Insider Threat Detection Model for the Cloud

Lucky Nkosi, Paul Tarwireyi and Matthew O Adigun
Department of Computer Science
University of Zululand
KwaDlangezwa, South Africa
{mrlnkosi, ptarwireyi, profmatthewo}@gmail.com

***Abstract*—Cloud computing is a revolutionary technology that is changing the way people and organizations conduct business. It promises to help organizations save money on IT expenditure while increasing reliability, efficiency and productivity. However, despite the potential benefits that the cloud promises its users, it is facing some security challenges. Insider threats are some of the growing security concerns that are hindering the adoption of the cloud. Cloud providers are faced with a challenge of monitoring usage patterns of users so as to ensure that malicious insiders do not compromise the security of customer data and applications. Solutions are still needed to ensure that the data stored in the cloud is secure from malicious insiders of the cloud service provider. This paper presents an Insider Threat Detection Model that can be used to detect suspicious insider activities. An experimental system was designed to implement this model. This system uses sequential rule mining to detect malicious users by comparing incoming events against user profiles**

***Keywords*— sequential rule mining; insider; cloud; security**

I. INTRODUCTION

Discovering behavior patterns from a series of events stored in large databases provide a better understanding on accurately predicting how each user behaves in the system when given legitimate access. Monitoring behavior patterns of malicious insiders who have been given access to a system is one of the most significant problems facing providers in the cloud environment [1]. Insiders include current employees, contractors and former employees who have been given legitimate access. These types of users pose a serious threat to the confidentiality, integrity and availability of information stored in the cloud since they may deviate from their normal usage patterns and performs actions that are not related to their job role. Even though attacks originating from an insider of cloud service provider may not occur frequently, they have a high rate of going undetected because they are performed by knowledgeable users who are familiar with security controls in place; to detect and prevent unauthorized access [2]. According to the CSI / FBI report [3], insider abuse is one of the highly rated or cited types of attack even though many organizations employ different access control mechanisms [9]. As a result, immediate attention is required to detect the deviation of normal behavior of insiders in order to reduce the risk posed by Insiders.

This paper presents an Insider Threat Detection Model that acknowledges that the insider threat problem is complex and cannot be tackled by technical solutions alone. It needs a mixture of both technical and non-technical solutions. Even though the focus was on the monitoring component, this paper highlights some of the non-technical components that have interdependencies with our model. The rest of the paper is organised as follows: Section (II) and (III) present reviews of monitoring techniques and related models respectively. Section (IV) discusses the proposed model while Section (V) presents a proof of concept. Section (VI) discusses the results and Section (VII) presents some concluding remarks and future work.

II. MONOTORING TECHNIQUES

Different techniques such as machine learning algorithms and statistical modelling techniques have been used in several works in the literature, to identify the deviation of user behaviour patterns when accessing the system [4][5][8][11][12][10][11][12].

One of the recent techniques that have been used as an input for different data mining techniques in the web usage mining is Web server logs [6]. This technique matches the active user session with the previous stored user profile to judge the behaviour of the user. However, web server logs do not provide sufficient information to facilitate effective matching. This is because of the caching mechanism that is mostly used in web browsers.

Another popular technique that is used to discover temporal relations between events in discrete time is sequential pattern mining [7]. This approach aims at finding a sequence of events that appear frequently in the database in order to predict the behavior pattern of a particular user. This is to a certain extent is similar to what happens in web usage mining. Because sequential pattern mining uses unordered elements, identical but shifted patterns are combined and filtered from the output. That is, the sequences “XYX” and “XXY” are considered the same.

System calls are one of the approaches that have been used to build profiles for monitoring the behaviour of insiders within a system [15]. This approach uses two methods; user-oriented model and process-oriented model. In the first method, access patterns are obtained to generate meaningful data that can be used to build a profile for a particular user

This work is based on the research supported in part by the National Research Foundation of South Africa -Grant UID: TP11062500001 (2012-2014). Funds were also received from industry partners: (Telkom SA Ltd, Huawei Technologies SA (Pty) Ltd and Dynatech Information Systems, South African in support of this research.

within the system. Process-oriented determine how processes are used to access files in the system. The approach proposed Nguyen in (2003), presents some uniqueness when it is used to counter malicious behaviour of individuals within a system, as it analyses the system trace performed by the user and build a profile that can be used for monitoring purposes [15]. However, this approach also has its own drawbacks. The first one being that it mainly focuses on the system call level only hence it can only be effective for attacks that are known to the system. Another issue associated with this approach is that the false alarm rate is often higher because of the possibilities of a large number of system calls that can be made by a user.

III. Related Models

A. *Cert Model*

In an effort to address the insider threat problem, CERT proposed a generic model that can be used to reduce insider threats [13]. This model is applicable not only in the cloud computing environment, but in every system that stores data. Fig. 1 shown below, depicts the CERT model with its various controls such as preventive controls, detective controls and corrective controls. Preventive controls are used as a first line of defence, while detective controls are used as a second line of defence; when a malicious user is not prevented at least a malicious user must be detected. The corrective control is responsible for reacting or responding on the detected threat.

