items in S.

$$w_s = \frac{\sum w_{a_i}}{N} \tag{6.13}$$

where N = len(S) and $a_i = i^{th}$ attribute in sequence S. Table 6.21 illustrates the sequences belonging to numerous user roles R1, R2, R3, R4, R5.

Role ID	Sequences Taken
R1	$\langle FAMn \rangle$
R2	$\langle BCAQSTr \rangle$
R3	$\langle BCAVu \rangle$
R4	$\langle JIAbc \rangle$
R5	$\langle ABCDEF \rangle$

Table 6.21: Example set of the sequences

Table 6.21 illustrates the sequences belonging to numerous user roles {R1, R2, R3, R4, R5}.

For example: Since the weights of the attributes in the itemset {F, A, M, n} belonging to the sequence $\langle FAMn \rangle$ are 0.81, 0.75, 0.75, 0.90 respectively, and the number of items

in $\langle FAMn \rangle$ is 4, hence $w_{\langle FAMn \rangle} = (0.81 + 0.75 + 0.75 + 0.90)/4 = 0.8025$.

Definition 15(Maximal Sequence Weight): The **Maximal Sequence Weight** of a sequence S, denoted by w_{MS} is the maximal weight among all the attributes in sequence S.

$$w_{MS} = max(w_{a_i}) \tag{6.14}$$

where $a_i = i^{th}$ attribute in S

For example, in Table 6.21 for the sequence $\langle FAMn \rangle$, $w_{l\langle FAMn \rangle} = maximum\{0.81, 0.75, 0.75, 0.9\} = 0.9$

Definition 16(Aggregated Maximal Sequence Weight): The **aggregated Maximal Sequence Weight** of a Sequence Database SDB, denoted by aw_M is the sum total of the Maximal Sequence Weights of all the sequences in SDB.

$$aw_M = \sum w_{MS} \tag{6.15}$$

where S is any sequence in SDB.

For example, in Table 6.21 the Maximal Sequence Weights of the five sequences are 0.9, 0.85, 0.75, 0.75 and 0.81 respectively. Then, $aw_M = 0.9 + 0.85 + 0.75 + 0.75 + 0.81 = 4.06$

Definition 17(Weighted Support): The **weighted support** of a subsequence S, denoted by $w_{sup}(S)$ is the sum total of the weights of subsequence S in all the sequences in Sequence Database(SDB) which are including S over the aggregated Maximal Sequence Weight(aw_M) of that SDB.

$$w_{sup}(S) = \frac{\sum w_S}{aw_M} \tag{6.16}$$

For example, in Table 6.21 the subsequence $\langle BC \rangle$ *is included by sequences* $\langle BCAQSTr \rangle$, $\langle BCAVu \rangle$ *and* $\langle ABCDEF \rangle$. *Hence weighted support of* $\langle BC \rangle$ *is:* $w_{sup}(\langle BC \rangle) = (w_{\langle BC \rangle} + w_{\langle BC \rangle} + w_{\langle BC \rangle})/aw_M = (0.235+0.235+0.235)/4.06 = 0.17365$

Definition 18(Weighted Sequential Pattern): A subsequence S is called a **weighted**

sequential pattern if the weighted support of that subsequence S, i.e $w_{sup}(S)$ is greater than or equal to a pre-defined weighted support threshold β_{wsup}.
For example, if we assume $\beta_{wsup} = 0.5$, then $\langle BC \rangle$ is not a weighted sequential pattern in Table 6.21 as $w_{sup}(\langle BC \rangle) = 0.17365$ which is not greater than or equal to β_{wsup}.

(c) Sensitivity Calculator
After the sequential patterns have been mined, the next step is the calculation of Sensitivity. Since these patterns are used for the computation of sensitivity, the process is said to have a dynamic nature [105]. Sensitivity is based on the concept of attribute accessibility(π(atr)), which is a metric used to assess the normalized use of each attribute at the role for every read and write operation performed.

Definition 19(Dynamic Sensitivity): **Dynamic Sensitivity** ($\Omega(atr)$) indicates the usage of the attribute at the role level.

The metrics used for Sensitivity calculation are:

1. $\boldsymbol{\mu_t(atr)}$: The value of μ_t for an attribute 'atr' is the number of weighted sequential patterns including 'atr' and
2. $\boldsymbol{\mu_r(atr)}$: The value of μ_r for an attribute 'atr' is the number of unique roles accessing 'atr'

The sensitivity of each operation varies from 0.1-0.9 as no attribute is entirely accessible (π(atr) = 1) or entirely inaccessible (π(atr) = 0). After mining sequential patterns for different user roles we generate an Operation Counting Table(OCT), which keeps a track of $\mu_t(atr)$ and $\mu_r(atr)$ for read and write operations on all the attributes. These metrics are independently normalized by Feature Scaling within the limits of 0.1 and 0.9.

We calculate the total accessibility ($\pi(O(atr))$) of an operation O on attribute 'atr' as the harmonic mean of $\mu_t(atr)$ and $\mu_r(atr)$.

$$\pi(O(atr)) = \frac{2}{\frac{1}{\mu_t} + \frac{1}{\mu_r}} \tag{6.17}$$

Harmonic mean is adopted against arithmetic mean for computing accessibility as it

emphasis whether $\mu_t(atr)$ and $\mu_r(atr)$ lies at the boundaries or amidst the scale. Dynamic Sensitivity($\Omega(O(atr))$) is defined from the total accessibility ($\pi(O(atr))$) as:

$$\Omega(O(atr)) = 1 - \pi(O(atr)) \tag{6.18}$$

Table 6.22 illustrates the computation of Dynamic Sensitivity of the read operations on a subset of the attributes {A, B, C, D} used in the database schema respectively.

Attribute	μ_t	$\mu_t(Normalized)$	μ_r	$\mu_r(Normalized)$	π	Ω
A	0.54320	0.9	1.0	0.9	0.9	0.09999
B	0.49387	0.82195	1.0	0.9	0.85920	0.14079
C	0.43209	0.72439	1.0	0.9	0.80270	0.19729
D	0.29629	0.50975	1.0	0.9	0.65086	0.34913

Table 6.22: Dynamic Sensitivity(Read operations)

6.2.1.3 Processing Modules

(a)Specification Level 1: Training by Outlier Analysis

In order to detect the presence of an outlier, the sequential patterns mined are clustered by using the Fuzzy c-means clustering algorithm [98]. The generated clusters are used to find the presence of a global outlier in the detection phase. It is based on minimization of the following objective function:

$$O_i = \sum_{u=1}^{L} \sum_{v=1}^{C} M_{uv}^{i} \, \|P_u - D_v\|^2 \;\; , 1 \leq i \leq \infty \tag{6.19}$$

where i is any real number greater than 1,

M_{uv} is the degree of membership of P_u in the cluster v,

P_u is the u^{th} of d-dimensional measured data and

D_v is the d-dimension center of the cluster

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership M_{uv} and the cluster centers D_v by:

$$M_{uv} = \frac{1}{\sum_{k=1}^{C} \| \frac{P_u - D_v}{P_u - D_k} \|^{\frac{2}{i-1}}} \tag{6.20}$$

$$D_v = \frac{\sum_{u=1}^{L} M_{uv}^{i} \; P_u}{\sum_{u=1}^{L} M_{uv}^{i}} \tag{6.21}$$

Cluster 1	**Cluster 2**	**Cluster 3**
4.972466	5.105628	8.439778
3.370616	6.230937	6.921812
2.084401	8.404267	9.674275
0.902631	8.081959	15.825793
0.550029	1.198404	0.940695
0.073882	0.232600	0.169938
0.078807	0.248107	0.181267
0.083732	0.263614	0.192597
0.088658	0.279120	0.203926

Table 6.23: Cluster Centers

The cluster with which the input query holds the maximum membership will be selected for detection at Outlier Level. Table 6.23 illustrates the cluster centers of 3 clusters generated after applying the fuzzy c-means clustering algorithm on the mined sequences.

(b) Specification level 2: Role Induced Learning Using ARM

As employees possessing the same roles frequently performs similar tasks and acquire identical access controls, therefore the sequential patterns developed for an employee in an organization are assumed to be consistent with the sequential patterns of other employees possessing the same role. Consequently we employ these weighted sequential patterns to develop association rules for each role by Association Rule Mining(ARM). These rules will be utilized in the detection phase to bifurcate transactions as malicious or non-malicious.

Rule Generator

Data dependency rules are generated for each user role by the Rule Generator from the frequent sequences mined by Sequential Pattern Miner using role sensitivities produced by sensitivity Calculator.

Definition 16(Minimum Sensitivity Threshold): The operations which have role sensitivity strictly greater than a pre-defined Minimum Sensitivity Threshold(§) are con-

sidered to be **Sensitive operations**. Rules are generated with respect to sensitive operations.

If and only if the key operation is sensitive, the generated read and write rules are added to the set of rules for that user role and will be used in detection phase by the Role Level Detector . The rule-generation algorithm is based on the observation that a set of less sensitive operations must be performed in order to access a highly sensitive operation.

Algorithm 5 represents **Sensitivity Galvanized Rule Extraction Algorithm(SGREA)** for the extraction of rules(satisfying the minimum sensitivity constraint) from the weighted sequential patterns aided by the Dynamic sensitivities for every read/write operation for every user role, thereby fulfilling the constraints established by RBAC. The Input to the rule mining algorithm includes a set of weighted sequential patterns for every user role, a well-defined mapping between the operation on an attribute (read/write) and its Dynamic Role sensitivity which works in O(1) time complexity and a minimum sensitivity threshold(K) for identifying the Key Operation in a subsequence. The algorithm gives two distinct lists comprising of read and write rules integrated with every user role.
Lines 12 to 22 deal with the generation of read rules. As it is evident from the line 11, for every sensitive operation in a sequence, all the read operations proceeding it are taken to be the antecedent with the key operation itself being the consequent. A check is made at line 21 to ensure that there is no redundancy of the same rule for that role. The generation of write rule follows a similar pattern and is well elucidated from lines 25 to 35 following the fact that for rule generation, all the write operations following the key operation in the sequence are taken to be the consequent with the key operation serving as the antecedent.

