CONTINUOUS AUTHENTICATION BASED ON LEARNING USER COMMAND SEQUENCE

by

Bijan Khalilian
B.Sc., Ryerson University, Toronto, Ontario, Canada, 2007

A Dissertation Submitted in Partial Fulfillment of the Requirements for the

Degree of

MASTER OF SCIENCE

in the Program of Computer Science

Toronto, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis or dissertation.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

CONTINUOUS AUTHENTICATION BASED ON LEARNING USER COMMAND SEQUENCE

Bijan Khalilian

M.Sc., Computer Science, Ryerson University, 2010

Abstract

In the context of information and computer security, a masquerader is an individual who can gain access to a system by disguising itself as a legitimate user. One of the prominent and popular methods for authenticating masqueraders is by using an intrusion detection system (IDS). This thesis promotes the idea that learning the user command sequence can be served as an alternative for addressing intrusion detection. Several approaches have been proposed in the literature, where this idea has been explored. To our knowledge, the method by Maxion and Townsend produces the best results of all past techniques so far in terms of detection rate (82.1% using the Greenberg dataset). In this thesis, we propose an IDS-based approach that consists in combining a novel Naïve Bayes classifier with a recently proposed sequential sampling technique for continuous authentication, applied to user command sequence, to detect masqueraders. Our experimental evaluation shows that our proposed scheme achieves a detection rate of 98%.

Acknowledgments

It is a pleasure to thank those who made this thesis possible. I owe my deepest gratitude to my supervisor Dr. Isaac Woungang and my co-supervisor Dr. Issa Traore for sharing their knowledge and providing continuous support. Without their help and guidance the completion of this work would have not been possible. I would also like to thank William Zereneh for making his valuable time available to me and many of my colleagues. I am grateful for his timely help and for always keeping his door open for many of us.

Dedication

I dedicate this thesis to my loving parents, Hamid and Mansoureh. Without their patience, support, and most of all love, the completion of this work would have not been possible. Furthermore, I would also like to dedicate this work to my wonderful and loving siblings Bita, Anahita and Aria.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ANN	Artificial Neural Network
CA	Continuous Authentication
CR	Confidence Ratio
DR	Detection Rate
FAR	False Accept Rate
FRR	False Reject Rate
HCS	Hybrid Command Sequence
IDS	Intrusion Detection System
CABLUCS	Continuous Authentication Based on Learning User Command Sequence
MTTA	Mean Time-to-Alarm
NN	Neural Network
PHP	PHP: Hypertext Processor
SVM	Support Vector Machine
TTA	Time-to-Alarm

Chapter 1: Introduction

1.1 Context of Our Study

This thesis introduces a new approach in detecting masquerade attacks in systems, by implementing an intrusion detection system that consists of a Naïve Bayes classifier complemented with a recently proposed sequential sampling technique [1] for continuous authentication, applied to user command sequence. The Naïve Bayes classifier is used to train a dataset that consists of command-line history taken from 168 different users.

1.2 Motivation

In general, intrusion detection systems (IDS) are designed to handle masqueraders, i.e. users who impersonate other users, trying to gain access within a secure network. Typically, it is assumed that sophisticated masqueraders possess insider's knowledge on various features of the system such as topologies, potential vulnerabilities and the how various security products have been installed. To protect systems against masqueraders, security technologies such as firewalls, network-based intrusion detection systems, and strong authentication protocols are utilized.

In general, most authentication systems are only concerned about the point of entry [2]. Once a user has successfully passed the initial phase of authentication (i.e. user login), the user is deemed genuine. However, this assumption can be quite costly. In the case where such initial phase of authentication is compromised, the entire system can be in dire jeopardy. IDSs in conjunction with continuous authentication can be used to address this problem since these systems have been designed to detect different types of security hazardous behaviours after the user has already been given permission to access the system. The general idea of using continuous authentication is that the legitimacy of an active user session can be validated continuously, leading to a predefined and distinctive user profile (i.e. signature) during a live session within intermediate intervals. To this effect, a continuous authentication system can be used to investigate a user's typing habits [3], mouse dynamics [4] or command line sequence, in order to determine the legitimacy of a given user.

IDSs differ from conventional firewall systems and authentication protocols in the sense that in addition to prevent non-privileged users from accessing sensitive data or performing restricted tasks, they can also be used to control the access capability for users with the appropriate and official privileges who abuse their concessions. In other words, IDSs can be used to detect malicious insiders that use their privileges to perform unauthorized actions. For this reason, IDSs are considered as network security schemes of choice [5; 6]. Designing an IDS that can achieve a high level of accuracy while detecting masquerade attacks, is the primary motivation of our work.