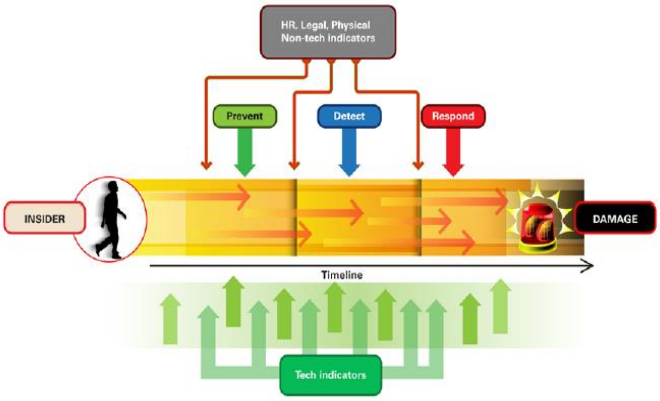


Fig. 1: Cert Model [13]

From the above figure, it is evident that in order to effectively reduce insider threats technical controls alone cannot be effective. They need to work in conjunction with non-technical controls. The combination of technical controls such as tech indicators, preventive, detective and correctness controls and non-technical controls such as policies that are enforced in HR before any user can access the system can yield to a better solution that can reduce insider threats. This paper is arguing that the model proposed by CERT have all the necessary components that are needed in addressing the insider threat problem. However, the model was discussed from a high level point of view without necessarily going deeper in terms of demonstrating the types of preventive controls, detective controls and correctness controls that are implemented in order to reduce the insider threats.

This contribution this paper to the model is on detective controls. The model proposed in this paper zooms into the CERT model and provides a solution that can be used to detect malicious insider based on behavioural patterns

B. *Model Based Prediction*

The predictive model is one of the models that have been used in addressing the insider threat problem. This model employs a hybrid approach which uses pattern recognition and model-based reasoning in identifying deviations from the normal behaviour [14]. The Model-based approach shows how a malicious user can be detected by analyzing data that has been collected.

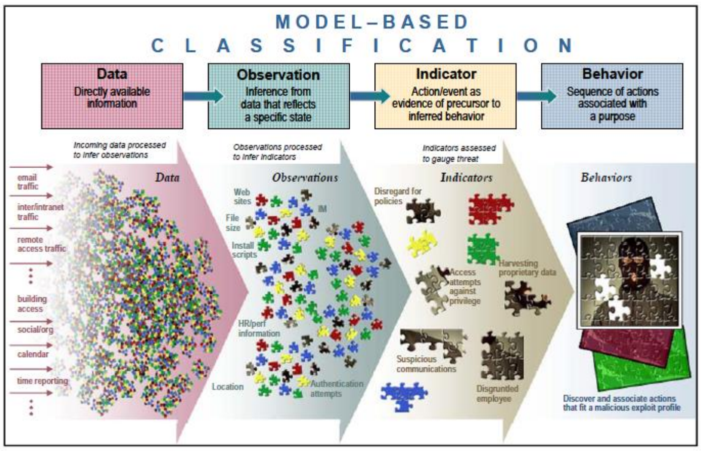

MODEL–BASED
CLASSIFICATION
Data
Directly available information
Observation
Inference from data that reflects a specific state
Indicator
Action/event as evidence of precursor to inferred behavior
Behavior
Sequence of actions associated with a purpose
Incoming data processed to infer observations
Observations processed to infer indicators
Indicators assessed to gauge threat
email traffic
inter/intranet traffic
remote access traffic
building access
social/org
calendar
time reporting
Data
Observations
Web sites
IM
File size
Install scripts
HR/perf information
Location
Authentication attempts
Indicators
Disregard for policies
Harvesting proprietary data
Access attempts against privilege
Suspicious communications
Disgruntled employee
Behaviors
Discover and associate actions that fit a malicious exploit profile

Fig. 2: Model Based Prediction

Fig. 2 depicts the predictive model with all the steps that are involved in analysing the behaviour of users when they are accessing the system.

The model has steps which include data collection, observations that can be made from the collected data, indicators that show actions that have been carried out by a user and the behaviour of the user composed of sequences associated with that user. However, there are some challenges associated with this model; one is to conduct model-based reasoning on the recognized pattern at the semantic level rather than applying template recognition. Another issue is with the noise data that is collected, which usually needs to be filtered before any observations can be made.

Although much work has been done in the area of minimizing insider threats in dynamic environments, the focus was either generic or confined to a single category of solutions such as purely technical or purely non-technical. Based on the reviewed works, it is evident that there is still a need for a solution that can provide monitoring in order to detect the users deviating from their normal usage patterns.

In the following section, we present our proposed insider threat detection model, which aims at monitoring user behaviour pattern, based on the generated sequences of events, in order to detect the deviation of users from their normal behaviour.

IV. PROPOSED MODEL

Fig. 3 illustrates the model that was developed in this research to monitor user behaviour patterns when legitimate users are given system access. The proposed model has seven components that work together with the aim of monitoring users that are deviating from their normal behaviour patterns. These components are highlighted in this section.

A. Management of User Identities

This component encompasses some of the non-technical solutions that have to be in place to ensure that the right employees are hired in the first place. The cloud HR departments should conduct pre-employment background checks of all new administrators to get more information on each candidate and make informed decisions [16]. This might include checking credit reports, criminal records, school and medical reports. The legal department must craft policies and procedures to govern the access and use of IT resources.

Once the right candidate has been hired, they have to be assigned to the proper roles and privileges. This ensures that the hired administrators possess valid credentials and have access to appropriate cloud resources.

Since different administrators fall in different domains and require different roles to perform tasks, we do not cluster many users in one role but we decentralized the process of role assignment by mapping one role to a specific account for a user.

As part of managing user identities, roles and privileges should always change to match the current job profile of the administrator at any given point in time. This means that privileges should be terminated when the administrator leaves the company. The literature has shown most insider attacks are performed by disgruntled employees or employees who have resigned and are currently serving their notice [16]. The HR department should work closely with the IT department to ensure that user identities are monitored closely under these circumstances and take appropriate action. There should be proper policies and procedures to govern this.