(c) Specification Level 3: User Behaviour Learning by Feature Extraction

At specification level 2, we produced same data dependency rules for all the users belonging to the same role which thereby neglects the individuality of a user. This leads to a flaw that these rules generic to the users affiliated with the same role may generate bogus signals for valid queries that are unique to these users but not coherent with other queries in the transaction log associated with that role. Hence we apply User Behaviour Learning by Feature Extraction in which we analyze the individual behaviour of all the

Algorithm 5: SGREA

Data : A list **'patterns'** of Weighted Sequential Patterns along with the User Roles **'roleList'**, a mapping **dictSens** between the read and write operations and their respective sensitivities, **K**=sensitivity threshold

Result : Sets of Read and Write Rules for every user role

```
1  Initialization:
2  readRules = []
3  writeRules = []
4  for every pattern (p,r) in (patterns, roleList) do
5      i← 0
6      n ← length(p)
7      while i < n do
8          key ← p[i]
9          if dictSens[key] > K then
10             x ← 0
11             leftPart ← []
12             while x < i do
13                 if p[x] is a read operation then
14                     add p[x] to leftPart
15                 end
16             end
17             if length(leftPart) > 0 then
18                 rule = (leftPart → key , r)
19                 if rule not in readRules for role r then
20                     add rule to readRules
21                 end
22             end
23             x ← i + 1
24             rightPart ← []
25             while x < n do
26                 if p[x] is a write operation then
27                     add p[x] to rightPart
28                 end
29             end
30             if length(rightPart) > 0 then
31                 rule = (key → rightPart , r)
32                 if rule not in writeRules for r then
33                     add rule to writeRules
34                 end
35             end
36         end
37     end
38 end
```

users by learning the order in which the user accesses the relations from transaction logs that are unique to the user and later use it in the detection phase.

User Behaviour RAP Generator

User Behaviour RAP Generator utilizes the Relation access paths(RAPs) generated by the QTG to compute the Most Frequent Relation access path for all the unique users.

Definition 20(Most Frequent Relation access path(MFRAP)): **MFRAP** of a user with User ID 'u' denoted by $\alpha[u]$ is the relative order in which all the relations are accessed by that user. **MFRAP** is a string of the Table IDs and it is computed by considering the Relation access path in all the transactions committed by that user. For each position in **MFRAP**, the most frequent relation accessed at that position is considered.

Once we generate MFRAP for all the users, they are stored to be used later in detection phase for computing divergence of incoming query with other past queries of that user. *Figure 6.7 illustrates the computation of the most frequent relation access path. At every position, the most prominent Table ID is taken from all the RAPs.*

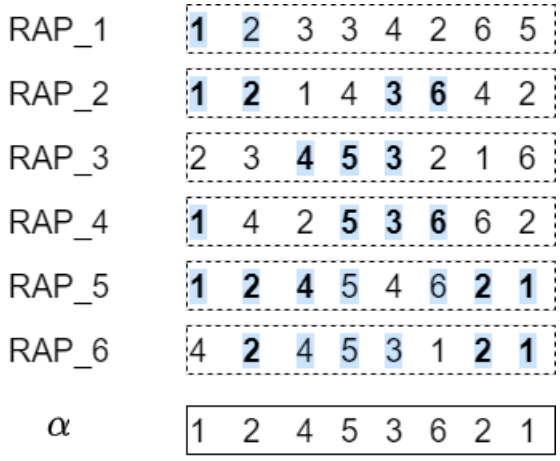


Figure 6.7: Most Frequent Relation access path

Algorithm 6 illustrates **User Behaviour RAP Generator(UBRAPG)**.The input to the algorithm includes a list of Relation access paths 'RAP' of all the transactions accomplished by a user 'u'. The algorithm results the Most Frequent Relation access path(α) for the user 'u'. In lines 3 to 8 we find the maximum length(which ultimately will be

the length of α) out of all the Relation access paths we have in the list RAP[]. For α we iterate through each position of all the RAPs, find the most frequent table accessed by using the dictionary count{} and append it to α as described in lines 9 to 16. Finally we convert α to string in line 18. *Table 6.24 illustrates the generated User Behaviour RAPs for 4 unique user ids taken.*

Algorithm 6: UBRAPG

```
1  Input: List of Relation access path 'RAP' of all the transactions performed by
     user 'u'
2  Output: MFRAP(α) for the user 'u' lenRAP ← []
3  α = []
4  for i in RAP do
5  |  lenRAP.append(length(i))
6  end
7  positions ← max(lenRAP)
8  for i in range(positions) do
9  |  count = {1:0, 2:0, 3:0, 4:0, 5:0, 6:0}
10 |  for j in RAP do
11 |  |  if length(j) >i then
12 |  |  |  count[j[i]] = count[j[i]] + 1
13 |  |  end
14 |  end
15 |  (key_max, val_max) ← max(count)
16 |  α.push(key_max)
17 end
18 α = toString(α)
```

$\alpha[U1]$	$\alpha[U2]$	$\alpha[U3]$	$\alpha[U4]$
'12453621'	'46521324'	'14235632'	'34212456'

Table 6.24: MFRAPs

6.2.2 Detection Phase

Detection phase is a concatenation of three components listed below in the increasing order of specificity:

1. Outlier Detection(Level 1)
2. Role Induced Rule level Detection(Level 2)
3. User Behaviour Analysis(Level 3)

The Detection Phase is indexed at these distinct Specification Levels which enhances the granularity of our approach for every rise in the level as illustrated by figure 6.8. Specification Level 1 involves ubiquitous detection at a global platform by performing Outlier Analysis to catch the presence of an outlier. The transaction is termed malicious if the existence of an outlier is confirmed. The control passes on to Specification Level 2 only in case an outlier is not detected. To reduce the false negative rate, role level behavioural check is conducted to determine the coherence of the new transaction with the previously mined rules. The comparison of the coherence count with a specific threshold lays of the foundation for the execution of Specification Level 3. A high coherence implies a valid transaction, else the control accelerates to next level. At level 3, User Behaviour is analysed by computing the divergence between the current user Relation access path and the Most Frequent Relation access path(MFRAP) of current user inherited from the learning module. The final decision is made from the value of divergence which bifurcates the transaction to be malicious or legitimate. At every level, all the computed parameters are compared against respective threshold constants as illustrated in Table 6.25.

Symbol	**Level**	**Description**
Δ_1	Level 1	Deviation Threshold
Δ_2	Level 2	Coherence Threshold
Δ_3	Level 3	Divergence Threshold

Table 6.25: Threshold Constants for the 3 levels

6.2.2.1 Specification Level 1: Outlier Detection

The fuzzy clusters generated the learning phase are used to calculate the global outlier. The incoming transaction query is parsed through the QTG and then processed in the Sequential Pattern Miner to generate the input sequence. The cluster with which the input sequence scores the maximum membership is taken for the computation of the Deviation Metric(Ψ).

Definition 21(Primary Cluster): It is the Cluster PC closest to the issued transaction T_i. As T_i beholds maximum membership with the cluster PC, all the transactions in PC possess the highest degree of similarity with the issued transaction T_i.

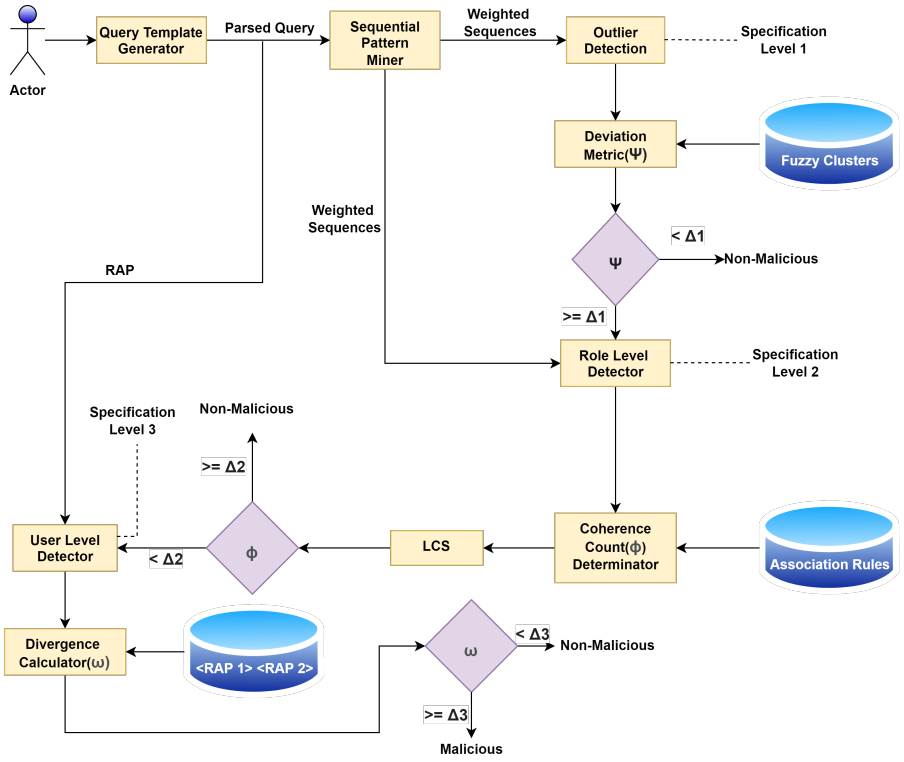


Figure 6.8: Detection Phase Architecture

Definition 22(Deviation Metric): The minimum distance between the issued transaction and the cluster yielding the maximum membership(PC) is called the **Deviation Metric**(Ψ).