B. Data Collection Process

After the users have been given access to the system, the first step is to train them on how to use the system. The system should keep track of all the actions that are performed by the user and record these actions since the raw data is generated when the user is accessing the system.

C. Log

All events are sent to log files in order to discover the pattern for each user. In order to identify an interesting pattern that characterize each user when interacting with the system, the log file is analysed and the events are filtered and extracted to create a user profile.

D. Pattern Generation

Once events are captured and stored in the log, the next step is to perform pattern generation that forms the profile of the user based on the set of events executed by the user.

In order to accurately determine the behaviour patterns of the users, the behavioural pattern of the user is profiled by training each user on how they are expected to behave in the system when given legitimate access.

Determining whether a user is a malicious user or not without having a predefined behaviour pattern is difficult. We are not claiming that profiling the behaviour pattern of the user is the only solution to the problem of mitigating malicious insiders, but what we are emphasizing is that profiling user behaviour pattern provides a way of dealing with the problem of insider threats.

E. User Profile

This component is used to store the training dataset that depicts the behaviour of the trained user and testing dataset that depict the behaviour of untrained users.

F. Policy Base

This component deals with the aspect of non-technical controls such as policies and procedures that must be enforced since technical controls alone are not enough to mitigate insider threats.

G. Pattern Matching

In order to detect the user behaviour pattern, we employ the sequential pattern mining technique proposed in this paper. Sequential rule matching approach uses a sequential pattern mining technique in order to identify patterns, by comparing the current generated pattern with the one stored in the user profile to find out whether the behaviour pattern is still consistent with the previous behaviour pattern. If the behaviour pattern stored in the user profile is not the same with the current pattern we assume that there is a probability that a user is a malicious user or else if the behaviour pattern is consistent with the one stored in the user profile, we assume that the user is a normal user.

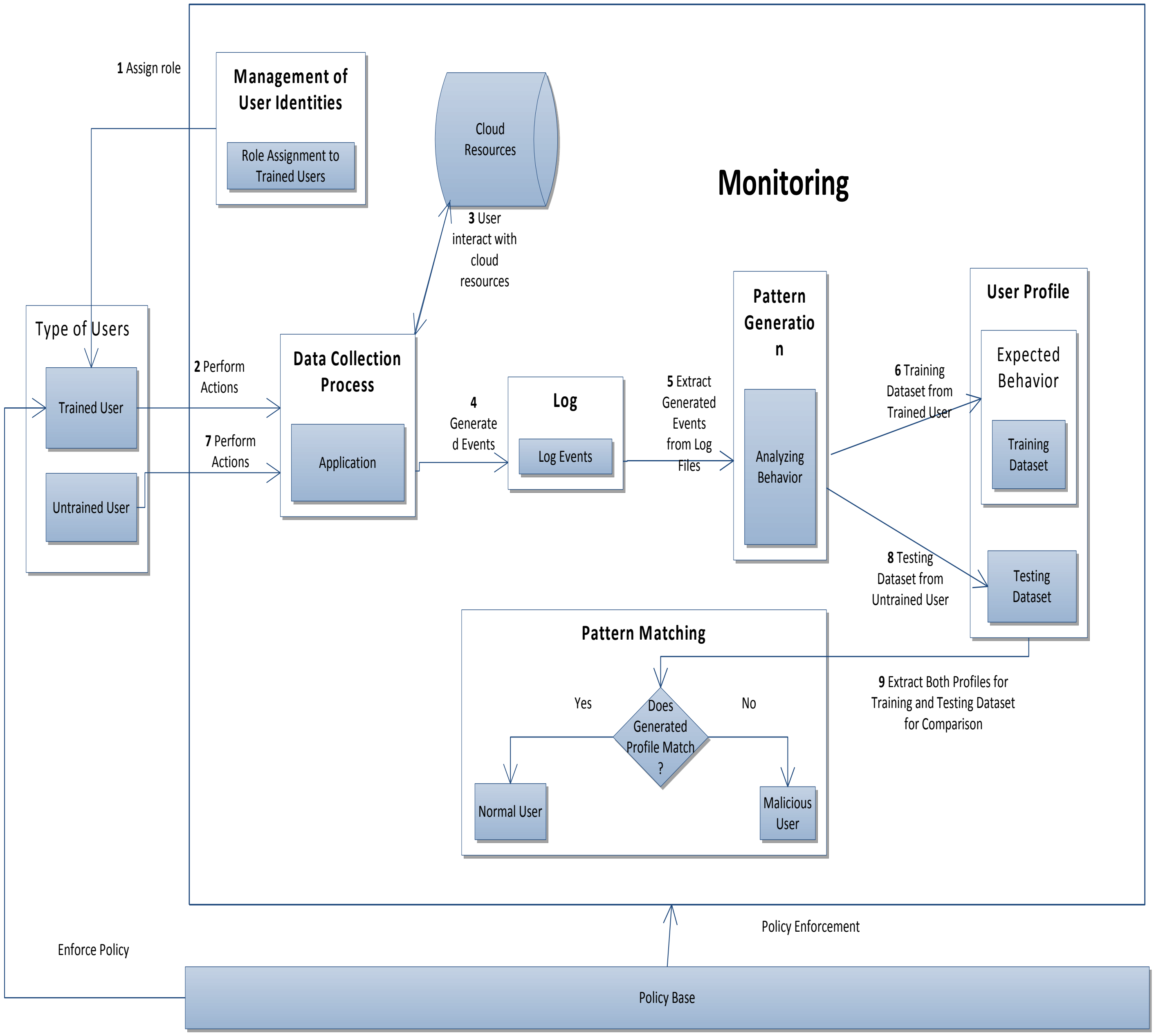


Fig. 3: Insider Threat Reduction Model

V. PROOF OF CONCEPT

To test the application that was developed, we used three machines. Due to resource constraints, for the deployment of the cloud we used desktop machines running Ubuntu 12.10 to deploy an OpenStack Folsom private cloud. Three i7 machines with 8Gig of RAM each were used. A simple web service based application which implements basic access control features was developed using Netbeans 7.1.1 and was deployed in the cloud. We should be clear here, that for the purposes of our demonstration, we tested our implementation at the Software as a Service (SaaS) layer of our cloud. Therefore, Platform as a Service (PaaS) and Infrastructure as a Service are out of the scope of this work. As part of this application, there was also a service to monitor and log all the user events for the various activities in a session.

In order to generate patterns that form user profiles, we used the Sequential Pattern Mining Framework (SPMF). This is because SPMF it is an open-source data mining platform written in Java and it offers the implementations of 50 data mining algorithms. It also provides the flexibility that any source code of each algorithm can be integrated in any java platform. Moreover, SPMF has a well-developed API for mining the data and analyse the behaviour pattern of the users in a distributed environment [8].

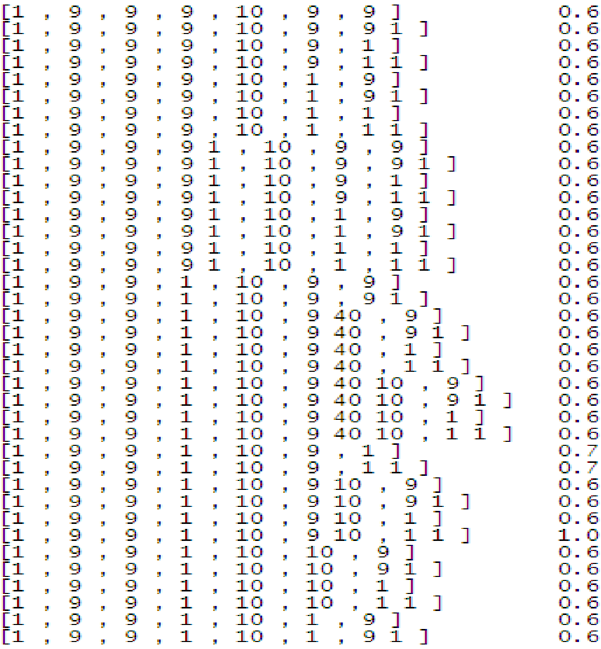
Since our goal was to find frequent sequences of events that appear from user actions that form a pattern. We used the PrefixSpan algorithm defined in SPMF for mining frequent sequential pattern from the sequence of events. PrefixSpan algorithm uses the Apriori-based approach and pattern-growth-based approach. In the PrefixSpan algorithm, we implemented different methods in order to cater for our desired goal to get sequence of events that are interesting to a user that form a profile of a trained user. After pattern analysis is done, the profile characterizing a certain user in the system is stored.

As specified in the model, we ensure that every sequence of events performed by the user is mapped directly to a user profile. This mapping assists us in ensuring that at any given point in time, we won't experience a situation where two users' behaviour in the same way in the system when given legitimate access. We then use trained user and untrained user for the purpose of testing whether the system can reason about the behaviour pattern that is formed and differentiate between malicious user and normal user. In the process of building a profile for the user, the steps 1- 4 depicted in the model are the major steps that need to be considered before a profile of a trained user is formed and stored in the repository which is called training data set. While steps 5 – 2 – 3 – 6 illustrate the generation of the untrained user profile so that it can be compared with the profile that is previously stored in the repository of a trained user, which is called a testing data set. To match patterns in different profiles in order to judge the behaviour of the user, we use Sequential rule mining. Since sequential rule mining mainly help in finding rules that appear in the same order that support a predefined minimum support and confidence and have a form of A implies T, where both A and T are itemsets. These rules are interpreted in such a way that if event A appears, event T is mostly to appear as well. In the new profile that is generated we check whether all rules that have been generated in the previous profile that show the behavior pattern of a normal user still hold or not. If the behavior is not consistent with the previous profile, then the user is most likely to be a malicious user.

VI. Results Analysis

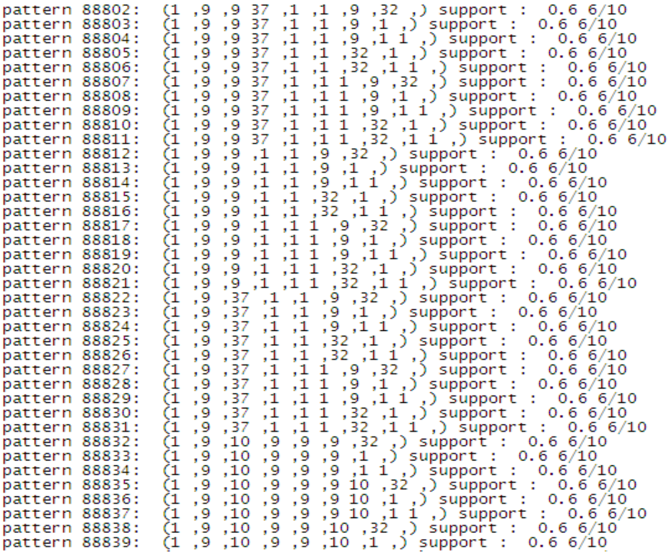
This section details the experiments conducted and discusses the results that were obtained. The main aim of the experiment is to distinguish between a malicious user and a normal user through comparing sets of actions performed by trained user and untrained user. The process of identifying or detecting a malicious and a normal user was done by first training one real user on how to execute operations in the system based on the role that was assigned to the user. All sequences of events that are generated when the trained user interacted with the system were extracted to the PrefixSpan algorithm to generate sequential rules that are called the training dataset, which was used to form a profile for the user. The figures (Fig. 4 and Fig. 5) that follow show the sequential rules that were generated in order to form user profiles. These sequential rules indicate how each user is expected to behave in the system.