The value of Deviation Metric(Ψ) is compared against the Deviation Threshold(Δ_1).

- If $\Psi >= \Delta_1$, the presence of an outlier is justified due to a considerable value of the deviation metric. Hence, the input transaction is alarmed as malicious and the detection process terminates at this stage.
- If $\Psi < \Delta_1$, the input transaction is not taken as an outlier due to **low** value of the deviation metric, implying a high affinity towards the legitimate transactions. However, to reduce the false negative rate, behavioural check is made at role level as illustrated in Specification Level 2.

6.2.2.2 Specification Level 2: Role Induced Rule Level Detection

The issued transaction query is used to produce read and write sequential patterns which are utilized by the Role Induced Detector along with the association rules generated in

the learning phase to compute Coherence Count($\not\in$).

Coherence Count($\not\in$) Determinator

Coherence Count is a score laid on the count of data item dependencies that a transaction satisfies. A rule matching technique is used to compute the Coherence Count for a query issued by a role for detection phase by utilizing the concept of LCS(Longest Common Subsequence) [106, 107]. For any given rule, the matching is done independently for both the antecedent and the consequent.

Algorithm 7 illustrates the working of Longest Common Subsequence(LCS). The input comprises of 2 strings denoted as X and Y respectively. Through a recursive implementation, the length of the largest subsequence common to both the strings is computed. Algorithm 7 is inherited in Algorithm 8 for the computation of rule coherence.

Algorithm 7: LCS

```
1  Input: 2 Sequences X and Y
2  Output: Length of the lcs in X and Y
3  n ← length(X)
4  m ← length(Y)
5  if m == 0 or n == 0 then
6  |  return 0
7  end
8  if X[0] == Y[0] then
9  |  return 1 + LCS(X[1 :] , Y[1 :])
10 else
11 |  return max(LCS(X[1 :] , Y) , LCS(X , Y[1 :]))
12 end
```

Algorithm 8 elucidates the Coherence Count Determinator(CCD) for a particular sequence 'P' with respect to the previously mined read and write rules for the same role 'r' as shown by lines 3 and 4. Lines 5 to 8 follow the segregation of antecendents and consequents of the existing read and write rules. Line 10 involves the extraction of the new read rules and new write rules from the input sequence by using the sensitivity Galvanized Rule Extraction Algorithm(SGREA) as illustrated by Algorithm 5. The final Coherence Count is the overall summation of the lengths of the longest common subsequences of the

left and right components of every read and write rule with the respective left and right components of the previously mined rules as shown in lines 15 to 35.

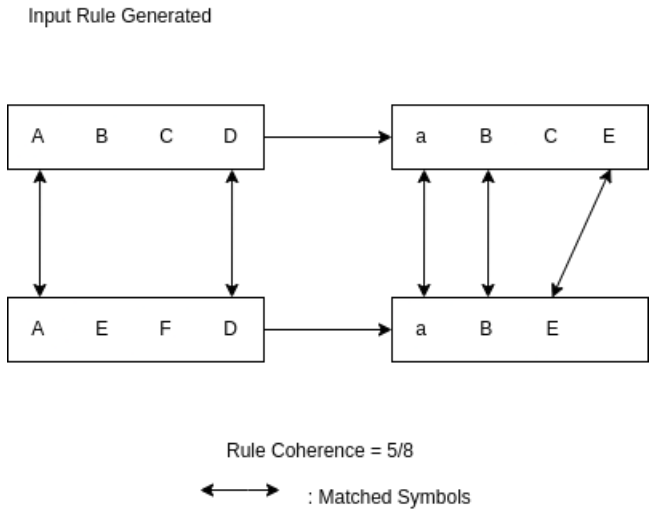


Figure 6.9: Coherence Count Demonstration

Figure 6.9 illustrates the working of Longest Common Subsequence for the computation of Rule Coherence. The rule fetched from the incoming transaction is 'ABCD → aBCE' which is matched against the rule 'AEFD → aBE'. The total count of matched symbols is 5 as follows:
Number of symbols matched on the LHS: 2{A, B}
Number of symbols matched on the RHS: 3{a , B , E}
Hence Coherence Count = 5/8 = 0.625

The value of Coherence Count($\not\epsilon$) is compared against the Coherence Threshold(Δ_2).

- If $\not\epsilon >= \Delta_2$, the input transaction clarifies the RBAC policy and obtains a high coherence with the preciously mined association rules. Hence, the input transaction qualifies to be legitimate and the detection process halts.
- If $\not\epsilon < \Delta_2$, the transaction is further examined for User Behaviour Analysis to reduce the false positive rate.

6.2.2.3 Specification Level 3: User Behaviour Analyser

At the user level, the divergence between this incoming RAP and the previously mined Most Frequent Relation access path for that user(in learning phase) is computed by using

Algorithm 8: Coherence Count Determinator

```
1  Input: A sequence 'P', a user role 'r'
2  Output: Coherence Count(∉) of 'P' with the read and write rules
3  readR ← readRules[r]
4  writeR ← writeRules[r]
5  lRd ← LHS(readR)
6  rRd ← RHS(readR)
7  lWrt ← LHS(writeR)
8  rWrt ← RHS(writeR)
9  ∉ ← 0
10 newReadRules , newWriteRules = SGREA(P , r)
11 leftRead = LHS(newReadRules)
12 rightRead = RHS(newReadRules)
13 leftWrite = LHS(newWriteRules)
14 rightWrite = RHS(newWriteRules)
15 for i in length(leftRead) do
16     for j in length(lRd) do
17         scoreL ← LCS(leftRead[i] , lRd[j])
18         scoreR ← LCS(rightRead[i] , rRd[j])
19         if scoreL ! = 0 and scoreR ! = 0 then
20             var = (scoreL+scoreR)
21             len = length(leftRead[i]) + length(rightRead[i])
22             var = var/ len
23             ∉ = ∉ + var
24         end
25     end
26 end
27 for i in length(leftWrite) do
28     for j in length(lWrt) do
29         scoreL ← LCS(leftWrite[i] , lWrt[j])
30         scoreR ← LCS(rightWrite[i] , rWrt[j])
31         if scoreL ! = 0 and scoreR ! = 0 then
32             var = (scoreL+scoreR)
33             len = length(leftWrite[i]) + length(rightWrite[i])
34             var = var/ len
35             ∉ = ∉ + var
36         end
37     end
38 end
```

Levenshtein distance.

Divergence(ω) Calculation

Given two MFRAPs,

P =$\alpha[u]$ and

Q =$\alpha'[u]$

For comparing the Most Frequent Relation access paths, α and α', we utilize **Levenshtein distance** to measure the number of changes required to transform one RAP to another. Algorithm 9 represents computation of Levenshtein distance(LD) for X and Y implemented through a recursive approach. The algorithm yields the minimum number of changes in the form of insertions, deletions or substitutions [108] required to make the 2 RAPs exactly similar.

Algorithm 9: LD

```
1  Input: 2 RAPs X and Y
2  Output: Levenshtein distance of X and Y n ← length(X) m ← length(Y)
3  if m == 0 then
4  |  return n
5  end
6  if n == 0 then
7  |  return m
8  end
9  if X[0] == Y[0] then
10 |  return LD(X[1 :] , Y[1 :])
11 else
12 |  op1 = LD(X[1 :] , Y)
13 |  op2 = LD(X , Y[1 :])
14 |  op3 = LD(X[1 :] , Y[1 :])
15 |  return 1 + minimum(op1 , op2, op3)
16 end
```

For Example, $\alpha[u]$ = '12453621' and $\alpha'[u]$ = '42431652', the result of Levenshtein distance

for $\alpha[u]$ and $\alpha'[u]$ is 5. Hence, the value of ω can be computed as:

$$\omega = \frac{LD(\alpha[u], \alpha'[u])}{maximum(len(\alpha[u], \alpha'[u]))} \tag{6.22}$$

Hence, ω = 5/8 = 0.625.

The value of Divergence(ω) is compared against the Divergence Threshold(Δ_3).

- If $\omega >= \Delta_3$, the input transaction is classified as malicious due to anomalous behaviour at user level resulting in high divergence.
- If $\omega < \Delta_3$, the transaction is labelled as legitimate after the entire three tier validation.

Algorithm 10: The Detection Phase

1 **Input:** An input sequence 'P', a user role 'r', user id 'u', Current Relation access path($\alpha'[u]$) for the user u
2 **Output:** Classification of the input sequence as being malicious or legitimate.
3 fuzzyOutput = FCM.predict(P)
4 index = argmax(fuzzyOutput[0])
5 $\psi \leftarrow fuzzyOutput[2][index]$
6 **if** $\psi >= \Delta_1$ **then**
7 | return "Malicious Transaction"
8 | **quit**(The Detection Phase)
9 **end**
10 $\not\psi \leftarrow CCD(P, r)$
11 **if** $\not\psi >= \Delta_2$ **then**
12 | return "Legitimate Transaction"
13 | **quit**(The Detection Phase)
14 **end**
15 $\omega \leftarrow$ LD($\alpha[u]$,$\alpha'[u]$)/maximum(len($\alpha[u]$,$\alpha'[u]$))
16 **if** $\omega >= \Delta_3$ **then**
17 | return "Malicious Transaction"
18 **else**
19 | return "Legitimate Transaction"
20 **end**

Algorithm 10 illustrates the mechanism behind the detection phase which involves classification of transactions by employing a three tier security check. At specification level 1, we compute the Deviation Score(Ψ) by finding the distance from the cluster that beholds maximum membership with the input sequence as illustrated by Lines 3 to 5. The function **FCM.predict(P)** in line 3 returns the fractional memberships(**fuzzyOutput**[0]) of the input transaction with the previously mined fuzzy clusters along with the respective distances(**fuzzyOutput**[2]) from each clusters. The cluster with the maximum membership(**index**) is selected and the corresponding distance(**fuzzyOutput**[2][$index$]) is taken as the value for deviation score. If the value of Deviation Score is greater than or equal to Deviation Threshold(Δ_1) then it is detected as an outlier and hence malicious.This leads

to the termination of the algorithm as shown in line 8.If the transaction is termed as Non-malicious, we move to the 2nd Specification Level in which the computation of Coherence Count($\not\in$) is done by using the Algorithm 8, Coherence Count Determinator**(CCD)** as depicted in line 10. If the value of Coherence Count($\not\in$) is greater than or equal to Coherence Threshold(Δ_2) then the transaction is classified as legitimate. In the other case we move towards the 3rd and final Specification Level where we calculate divergence(Ω). Lines 15 depicts the calculation of divergence between 2 RAPs by computing the Levenshtein distance(LD) . between $\alpha[u]$ and $\alpha'[u]$. Finally if Divergence(ω) is greater than or equal to Divergence Threshold(Δ_3) then transaction is detected to be malicious else it is classified as legitimate.