Fig. 4 illustrates the profile of a trained user with the set of sequential rules. What can be observed from sequential rule one is that, when event 1 occurs, event 9, 9, 9, 10, 9 implies event 9 is mostly to occur with 0.6 minimum supports. An event is represented by a numeric number indicating a specific operation performed by a user in the system. The way in which each sequential rule is executed indicates the way each user was behaving when interacting with the system. An untrained user was also given the same credentials used above, with the aim of observing the behavior pattern based on the sequential rules that are generated from the untrained user. Fig. 5 shows the sequence of events generated from an untrained user that form a test data set. Sequence of events indicates how the user was executing tasks in the system.



```
[1 , 9 , 9 , 9 , 10 , 9 , 9 ]                 0.6
[1 , 9 , 9 , 9 , 10 , 9 , 9 1 ]               0.6
[1 , 9 , 9 , 9 , 10 , 9 , 1 ]                 0.6
[1 , 9 , 9 , 9 , 10 , 9 , 1 1 ]               0.6
[1 , 9 , 9 , 9 , 10 , 1 , 9 ]                 0.6
[1 , 9 , 9 , 9 , 10 , 1 , 9 1 ]               0.6
[1 , 9 , 9 , 9 , 10 , 1 , 1 ]                 0.6
[1 , 9 , 9 , 9 , 10 , 1 , 1 1 ]               0.6
[1 , 9 , 9 , 9 1 , 10 , 9 , 9 ]               0.6
[1 , 9 , 9 , 9 1 , 10 , 9 , 9 1 ]             0.6
[1 , 9 , 9 , 9 1 , 10 , 9 , 1 ]               0.6
[1 , 9 , 9 , 9 1 , 10 , 9 , 1 1 ]             0.6
[1 , 9 , 9 , 9 1 , 10 , 1 , 9 ]               0.6
[1 , 9 , 9 , 9 1 , 10 , 1 , 9 1 ]             0.6
[1 , 9 , 9 , 9 1 , 10 , 1 , 1 ]               0.6
[1 , 9 , 9 , 9 1 , 10 , 1 , 1 1 ]             0.6
[1 , 9 , 9 , 1 , 10 , 9 , 9 ]                 0.6
[1 , 9 , 9 , 1 , 10 , 9 , 9 1 ]               0.6
[1 , 9 , 9 , 1 , 10 , 9 40 , 9 ]              0.6
[1 , 9 , 9 , 1 , 10 , 9 40 , 9 1 ]            0.6
[1 , 9 , 9 , 1 , 10 , 9 40 , 1 ]              0.6
[1 , 9 , 9 , 1 , 10 , 9 40 , 1 1 ]            0.6
[1 , 9 , 9 , 1 , 10 , 9 40 10 , 9 ]           0.6
[1 , 9 , 9 , 1 , 10 , 9 40 10 , 9 1 ]         0.6
[1 , 9 , 9 , 1 , 10 , 9 40 10 , 1 ]           0.6
[1 , 9 , 9 , 1 , 10 , 9 40 10 , 1 1 ]         0.6
[1 , 9 , 9 , 1 , 10 , 9 , 1 ]                 0.7
[1 , 9 , 9 , 1 , 10 , 9 , 1 1 ]               0.7
[1 , 9 , 9 , 1 , 10 , 9 10 , 9 ]              0.6
[1 , 9 , 9 , 1 , 10 , 9 10 , 9 1 ]            0.6
[1 , 9 , 9 , 1 , 10 , 9 10 , 1 ]              0.6
[1 , 9 , 9 , 1 , 10 , 9 10 , 1 1 ]            1.0
[1 , 9 , 9 , 1 , 10 , 10 , 9 ]                0.6
[1 , 9 , 9 , 1 , 10 , 10 , 9 1 ]              0.6
[1 , 9 , 9 , 1 , 10 , 10 , 1 ]                0.6
[1 , 9 , 9 , 1 , 10 , 10 , 1 1 ]              0.6
[1 , 9 , 9 , 1 , 10 , 1 , 9 ]                 0.6
[1 , 9 , 9 , 1 , 10 , 1 , 9 1 ]               0.6
```