6.2.3 Results and Experiments

Various experiments were carried out to analyze the proposed approach and access accuracy on the dataset developed using TPC-C benchmark [63]. The final classification is awarded on the basis of Deviation Score, Rule Coherence and User Level Divergence. The overall analysis is done at multiple thresholds of support and sensitivity both at the user and role level. As there is no specific dataset existing in this domain for the evaluation, the dataset was generated synthetically keeping in track the RBAC policy and user behaviour for different attribute domains. Two different sets of transactions were created, one containing malicious transactions and the other containing normal transactions. To verify the adherence of generated queries with TPC-C standards, the fraction of transactions in which a particular attribute is present in our dataset is calculated which can be depicted from Table 6.26 and Table 6.27. For example, read operation on DeptNo (G) is done on 32.10% of queries, which can also be written as 0.3211 in the form of fraction.
The calculations were conducted in terms of accuracy, precision, recall, F1 score, execution time, the number of patterns and rules. Execution time is the total time taken to transact our algorithm and classify the incoming transaction.
As justified in Section 3, users pertaining to a particular role tend to have congruent access patterns. This implies the fact that a set of attributes has limited dependency on the other set of attributes. This property is used for the partitioning of the incoming transactions as being malicious or safe.
Figure 6.10 shows the plot of total sequential patterns obtained from the database trans-

Attribute	Fraction
G	0.3211
F	0.3612
H	0.2575
I	0.0569
A	0.301
B	0.1806
C	0.2007
D	0.1204
E	0.2609

Table 6.26: Fraction of transaction Containing Read operation

Attribute	Fraction
i	0.2408
g	0.0803
e	0.1806
d	0.2207
f	0.1204
a	0.0803
b	0.1003
c	0.0803
h	0.1405

Table 6.27: Fraction of transaction Containing Write operation

actions against different values of weighted support threshold β_{wsup}. As the support threshold increases from 0.1 to 0.5, the number of patterns obtained decreases from 175 to 8 which is evident as when the support threshold increases, the weighted support w_{sup} required for the patterns to be retained increases which eventually decreases the number of sequential patterns generated. *Table 6.28 depicts the exact values of number of patterns obtained for different values of support threshold varying from 0.1 to 0.5.*

Weighted Support Threshold	Sequential Patterns
0.1	175
0.2	50
0.3	13
0.4	10
0.5	8

Table 6.28: Variation in number of patterns with support threshold(with weights)

Figure 6.11 depicts the number of patterns for each user role. The size of the bubble indicates the number of patterns mined for that role. The roles are clusters of related users based on their data access behavior patterns. *The corresponding percentages are*

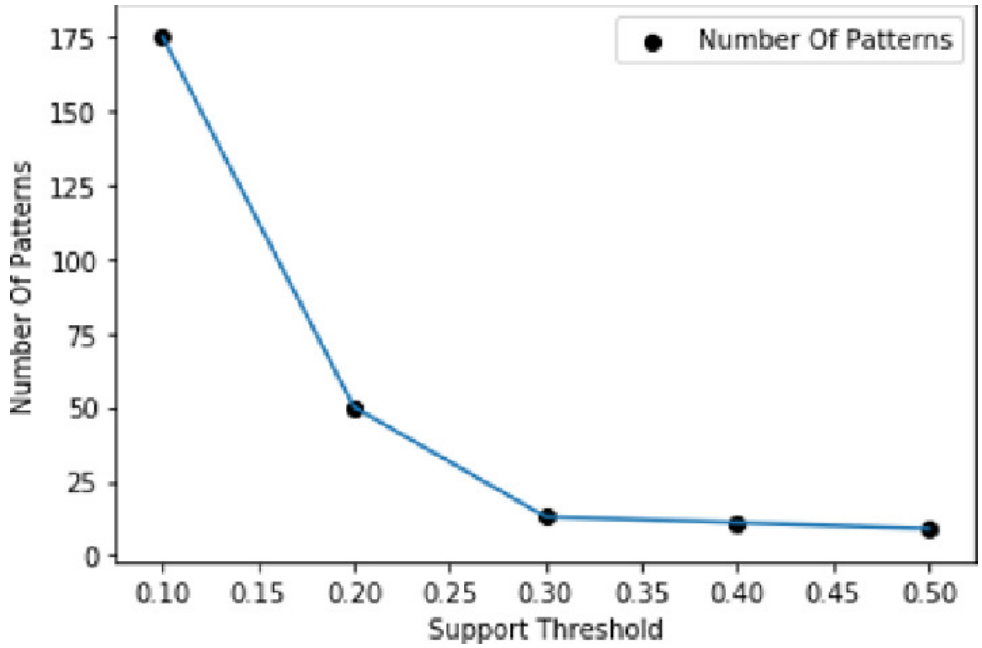


Figure 6.10: Line graph exhibiting variation of number of patterns obtained with threshold support

mentioned in Table 6.29 and a visual is shown by the pie-chart in Figure 6.13

Role ID	**Patterns**	**Percentage**
R1	30	10.2
R2	5	1.70
R3	162	55.10
R4	75	25.51
R5	22	7.48

Table 6.29: Number of patterns and their percentages for each role

Figure 6.12 elucidates the decrease in the total execution time of the model as the support threshold increases. The execution time decreases as the number of patterns and hence rules mined also decreases with the rise in support threshold as depicted in Figure 6.10 which is the consequence of the fact that due to reduction in the amount of patterns and rules mined, the time needed to compute the Intrusion Metric also decreases.

A Confusion Matrix provides a visual representation of the performance of an algorithm. The Confusion Matrix in Table 6.31 gives a distinction between the performance parameters by using the notations Malicious) and Legitimate.The percentage of legitimate transactions being predicted as miscellaneous is rather high . This problem arises because the dataset has been synthetically generated and certain attributes have been randomly

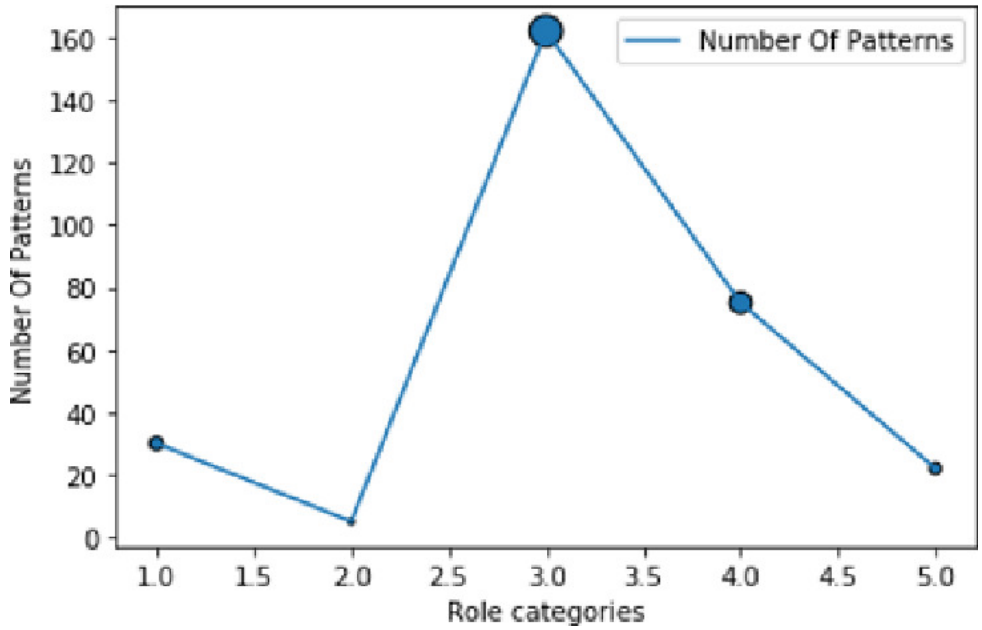


Figure 6.11: Line graph exhibiting number of patterns obtained for different user roles

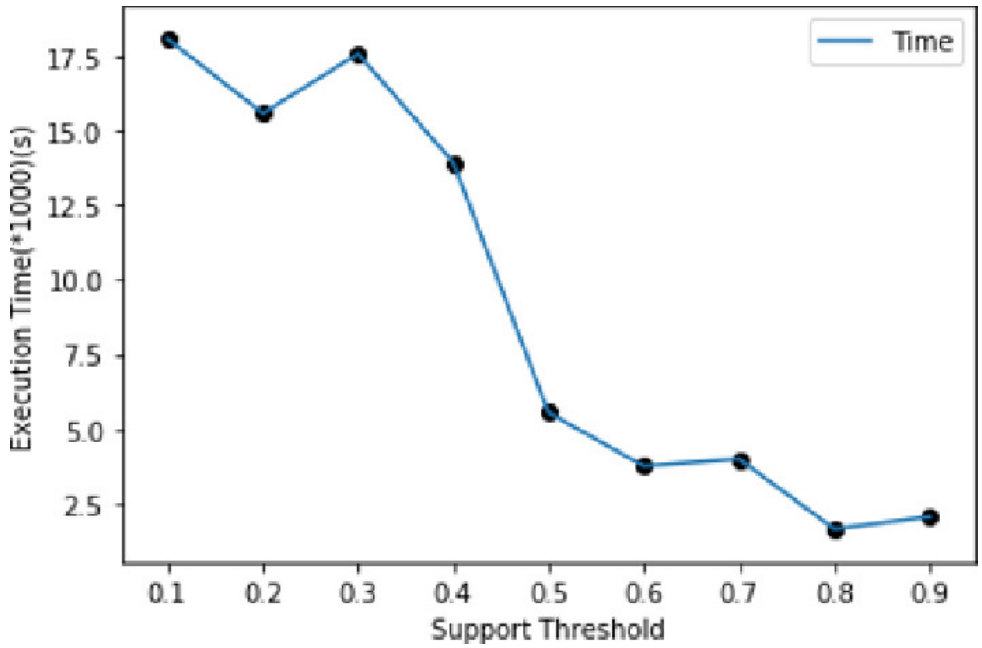


Figure 6.12: Line graph exhibiting the Execution Time with various Support Thresholds

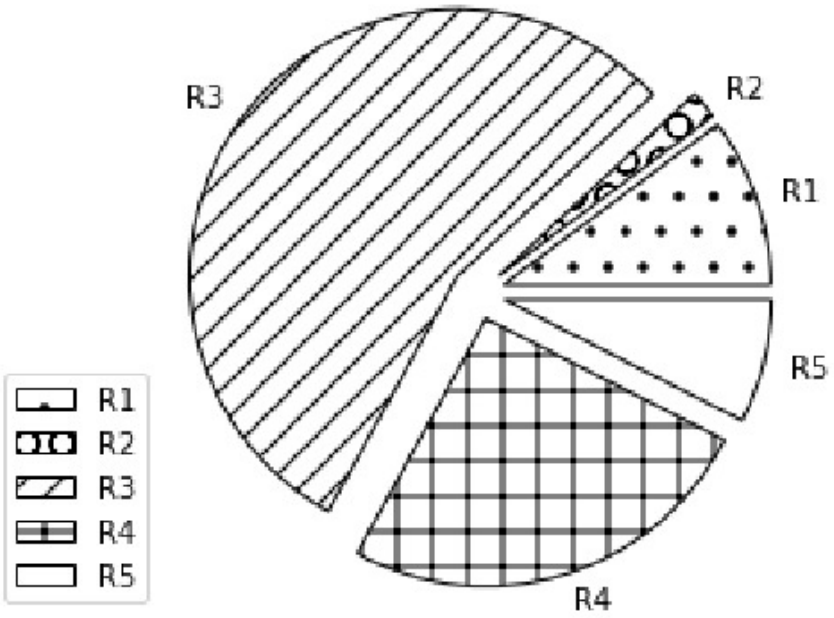


Figure 6.13: Pie Chart for the percentage of patterns per role

replaced. The replaced values may be a novelty within the dataset and due to this some occurrences of such dependencies are incorrectly classified which is not likely to occur in the real-world. Additionally, some records may just be exceptions. However, if provided with the real-world data, the model is capable to correctly classify the transactions.
Table 6.30 shows the evaluated dynamic sensitivity of read and write operations on all attributes as described in section 3.

Attribute	**Read Operation Sensitivity**	**Write Operation Sensitivity**
A	0.1867	0.8429
B	0.4911	0.696
C	0.4552	0.8429
D	0.691	0.1529
E	0.2545	0.3838
F	0.0999	0.67
G	0.156	0.8273
H	0.2606	0.84
I	0.9	0.2565

Table 6.30: Dynamic Sensitivity of Read and Write operations on all attributes

For a specific support threshold of 0.5 the precision, recall and F1 score have been calculated and are displayed in Table 6.32, which provides a detailed overview of the values of these metrics that have been obtained. Figure 6.14, Figure 6.15 and Figure 6.16 illustrate the variation in Precision, Recall, F1-Score with the number(#) of transactions at the

		Predicted	
		Malicious	Legitimate
Actual	Malicious	688	24
	Legitimate	112	176

Table 6.31: Confusion Matrix of our proposed OIUWSPM

same support threshold(0.5).

# of Transactions	Precision	Recall	F1-Score
400	0.75	0.75	0.75
500	0.86	0.85	0.85
600	0.84	0.93	0.88
700	0.86	0.94	0.90
800	0.87	0.95	0.91
900	0.85	0.96	0.90
1000	0.86	0.96	0.91

Table 6.32: Variation in performance measures with the number of transactions

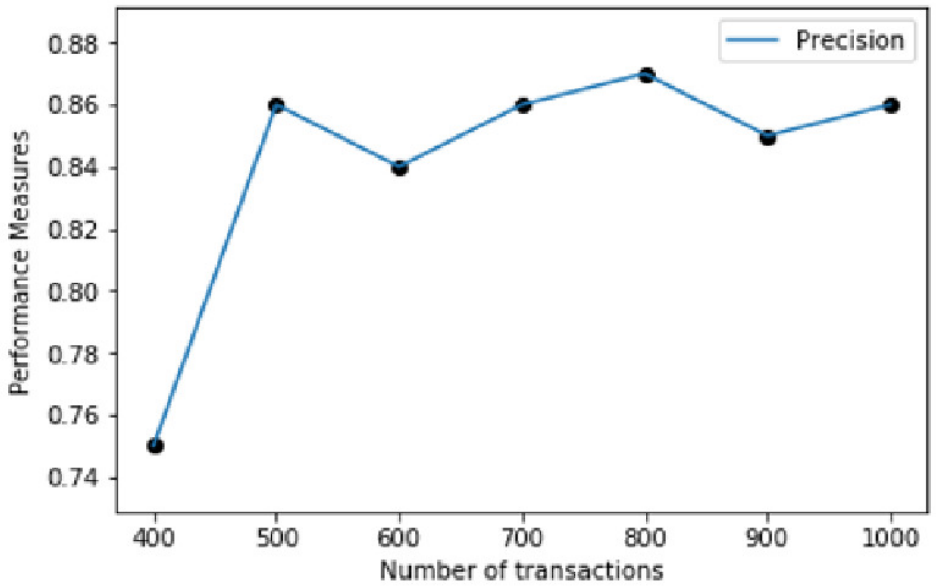


Figure 6.14: Variation in Precision with the number of transactions

As evident from Figure 6.14 and Figure 6.15 the Precision increases with the number of transactions and reaches a maximum value of 0.86. Similarly the value of recall rises to 0.96 accounting for a maximum of 0.91 for the F1-Score. This trend can be justified by the increase in the number of sequential patterns with increase in number of transactions, hence more number of rules can be consistently mined, thereby making it more agile. Due to the increase in the values of Precision and Recall, the value of F1-Score also increases