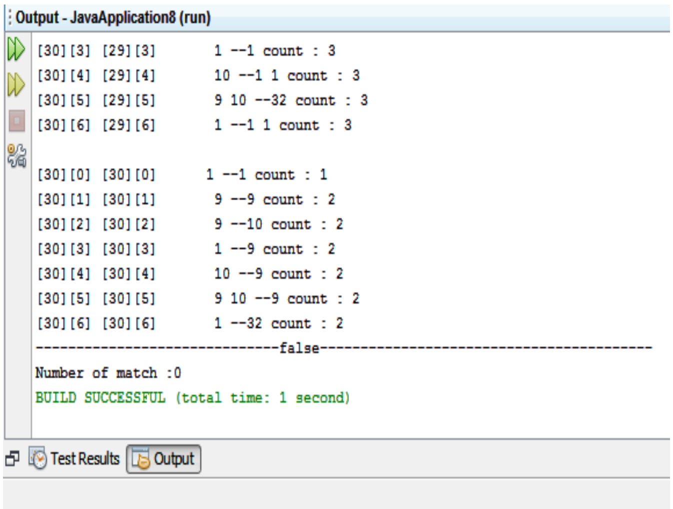
Fig. 4: User Profile



```
pattern 88802:  (1 ,9 ,9 37 ,1 ,1 ,9 ,32 ,) support :  0.6 6/10
pattern 88803:  (1 ,9 ,9 37 ,1 ,1 ,9 ,1 ,) support :  0.6 6/10
pattern 88804:  (1 ,9 ,9 37 ,1 ,1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88805:  (1 ,9 ,9 37 ,1 ,1 ,32 ,1 ,) support :  0.6 6/10
pattern 88806:  (1 ,9 ,9 37 ,1 ,1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88807:  (1 ,9 ,9 37 ,1 ,1 1 ,9 ,32 ,) support :  0.6 6/10
pattern 88808:  (1 ,9 ,9 37 ,1 ,1 1 ,9 ,1 ,) support :  0.6 6/10
pattern 88809:  (1 ,9 ,9 37 ,1 ,1 1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88810:  (1 ,9 ,9 37 ,1 ,1 1 ,32 ,1 ,) support :  0.6 6/10
pattern 88811:  (1 ,9 ,9 37 ,1 ,1 1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88812:  (1 ,9 ,9 ,1 ,1 ,9 ,32 ,) support :  0.6 6/10
pattern 88813:  (1 ,9 ,9 ,1 ,1 ,9 ,1 ,) support :  0.6 6/10
pattern 88814:  (1 ,9 ,9 ,1 ,1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88815:  (1 ,9 ,9 ,1 ,1 ,32 ,1 ,) support :  0.6 6/10
pattern 88816:  (1 ,9 ,9 ,1 ,1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88817:  (1 ,9 ,9 ,1 ,1 1 ,9 ,32 ,) support :  0.6 6/10
pattern 88818:  (1 ,9 ,9 ,1 ,1 1 ,9 ,1 ,) support :  0.6 6/10
pattern 88819:  (1 ,9 ,9 ,1 ,1 1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88820:  (1 ,9 ,9 ,1 ,1 1 ,32 ,1 ,) support :  0.6 6/10
pattern 88821:  (1 ,9 ,9 ,1 ,1 1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88822:  (1 ,9 ,37 ,1 ,1 ,9 ,32 ,) support :  0.6 6/10
pattern 88823:  (1 ,9 ,37 ,1 ,1 ,9 ,1 ,) support :  0.6 6/10
pattern 88824:  (1 ,9 ,37 ,1 ,1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88825:  (1 ,9 ,37 ,1 ,1 ,32 ,1 ,) support :  0.6 6/10
pattern 88826:  (1 ,9 ,37 ,1 ,1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88827:  (1 ,9 ,37 ,1 ,1 1 ,9 ,32 ,) support :  0.6 6/10
pattern 88828:  (1 ,9 ,37 ,1 ,1 1 ,9 ,1 ,) support :  0.6 6/10
pattern 88829:  (1 ,9 ,37 ,1 ,1 1 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88830:  (1 ,9 ,37 ,1 ,1 1 ,32 ,1 ,) support :  0.6 6/10
pattern 88831:  (1 ,9 ,37 ,1 ,1 1 ,32 ,1 1 ,) support :  0.6 6/10
pattern 88832:  (1 ,9 ,10 ,9 ,9 ,9 ,32 ,) support :  0.6 6/10
pattern 88833:  (1 ,9 ,10 ,9 ,9 ,9 ,1 ,) support :  0.6 6/10
pattern 88834:  (1 ,9 ,10 ,9 ,9 ,9 ,1 1 ,) support :  0.6 6/10
pattern 88835:  (1 ,9 ,10 ,9 ,9 ,9 10 ,32 ,) support :  0.6 6/10
pattern 88836:  (1 ,9 ,10 ,9 ,9 ,9 10 ,1 ,) support :  0.6 6/10
pattern 88837:  (1 ,9 ,10 ,9 ,9 ,9 10 ,1 1 ,) support :  0.6 6/10
pattern 88838:  (1 ,9 ,10 ,9 ,9 ,10 ,32 ,) support :  0.6 6/10
pattern 88839:  (1 ,9 ,10 ,9 ,9 ,10 ,1 ,) support :  0.6 6/10
```

Fig. 5: Sequence of Events Generated From Untrained User.

From Fig. 5 above, we can observe the way the untrained user was executing tasks in system when given valid credentials of a legitimate user. The generated sequence of events of the untrained user was then extracted to the matching algorithm in order to reason about the behavior pattern of an untrained user.



```
[30][3]  [29][3]        1 --1 count : 3
[30][4]  [29][4]        10 --1 1 count : 3
[30][5]  [29][5]        9 10 --32 count : 3
[30][6]  [29][6]        1 --1 1 count : 3

[30][0]  [30][0]       1 --1 count : 1
[30][1]  [30][1]        9 --9 count : 2
[30][2]  [30][2]        9 --10 count : 2
[30][3]  [30][3]        1 --9 count : 2
[30][4]  [30][4]        10 --9 count : 2
[30][5]  [30][5]        9 10 --9 count : 2
[30][6]  [30][6]        1 --32 count : 2
------------------------------false------------------------------------------
Number of match :0
BUILD SUCCESSFUL (total time: 1 second)
```

Fig. 6: Pattern Matching

Fig. 6 depicts the outcome after the testing dataset and the training dataset were compared with the aim of detecting whether a user is a malicious user or not. The developed system managed to pick up a malicious user after comparing sequential rules from both parties involved. This means that, employing the data mining techniques such as the sequential rule mining technique can help in identifying malicious users. After the system has picked up a malicious user, the super administrator is then alert about such activities carried out by alicious user.