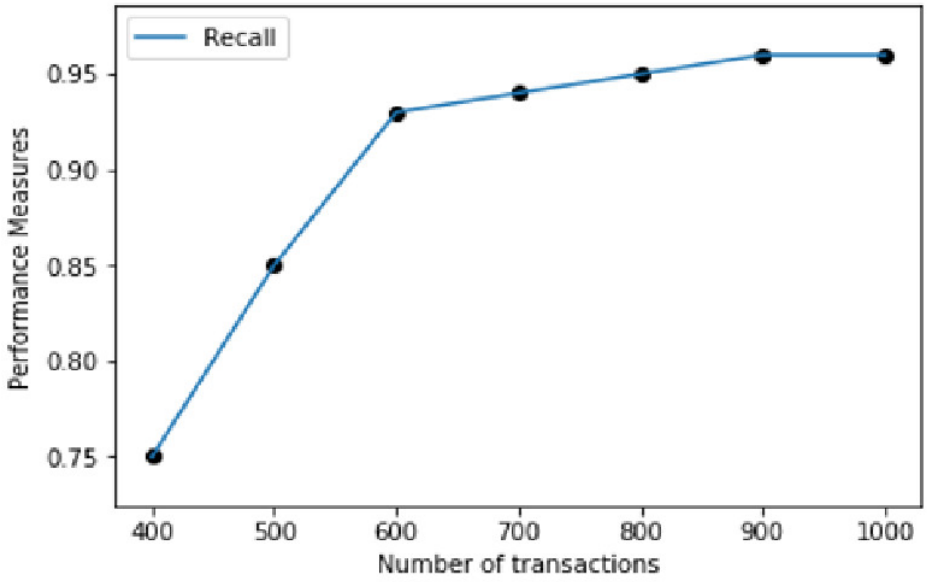


Figure 6.15: Variation in Recall with the number of transactions

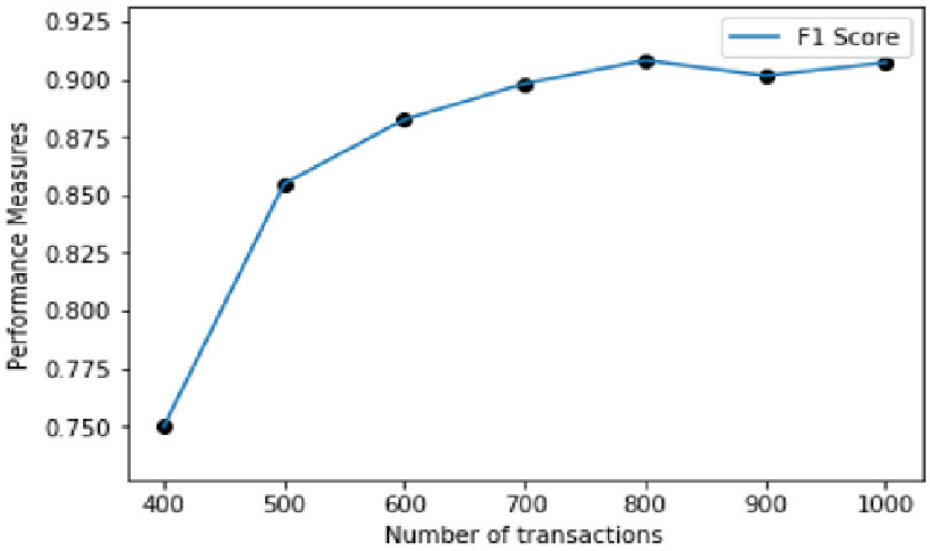


Figure 6.16: Variation in F1-Score with the number of transactions

Reference	Elements Analysed						
	Command Syntax	**Scalability**	**RBAC**	**Dynamic sensitivity**	**User Behaviour Analysis**	**Outlier Detection**	**Performance Measures**
Hu & Panda [109]	✓	✓	✗	✗	✗	✗	Recall = 0.9 Precision = 0.64
Srivastava et al. [110]	✓	✗	✗	✗	✗	✗	Precision = 0.8
Kamra et al. [31]	✓	✓	✓	✗	✓	✓	Recall = 0.63 Precision = 0.77
Hashemi et al. [87]	✓	✗	✗	✗	✗	✗	Recall = 0.90 Precision = 0.75
Kundu et al. [40]	✓	✓	✗	✗	✗	✗	Precision = 0.945
Ronao & Cho [55]	✓	✓	✓	✗	✗	✗	Recall = 0.95 Precision = 0.89
Kim & Cho [52]	✓	✓	✓	✗	✗	✗	Recall = 0.887 Accuracy of 0.933
Bu & Cho [88]	✓	✓	✓	✗	✗	✗	Recall = 0.93 Precision = 0.92
Bu et al. [103]	✓	✓	✓	✗	✗	✗	Recall = 0.85 Precision = 0.87
OIUWSPM (Our Approach)	✓	✓	✓	✓	✓	✓	Recall =0.96 Precision = 0.86

Table 6.33: Performance Comparison of OIUWSPM

as depicted in Figure 6.16.

Table 6.33 elucidates the bifurcation in the performance metrics of several techniques focusing on the algorithms utilized in their implementation, elements analyzed and their overall classification measures including Precision and Recall. The highest value of these measures has been compared for the systems. In terms of performance, our proposed approach(OIUWSPM) has a comparatively high efficiency attributed to its accuracy which is proportional to the analysis conducted on account of Dynamic Sensitivity, Outlier Detection, Coherence Count, User Level Divergence.

6.3 Summary of the chapter

In this chapter we developed 2 novel methods. Firstly We have devised a Trust factor based User behavior intrusion detection system (TFUBID), a two-phase database intrusion detection system that clusters users based on their attribute access patterns and the types of queries that they implement. The efficacy of our model lies in its ability to prevent illegitimate retrieval and modification of most sensitive data elements called IDA (Integral Data Attributes) and the attributes that are directly correlated with them. To avoid data breaches, TFUBID uses sequential pattern mining to ensure that each user follows a particular query pattern for accessing IDAs, i.e. the user associates themselves with his regular access behaviour. Any deviation from the regular arrangement may lead to depreciation of the trust factor of the user's profile, indicating the user's malicious intent. Our extensive evaluation shows that TFUBID is able to outperform other approaches in terms of various performance measures. In future, to make this model more robust, we would like to incorporate the temporal dimension of data access along with patterns mined from inter-query dependencies. We are also planning to introduce more rigorous measures to account for the sensitivity of operations which has been proven to directly impact the performance. Secondly, we proposed a novel approach for Intrusion Detection System for organizations synthesized with RBAC, which indispensably performs training through weighted sequential patterns and detection by means of a hierarchical analysis through an increase in granularity of the data. The proposed approach draws attention towards Role Level Analysis empowered by User Level Analysis and authorized by the notion of

Dynamic Sensitivity thereby reducing the number of false positives and accounting for the increase in performance metrics including accuracy. A Comprehensive experimentation on the synthetically generated dataset elucidated the fact that our proposed approach is efficient to determine malicious and legitimate transactions. Our future work will emphasize on a highly improved feature collection for profiling user behaviour. Different techniques for outlier detection involving numerous clustering algorithms can further be investigated. The implementation and evaluation of the proposed approach on real data can also serve as a supplementary accomplishment. Our thesis file will be concluded in the next chapter.

Chapter 7

CONCLUSION AND FUTURE SCOPE

Organizations today are employing databases on a large scale to store data essential for their functioning. Therefore, in current times, with data security being recognized as an irrefutable requirement within an organization, the importance of intrusion detection system has grown manifold. Malicious access and modifications of the databases may lead to adverse financial and legal implications. In recent years, security researchers have focused on detecting abuse of access privileges by employees of an organization. Identifying threats from insiders is hard because they are aware of the organization of the database in addition to having authorized access privileges. Therefore, Identification of outsider attacks as well as misuse of database privileges by authorized entities has been a primary requirement in modern intrusion detection systems.

This thesis focuses on current security issues in databases which include both insider and outsider attacks and discusses several limitations of existing DIDS systems. The main objective of this thesis is to prevent insider and outsider attacks on databases. This prevention is dynamic and adaptive to user access parameters. So, there is a need for a DIDS system which is capable of handling privilege abuses along with attacks by outsiders. The proposed solution in this thesis is composed of a fuzzy association data dependency rule miner, sequential pattern mining and user behavior analysis, Expectation Maximization Clustering and Sequential Pattern Mining, using meta-heuristic clustering by utilizing PSO and pattern mining by BIDE. The proposed solutions for DIDS will help organizations to ensure security to their databases from outsider and privilege abuse by members of the organization.

7.1 Summary of the contributions

This research work has led to several important contributions in the following directions:

- **A Systematic and comprehensive survey on DIDS:** A systematic survey is conducted in order to classify various approaches for detecting intrusion in databases.The work identifies open research questions and challenges, by methodically comparing existing strategies to combat malicious transactions in a database system, and also provides an insight into the applications of DIDSs.

- **FADDRM Model:** This model introduces a new data mining-based approach Fuzzy Association Data Dependency Rule Miner (FADDRM) for detecting malicious transactions. This data mining technique uses three phases which were based upon the sequential discovery of relevant patterns, the generation of a set of sequence and the generation of dependency rules

- **EMSPM Model:** This approach, Expectation Maximization Clustering and Sequential Pattern Mining (EMSPM) is a database intrusion detection system consisting of an initial training phase, rule mining followed by clustering and then classification. It computes the maximum of membership probabilities and following the calculation of the Cluster Deviation Index, classifies the transactions as malicious or non-malicious.

- **BPSOMQD Model:** This model combines the capabilities of BIDE and particle swarm optimization algorithms and presents a novel intrusion detection and prevention technique that accommodates the behaviour of user at individual as well as role level.The main idea behind this approach is to develop an intrusion detection system which is decoupled from the existing database management system so as to function independently in various environments without affecting the internal structure of the system. It aims to identify intrusion attacks and prevent execution of malicious transactions which could compromise the integrity and security of the database.

- **FPGWWO Model:** We have proposed a Database Intrusion Detection System (DIDS) to prevent unauthorised access from changing critical user details. Our method (FPGWWO) can identify threats from both within and outside the system

by incorporating a frequent sequential pattern mining algorithm and a hybrid meta-heuristic clustering using modified Grey Wolf optimization and Whale Optimization Algorithm (mhGW-WOA).

- **TFUBID Model:** We propose a two-phase intrusion detection and prevention model TFUBID (Trust factor based user behaviour analysis for database intrusion detection) that clusters users based on similarity of their data access patterns and the types of queries submitted by them, i.e. our model tracks the access patterns of all users and classifies them as normal or malicious. The superiority of our model lies in its ability to prevent unauthorised retrieving and modification of most sensitive data elements referred to as IDAs or Integral Data Attribute.

- **OIUWSPM Model:** A novel approach Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining(OIUWSPM), which performs training through weighted sequential patterns and detection by means of a hierarchical analysis through an increase in specificity of the data. The Outlier Detection module generates clusters based on the syntactic characteristics of transactions. Role level analysis is based upon dynamic usage of attributes local to every role domain. User profiling is structured by stockpiling legitimate transactions committed by the user.

7.2 Limitations of the proposed model

Rule classifier combined with role profile clustering is a good technique to prevent databases from malicious transaction attacks. This thesis introduced novel models which cover most of the security issues, especially insider threats. The result analysis shows improved performance of the proposed model. But, still, there are some limitations that are required to overcome in the future.

- Rule classifier used in the models discussed in this thesis only considered the sequence of queries in the transaction and not the amount of the data values that are being changed.

- Sequential pattern mining used in the models in this thesis mines more patterns as compared to closed and maximal frequent patterns. So, usage of maximal frequent

patterns or utility frequent patterns might have increased the efficiency.

- The risk analysis part is not considered in the proposed models.

7.3 Future Scope

To overcome the shortcomings present in the DIDS and to improve it several models were proposed. But several security issues that have been already discussed need to be taken care of. This thesis introduced novel models which are based on the behavior of users to address the mentioned shortcomings. But some new directions will be possible in the future which will improve the reliability of the model.

The following new directions will be possible in the near future:

- Along with the sequence of queries, values of data items will also be considered while mining frequent sequential patterns. This will increase the efficiency of the Detection of malicious transactions and will decrease the false-positive rate.
- Maximal frequent patterns or Utility frequent patterns will be utilized to mine sequential rules for the classification to increase the rate of true positives.
- Recent and better meta-heuristic optimization algorithms and their hybrids will be used to cluster the role profiles to enhance the robustness of the system to tackle privilege abuse by members of the organization.
- Risk analysis will be performed to enhance the accuracy of the model.

Appendix A

PUBLISHED WORK

JOURNAL RESEARCH PAPERS PUBLISHED

1. **Indu Singh, Rajni Jindal**, " Expectation maximization clustering and sequential pattern mining based approach for detecting intrusive transactions in databases", Multimedia Tools and Applications Journal, Vol-80, pp.27649-27681, **Springer 2021**. https://doi.org/10.1007/s11042-021-10786-3 , 2022 **(SCI-E Indexed)**

2. **Rajni Jindal, Indu Singh**, "Detecting malicious transactions in database using hybrid metaheuristic clustering and Frequent Sequential Pattern Mining", Cluster Computing Journal, **Springer 2022**.DOI 10.1007/s10586-022-03622-2 **(SCI-E Indexed)**.

3. **Rajni Jindal, Indu Singh** "A Survey on Database Intrusion Detection : Approaches, challenges and Application", International Journal of Intelligent Engineering Informatics **(IJIEI)** , Vol-7, Issue-6, pp.559-592, **2019**. DOI 10.1504/IJIEI.2019.104565. **(E-SCI Indexed)**

CONFERENCE PAPERS PUBLISHED

1. **Indu Singh, Rajni Jindal**, " Detecting Malicious Transactions using Fuzzy Association Rule Mining", Proceedings of 5th **IEEE** International Conference on Eco-friendly Computing and Communication Systems (ICECCS-2016), pp. 79-83, 8-9th December-2016, MANIT Bhopal, India. DOI 10.1109/Eco-friendly.2016.7893246

2. **Rajni Jindal, Indu Singh**, " Detection of Malicious Transactions using Frequent Closed Sequential Pattern Mining and Modified Particle Swarm Optimization Clustering", Proceedings of 6th **IEEE** International Conference for convergence

in Technology (I2CT-2021), pp. 1-8, 2-4th April-2021, Maharashtra, India. DOI 10.1109/I2CT51068.2021.9418217.

COMMUNICATED JOURNAL RESEARCH PAPERS

1. **Indu Singh, Rajni Jindal**, "Trust factor based analysis of user behaviour using sequential pattern mining for detecting intrusive transactions in databases", (communicated). **(SCI-E Indexed)**

2. **Indu Singh, Rajni Jindal**, "Outlier based Intrusion Detection in Databases for User Behaviour Analysis using Weighted Sequential Pattern Mining", (communicated). **(SCI-E Indexed)**

Bibliography

[1] K. Medhat, R. A. Ramadan, and I. Talkhan, "Towards distributed layered intrusion detection system for large scale wireless sensor networks," *International Journal of Intelligent Engineering Informatics*, vol. 5, no. 1, pp. 1–28, 2017.

[2] K. K. Wankhade and K. C. Jondhale, "An ensemble clustering method for intrusion detection," *International Journal of Intelligent Engineering Informatics*, vol. 7, no. 2-3, pp. 112–140, 2019.

[3] V. Kumar, J. Srivastava, and A. Lazarevic, "Managing cyber threats: issues, approaches, and challenges," 2006.

[4] G. M. Nazer and A. A. L. Selvakumar, "Current intrusion detection techniques in information technology-a detailed analysis," *European Journal of Scientific Research*, vol. 65, no. 4, pp. 611–624, 2011.

[5] R. J. Santos, J. Bernardino, and M. Vieira, "Approaches and challenges in database intrusion detection," *ACM Sigmod Record*, vol. 43, no. 3, pp. 36–47, 2014.

[6] R. Sandhu, D. Ferraiolo, R. Kuhn *et al.*, "The nist model for role-based access control: towards a unified standard," in *ACM workshop on Role-based access control*, vol. 10, no. 344287.344301, 2000.

[7] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.

[8] E. Burtescu, "Database security-attacks and control methods," *Journal of applied quantitative methods*, vol. 4, no. 4, pp. 449–454, 2009.

[9] R. Bace and P. Mell, "Nist special publication on intrusion detection systems," Booz-allen and Hamilton Inc MCLEAN VA, Tech. Rep., 2001.

[10] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer networks*, vol. 31, no. 8, pp. 805–822, 1999.

[11] E. Bertino and R. Sandhu, "Database security-concepts, approaches, and challenges," *IEEE Transactions on Dependable and secure computing*, vol. 2, no. 1, pp. 2–19, 2005.

[12] E. Lundin and E. Jonsson, "Survey of intrusion detection research," Chalmers University of Technology, Tech. Rep., 2002.

[13] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[14] S. A. El_Rahman, "An efficient approach for dynamic signature recognition," *International Journal of Intelligent Engineering Informatics*, vol. 5, no. 2, pp. 167–190, 2017.

[15] G. Liepins and H. Vaccaro, "Detection of anomalous computer session activity," in *Proceedings of the 1989 IEEE Research in Security and Privacy Conference*, 1989.

[16] A. J. Hoglund, K. Hatonen, and A. S. Sorvari, "A computer host-based user anomaly detection system using the self-organizing map," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 5. IEEE, 2000, pp. 411–416.

[17] R. Talpade, G. Kim, and S. Khurana, "Nomad: Traffic-based network monitoring framework for anomaly detection," in *Proceedings IEEE International Symposium on Computers and Communications (Cat. No. PR00250)*. IEEE, 1999, pp. 442–451.

[18] T. F. Lunt, A. Tamaru, and F. Gillham, *A real-time intrusion-detection expert system (IDES)*. SRI International. Computer Science Laboratory, 1992.

[19] E. Bertino, E. Terzi, A. Kamra, and A. Vakali, "Intrusion detection in rbac-administered databases," in *21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE, 2005, pp. 10–pp.

[20] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Proceedings 14th annual computer security applications conference (Cat. No. 98EX217)*. IEEE, 1998, pp. 13–24.

[21] A. N. Karkera, "Design and implementation of a policy-based intrusion detection system: Generic intrusion detection model for a distributed network," Ph.D. dissertation, University of Florida, 2002.

[22] M. Chagarlamudi, B. Panda, and Y. Hu, "Insider threat in database systems: Preventing malicious users' activities in databases," in *2009 Sixth International Conference on Information Technology: New Generations*. IEEE, 2009, pp. 1616–1620.

[23] A. K. Jones and R. S. Sielken, "Computer system intrusion detection: A survey," 2000.

[24] K. Scarfone, P. Mell *et al.*, "Guide to intrusion detection and prevention systems (idps)," *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.

[25] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," in *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 711–716.

[26] G.-C. Lan, T.-P. Hong, and H.-Y. Lee, "An efficient approach for finding weighted sequential patterns from sequence databases," *Applied Intelligence*, vol. 41, no. 2, pp. 439–452, 2014.

[27] I. C. Education, "Database security: An essential guide — ibm," https://www.ibm.com/cloud/learn/database-security, 08 2019, (Accessed on 06/27/2022).

[28] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.

[29] C. Y. Chung, M. Gertz, and K. Levitt, "Demids: A misuse detection system for database systems," in *Working Conference on Integrity and Internal Control in Information Systems*. Springer, 1999, pp. 159–178.

[30] Y. Zhong, Z. Zhu, and X.-L. Qin, "A clustering method based on data queries and its application in database intrusion detection," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 4. IEEE, 2005, pp. 2096–2101.

[31] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *The VLDB Journal*, vol. 17, no. 5, pp. 1063–1077, 2008.

[32] S. Subudhi and S. Panigrahi, "Application of optics and ensemble learning for database intrusion detection," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[33] V. C. Lee, J. A. Stankovic, and S. H. Son, "Intrusion detection in real-time database systems via time signatures," in *Proceedings Sixth IEEE Real-Time Technology and Applications Symposium. RTAS 2000.* IEEE, 2000, pp. 124–133.

[34] S. Y. Lee, W. L. Low, and P. Y. Wong, "Learning fingerprints for a database intrusion detection system," in *European Symposium on Research in Computer Security.* Springer, 2002, pp. 264–279.

[35] C. Bockermann, M. Apel, and M. Meier, "Learning sql for database intrusion detection using context-sensitive modelling," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 2009, pp. 196–205.

[36] A. Srivastava, S. Sural, and A. K. Majumdar, "Weighted intra-transactional rule mining for database intrusion detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2006, pp. 611–620.

[37] J. Fonseca, M. Vieira, and H. Madeira, "Online detection of malicious data access using dbms auditing," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 1013–1020.

[38] A. Spalka and J. Lehnhardt, "A comprehensive approach to anomaly detection in relational databases," in *IFIP Annual Conference on Data and Applications Security and Privacy.* Springer, 2005, pp. 207–221.

[39] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya, "A data-centric approach to insider attack detection in database systems," in *International workshop on recent advances in intrusion detection*. Springer, 2010, pp. 382–401.

[40] A. Kundu, S. Sural, and A. K. Majumdar, "Database intrusion detection using sequence alignment," *International Journal of information security*, vol. 9, no. 3, pp. 179–191, 2010.

[41] U. P. Rao, G. Sahani, and D. R. Patel, "Machine learning proposed approach for detecting database intrusions in rbac enabled databases," in *2010 second international conference on computing, communication and networking technologies*. IEEE, 2010, pp. 1–4.

[42] M. Sohrabi, M. M. Javidi, and S. Hashemi, "Detecting intrusion transactions in database systems: a novel approach," *Journal of Intelligent Information Systems*, vol. 42, no. 3, pp. 619–644, 2014.

[43] U. P. Rao and N. K. Singh, "Detection of privilege abuse in rbac administered database," in *Science and Information Conference*. Springer, 2014, pp. 57–76.

[44] Y. Sun, H. Xu, E. Bertino, and C. Sun, "A data-driven evaluation for insider threats," *Data Science and Engineering*, vol. 1, no. 2, pp. 73–85, 2016.

[45] Y. Chen and B. Malin, "Detection of anomalous insiders in collaborative environments via relational analysis of access logs," in *Proceedings of the first ACM conference on Data and application security and privacy*, 2011, pp. 63–74.

[46] K. Viet, B. Panda, and Y. Hu, "Detecting collaborative insider attacks in information systems," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2012, pp. 502–507.