Three experiments were carried out to evaluate the Insider Threat Detection System under discussion. The first experiment was carried out with the aim of assessing the effect of increasing minimum support on the quality of the profiles mined when observing specificity, sensitivity and precision. Minimum support is a user defined threshold that is used to control the number of rules generated [8]. Minimum support was used as a controlling factor to see the effect it has when it being decreased or increased in the proposed method. Specificity and sensitivity are statistical measures of performance of the binary classification test. These measures are defined as follows:

Sensitivity = Number of True Positives (TP) / Number of Positive (1)

Number of true positives: refer to the number of positive events that were correctly identified as positive and number of positive refers to the total size of all events. Where sensitivity is define as the probability that the test say a user is normal when a user is real normal user.

Specificity = Number of True Negatives / Number of Negative (2)

Number of true negatives refers to the events that were correctly labeled as negative. Specificity measures the proportion of negatives which are correctly identified. Precision was also considered as a metric, and is measured as follows.

Precision = True Positives / (True Positives + False Positives) (3)

TABLE: 1 TESTING AND TRAINING DATASET EXPERIMENT 1

Minsup	Sensitivity	Specificity	Precision
0.6	0.75	0.98	0.17
0.7	0.76	0.94	0.64
0.8	0.77	0.92	0.65
0.9	0.87	0.91	0.67

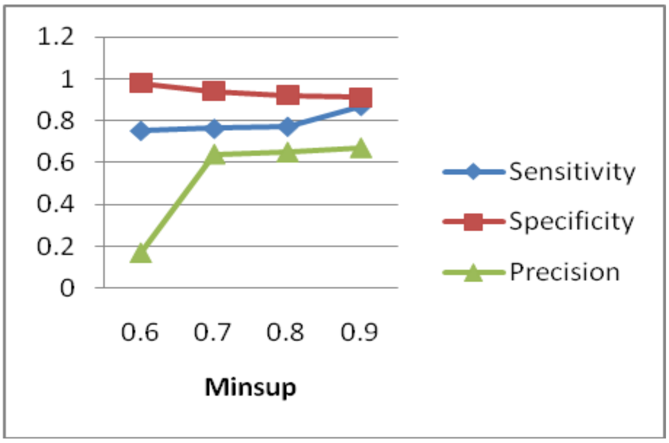


Fig. 7: Sensitivity vs Specificity and Precision

Fig. 7 show positive results based on the quality of profiles that are mined when investigating the quality of profile mined to determine the malicious user. From the figure, as the minimum support is increased, when observing specificity and sensitivity from (0.6 – 0.9), sensitivity increases. This means that the sequential rules that are generated from different users show how each trained user is expected to behave when using the system. Also, when looking at the specificity line, it is higher than sensitivity. This means that according to the sequential rules that were generated, an untrained user was behaving in a malicious way. Based on the results obtained, it becomes easy to identify a malicious user, by simply looking at specificity and sensitivity. What can be observed from the above figure (Fig. 7) is that the quality of profiles mined when the minimum support is increased is of the high quality. This is because of the reason that, at 0.6%, specificity, sensitivity and precision are high, and thus becomes easy to identify a malicious insider based on the quality of the mined profiles.

The second experiment was conducted to investigate type 1 error (False positive rates) with the reduction of minimum support. The aim was to observe whether decreasing minimum support increases or decreases the false positive rate. The computation of type 1 error depends on specificity. Thus, if specificity is high, the chances of the false positive rate to decrease as the minimum support is reduced are high.

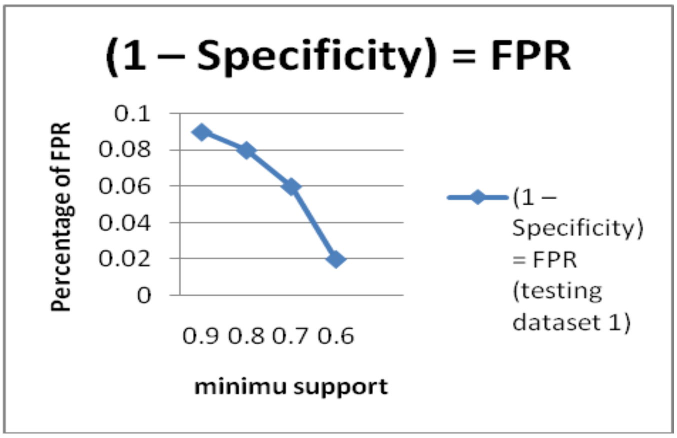


Fig. 8: False Positives Rate

As a result, if the false positive rate decreases, that means the developed model was able to effectively reduce the false positive rate alarm. The Formula for computing false positive rate is defined as follows:

False Positive Rate (FPR) = 1 – specificity (4)

From Fig. 8, it can be seen that the rate of false positives decreases when the minimum support is decreased. We observe the behavior pattern that show positive results obtained when running this experiments.

The goal of this experiment was to observe whether the false positive rate decreases or increases, when the minimum support is reduced. When computing FPR by *(1 - Specificity)* the rate decreases. This shows that the model that was developed is effective in terms of reducing the false positive rate alarm. As a result, insider attack originating from a malicious user is reduced within the organization.