[47] S. R. Hussain, A. M. Sallam, and E. Bertino, "Detanom: Detecting anomalous database transactions by insiders," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, 2015, pp. 25–35.

[48] A. Sallam, D. Fadolalkarim, E. Bertino, and Q. Xiao, "Data and syntax centric anomaly detection for relational databases," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 6, pp. 231–239, 2016.

[49] A. Sallam and E. Bertino, "Detection of temporal insider threats to relational databases," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2017, pp. 406–415.

[50] A. Sallam, E. Bertino, S. R. Hussain, D. Landers, R. M. Lefler, and D. Steiner, "Dbsafe—an anomaly detection system to protect databases from exfiltration attempts," *IEEE Systems Journal*, vol. 11, no. 2, pp. 483–493, 2015.

[51] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[52] T.-Y. Kim and S.-B. Cho, "Cnn-lstm neural networks for anomalous database intrusion detection in rbac-administered model," in *International Conference on Neural Information Processing*. Springer, 2019, pp. 131–139.

[53] S. Panigrahi, S. Sural, and A. K. Majumdar, "Two-stage database intrusion detection by combining multiple evidence and belief update," *Information Systems Frontiers*, vol. 15, no. 1, pp. 35–53, 2013.

[54] C. Chen, A. Liaw, L. Breiman *et al.*, "Using random forest to learn imbalanced data," *University of California, Berkeley*, vol. 110, no. 1-12, p. 24, 2004.

[55] C. A. Ronao and S.-B. Cho, "Anomalous query access detection in rbac-administered databases with random forest and pca," *Information Sciences*, vol. 369, pp. 238–250, 2016.

[56] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE, 2000, pp. 130–143.

[57] R. Motwani, S. U. Nabar, and D. Thomas, "Auditing sql queries," in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 287–296.

[58] J. Pei, S. J. Upadhyaya, F. Farooq, and V. Govindaraju, "Data mining for intrusion detection: techniques, applications and systems," in *Proceedings. 20th International Conference on Data Engineering*. IEEE, 2004, pp. 877–877.

[59] D. A. Kindy and A.-S. K. Pathan, "A survey on sql injection: Vulnerabilities, attacks, and prevention techniques," in *2011 IEEE 15th international symposium on consumer electronics (ISCE)*. IEEE, 2011, pp. 468–471.

[60] Y. Chen, S. Nyemba, and B. Malin, "Detecting anomalous insiders in collaborative information systems," *IEEE transactions on dependable and secure computing*, vol. 9, no. 3, pp. 332–344, 2012.

[61] M. Doroudian and H. R. Shahriari, "A hybrid approach for database intrusion detection at transaction and inter-transaction levels," in *2014 6th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2014, pp. 1–6.

[62] N. Singh and S. Singh, "A modified mean gray wolf optimization approach for benchmark and biomedical problems," *Evolutionary Bioinformatics*, vol. 13, p. 1176934317729413, 2017.

[63] Tpc-c homepage. [Online]. Available: https://www.tpc.org/tpcc/default5.asp

[64] "High-resolution record of northern hemisphere climate extending into the last interglacial period," *Nature*, vol. 431, no. 7005, pp. 147–151, 2004.

[65] C. K. Das, O. Bass, G. Kothapalli, T. S. Mahmoud, and D. Habibi, "Overview of energy storage systems in distribution networks: Placement, sizing, operation, and power quality," *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 1205–1230, 2018.

[66] B. Winzeler, N. Cesana-Nigro, J. Refardt, D. R. Vogt, C. Imber, B. Morin, M. Popovic, M. Steinmetz, C. O. Sailer, G. Szinnai *et al.*, "Arginine-stimulated copeptin measurements in the differential diagnosis of diabetes insipidus: a prospective diagnostic study," *The Lancet*, vol. 394, no. 10198, pp. 587–595, 2019.

[67] L. A. Slevin and A. Macfie, "Role based access control for a medical database," in *IASTED-Software Engineering and Applications Conference*. Citeseer, 2007, pp. 19–21.

[68] F. Zhang, D. Pant, and B. E. Logan, "Long-term performance of activated carbon air cathodes with different diffusion layer porosities in microbial fuel cells," *Biosensors and Bioelectronics*, vol. 30, no. 1, pp. 49–55, 2011.

[69] L. Rostad and O. Edsberg, "A study of access control requirements for healthcare systems based on audit trails from access logs," in *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*. IEEE, 2006, pp. 175–186.

[70] E. Bertino, J.-W. Byun, and A. Kamra, "Database security," in *Security, Privacy, and Trust in Modern Data Management*. Springer, 2007, pp. 87–101.

[71] K. C. Chan and W.-H. Au, "Mining fuzzy association rules," in *Proceedings of the sixth international conference on Information and knowledge management*, 1997, pp. 209–215.

[72] Q. Wang and V. Megalooikonomou, "A clustering algorithm for intrusion detection," in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005*, vol. 5812. SPIE, 2005, pp. 31–38.

[73] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *proceedings of the 17th international conference on data engineering*. Citeseer, 2001, pp. 215–224.

[74] P.-y. Liu, W. Gong, and X. Jia, "An improved prefixspan algorithm research for sequential pattern mining," in *2011 IEEE International Symposium on IT in Medicine and Education*, vol. 1. IEEE, 2011, pp. 103–108.

[75] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.

[76] Z. Shou and X. Di, "Similarity analysis of frequent sequential activity pattern mining," *Transportation Research Part C: Emerging Technologies*, vol. 96, pp. 122–143, 2018.

[77] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[78] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[79] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*. Springer, 1998, pp. 355–368.

[80] A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah, "A comparison study on similarity and dissimilarity measures in clustering continuous data," *PloS one*, vol. 10, no. 12, 2015.

[81] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *International Conference on Extending Database Technology*. Springer, 1996, pp. 1–17.

[82] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "Freespan: frequent pattern-projected sequential pattern mining," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 355–359.

[83] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2007, vol. 382.

[84] P. Mitra, S. K. Pal, and M. A. Siddiqi, "Non-convex clustering using expectation maximization algorithm with rough set initialization," *Pattern Recognition Letters*, vol. 24, no. 6, pp. 863–873, 2003.

[85] C. Ordonez and E. Omiecinski, "Frem: fast and robust em clustering for large data sets," in *Proceedings of the eleventh international conference on Information and knowledge management*, 2002, pp. 590–599.

[86] A. Srivastava, S. Sural, and A. K. Majumdar, "Database intrusion detection using weighted sequence mining," *Journal of Computers*, vol. 1, no. 4, pp. 8–17, 2006.

[87] S. Hashemi, Y. Yang, D. Zabihzadeh, and M. Kangavari, "Detecting intrusion transactions in databases using data item dependencies and anomaly analysis," *Expert Systems*, vol. 25, no. 5, pp. 460–473, 2008.

[88] S.-J. Bu and S.-B. Cho, "A convolutional neural-based learning classifier system for detecting database intrusion via insider attack," *Information Sciences*, vol. 512, pp. 123–136, 2020.

[89] A. J. Fernández-García, L. Iribarne, A. Corral, J. Criado, and J. Z. Wang, "A flexible data acquisition system for storing the interactions on mashup user interfaces," *Computer Standards & Interfaces*, vol. 59, pp. 10–34, 2018.

[90] J. Wang and J. Han, "Bide: Efficient mining of frequent closed sequences," in *Proceedings. 20th international conference on data engineering.* IEEE, 2004, pp. 79–90.

[91] D. Van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 1. IEEE, 2003, pp. 215–220.

[92] A. Sallam and E. Bertino, "Result-based detection of insider threats to relational databases," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, 2019, pp. 133–143.

[93] M. R. Keyvanpour, M. Barani Shirzad, and S. Mehmandoost, "Cid: a novel clustering-based database intrusion detection algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 1601–1612, 2021.

[94] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2014, pp. 40–52.

[95] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42, no. 1, pp. 31–60, 2001.

[96] D. Cappelli, A. Moore, R. Trzeciak, and T. J. Shimeall, "Common sense guide to prevention and detection of insider threats 3rd edition–version 3.1," *Published by CERT, Software Engineering Institute, Carnegie Mellon University, http://www. cert. org*, 2009.

[97] S. Furnell, "Enemies within: the problem of insider attacks," *Computer fraud & security*, vol. 2004, no. 7, pp. 6–11, 2004.

[98] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[99] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," 1973.

[100] A. Mangalampalli and V. Pudi, "Fuzzy association rule mining algorithm for fast and efficient performance on very large datasets," in *2009 IEEE international conference on fuzzy systems*. IEEE, 2009, pp. 1163–1168.

[101] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 2004, p. 31.

[102] T.-Y. Kim and S.-B. Cho, "Optimizing cnn-lstm neural networks with pso for anomalous query access control," *Neurocomputing*, vol. 456, pp. 666–677, 2021.

[103] S.-J. Bu, H.-B. Kang, and S.-B. Cho, "Ensemble of deep convolutional learning classifier system based on genetic algorithm for database intrusion detection," *Electronics*, vol. 11, no. 5, p. 745, 2022.

[104] G.-C. Lan, T.-P. Hong, and H.-Y. Lee, "An efficient approach for finding weighted sequential patterns from sequence databases," *Applied Intelligence*, vol. 41, pp. 439–452, 2014.

[105] I. Singh, S. Sareen, and H. Ahuja, "Detection of malicious transactions in databases using dynamic sensitivity and weighted rule mining," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, March 2017, pp. 1–8.

[106] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, Sep. 2000, pp. 39–48.

[107] M. Gondree and P. Mohassel, "Longest common subsequence as private search," 01 2009, pp. 81–90.

[108] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[109] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," 03 2004, pp. 711–716.

[110] A. Srivastava, S. Sural, and A. Majumdar, "Database intrusion detection using weighted sequence mining," *JCP*, vol. 1, pp. 8–17, 07 2006.