The third experiment was conducted to investigate type 2 (False negative rates) with the increase of minimum support. The aim was to observe whether increasing minimum support decreases false negative rate. The computation of type 2 error depends on sensitivity. If the false negative rate decreases, the better the model that was developed in reducing false negatives rate.

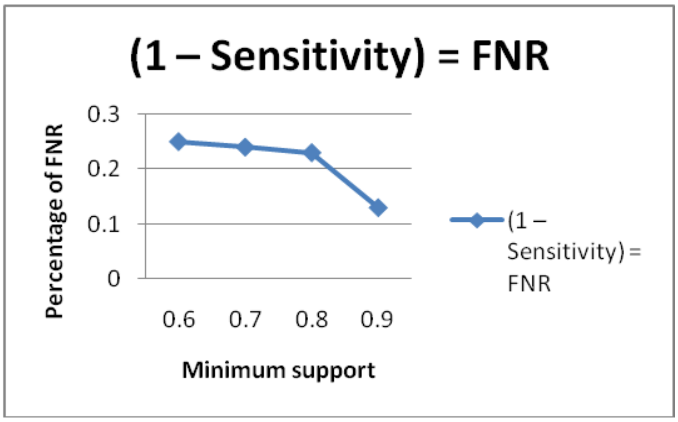


Fig. 9: False Negatives Rate

False Negative Rate (FNR) = 1 – sensitivity (5)

VII. Conclusion and Future Work

In this paper, we have proposed an Insider Threat Detection Model to detect malicious insiders. An experimental system to test this model was implemented to demonstrate the applicability of the proposed model in a SaaS cloud deployment model. This system uses the sequential rule mining approach to detect malicious usage patterns for a particular profile. We also evaluated the system to show what can be done to reduce false positive and false negative rates. The results show that malicious insiders can be detected based on the behavior pattern. Since the implementation discussed in this paper only focuses on the SaaS layer, the future work lies in investigating how this model can be applied in both IaaS, and PaaS layers and probably to other IT systems. Further work is also in examining the scalability of the data mining technique adopted when the minimum support is increased and also the effect of increasing minimum support on the number of rules generated.

References

[1] F. Rocha and M. Correia, 'Lucy in the sky without diamonds: Stealing confidential data in the cloud', 2011, pp. 129–134.

[2] C, Băsescu A. Carpen-Amarie, C. Leordeanu, A. Costan, and G. Antoniu, 'Managing Data Access on Clouds: A Generic Framework for Enforcing Security Policies', in 2011 IEEE International Conference on Advanced Information Networking and Applications (AINA), 2011, pp. 459–466.

[3] The Eighth Annual CSI/FBI 2003 report: "Computer Crime and Security Survey."

[4] D. Lui, X Wang, J Camp "Game-theoretic modelling and analysis of insider threats" International Journal of critical infrastructure protection Volume 1. 75-80 2008

[5] P.F Vinger, R. Nkambou , V.S Tsegi. "RuleGrowth: Mining Sequential Rule Common to Several by Pattern-Growth" 2011. Proceedings on the 2011 ACM Symposium on Applied Computer 956-961.

[6] M. Eirinaki and M. Vazirgiannis. Web mining for personalization. ACM Trans. Internet Tech., 3(1): 1-27, 2003.

[7] R. Agrawal and R. Srikant. Mining Sequential Pattern. In Proc.Int.Conf. on Data Engineering(Taipe, Taiwan, March 6-10, 1995), 3-14.

[8] Y.L Hsieh, D. Yang, J. Wu 2006. Using Data Mining to Study Upstream and Downstream Causal Relationship in Stock Market. Proc.2006 Joint Conference on Information Sciences.

[9] S. J Stolfo, Bellovin, S. M., Hershkop, S., Keromytis, A., Sinclair, S., & Smith, S. W. (Eds.). (2008). Insider attack and cyber security: beyond the hacker (Vol. 39). Springer.

[10] S. Sundararajan,., Narayanan, H., Pavithran, V., Vorungati, K., & Achuthan, K. (2011). Preventing Insider attacks in the Cloud. Advances in Computing and Communications, 488-500.

[11] M. T Khorshed,, Ali, A. S., & Wasimi, S. A. (2011, November). Monitoring insiders activities in cloud computing using rule based learning. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on (pp. 757-764).

[12] W Eberle, Graves, J., & Holder, L. (2010). Insider threat detection using a graph-based approach. Journal of Applied Security Research, 6(1), 32-81.

[13] W. R Claycomb, & Nicoll, A. (2012, July). Insider Threats to Cloud Computing: Directions for New Research Challenges. In Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual (pp. 387-394).

[14] F. L. Greitzer., Paulson, P., Kangas, L., Edgar, T., Zabriskie, M. M., Franklin, L., & Frincke, D. A. (2008). Predictive modelling for insider threat mitigation. Pacific Northwest National Laboratory, Richland, WA, Tech. Rep. PNNL Technical Report PNNL-60737.

[15] N. Nguyen, Reiher, P., & Kuenning, G. H. (2003, June). Detecting insider threats by monitoring system call activity. In Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society (pp. 45-52).

[16] M. R Randazzo,., Keeney, M., Kowalski, E., Cappelli, D., & Moore, A. (2005). Insider threat study: Illicit cyber activity in the banking and finance sector (No. CMU/SEI-2004-TR-021). Carnegie-Mellon Univ Pittsburg PA software Engineering inst.