1.2 Research Problem

Several IDS-based approaches for masquerade detection have been investigated, ranging from approaches based on support vector machine classifiers [7; 8]; to the pioneer approaches based on mouse dynamics [1] and keystroke dynamics to approaches based on sequence-based user commands profile [9; 10], to name a few. In the latter case, several attempts to learn user command sequence for masquerade detection have been investigated [10]. The proposal that yields the best result so far in terms of accuracy (using the Greenberg dataset) – measured by the level of detection achieved, is the work by Maxion and Townsend [10]. Yet, this performance is still inadequate, especially in the context of commercial-based systems. Therefore, designing IDS-based systems that can detect the above-mentioned canonical masquerade attack based on learning sequence-based user commands while producing a better level of accuracy compared to that obtained by the Maxion and Townsend's approach, would be highly desirable. This is the challenge addressed in this thesis.

1.3 Our Approach

In this thesis, to address the above-mentioned challenge, we follow up on an idea inherited from the pioneer work in [10], i.e. using a Naïve Bayes classification algorithm to learn sequence-based user commands, with the goal to provide a solution to the problem of masquerade detection. Our approach differs from that presented in [10] with respect to the use of updating mechanisms that dynamically recompute the classifier probabilities as monitored sequences are analyzed and classified by our Naïve Bayes classifier. More precisely, our approach consists of an integration of three components: (1) a data pre-processing module – that captures the user's data input and restructured them to a more manageable format; (2) a detector – which deploys a Naïve Bayes classification algorithm (as in [10]) in order to create a set of distinct user profiles; and (3) a dynamic sampling technique for continuous authentication (so-called sequential sampling technique) – which is inherited from a recently pioneered work on continuous authentication [1] and is used to distinguish the legitimacy of a given user based on a given user profile.

1.4 Contribution

There have been a lot of works dealing with learning using command sequence to detect masquerade attacks in systems [10]. To our knowledge, the best detection rate achieved so far is attributed to the approach proposed by Maxion and Townsend [10].

The contribution of this thesis is the design of a novel intrusion detection system that learns from sequence-based user commands profile to detect classical masquerade attacks while learning the behavioural tendencies of a given user. This design is realized by complementing a recently proposed evaluation technique for continuous authentication [1] (so-called sequential sampling) with a novel Naïve Bayes learning algorithm, applied to user command sequence. Our approach is shown to achieve a significant improvement over the above-mentioned performance by Maxion and Townsend [10].

1.5 Thesis Organization

This thesis is composed of the following Chapters.

<u>**Chapter 2: Background Research**</u>

In this chapter, we discuss previous works on the subject and their limitations.

<u>**Chapter 3: Continuous Authentication Based on Learning User Command Sequence (CABLUCS)**</u>

The chapter constitute the core of this thesis. We describe our CABLUCS intrusion detection system, including a discussion on its implementation.

Chapter 4: Experimental Evaluation

Validating the proposed Continuous Authentication Based on Learning User Command Sequence (CABLUCS) scheme is of course an essential part of this research work. In this chapter, we describe the experimental setup as well as the performance parameters and the obtained results.

Chapter 5: Conclusion

We conclude our work and present future possible works that can be done to extend the scope of the content of this thesis.

Chapter 2: Background Research

This chapter discusses related works on intrusion detection systems. Common methods and models employed within this field of research are discussed, as well as related research challenges. Finally, our new approach is contrasted against these related works.

2.1 Intrusion detection System Approaches

Various design approaches to Intrusion Detection Systems (IDSs) have been proposed in the literature, as well as a few attempts to produce taxonomies of IDSs [11], [12; 13; 14; 15; 16; 17], Typically, IDSs can be classified into three categories: sensors, detectors, and positive intrusion handlers (not including false alarms) [11]. Researches focus their attention mostly on detector entity along the system characteristics. There are currently two major types of detection approaches: Anomaly Detection and Signature Detection (Misuse Detection).

Anomaly Detection is based on abnormal behavior. It relies on self-learning for the purpose of detecting abnormal behaviors. A drawback of such system is that some behaviors may not be undesirable, leading to a high false positive rate [11]. Self-learning systems are broken down into two categories: non-time series systems and time series ones. Non-time series systems use stochastic models to determine what a normal behavior would be disregarding time constraints. On the other hand, time-series systems determine normality based on techniques such as Markov models, artificial neural networks, to name a few [11; 18]. As instance, sequence learning for anomaly detection is an example of approach that records the normal working state of a system (i.e. the system's call traces, network packet traces, resource consumption patterns) with regards to its user, then uses this dataset to differentiate between normal and suspicious behavior, by comparing expected behavior patterns with lively detected behavior patterns. The goal is to produce a cost-effective and preeminent model that can, swiftly and appropriately be managed by a certain mechanism. This latter criteria is important because once these systems are deployed in large organizations, the datasets can be very large and in some cases unreliable in terms of timely masquerade detection.

Misuse detection systems are based on previously known intrusion attempts that are fairly common; therefore they may not be reliable in terms of catching new malicious behaviors. However, these systems can be used to detect sequences of instructions that violate the security policies. They make use of rule sets to distinguish behaviors; thus are unable to detect violations that are unknown to these rule sets.

2.2 Quantitative Modeling Methods

For quantitative modeling purpose, artificial neural networks are considered as an important class of tools [19]. These types of systems or computing models have been applied to various problems in many different areas, particularly for identifying the fundamental relationships among a set of variables or patterns in the data [19]. Two important characteristics of these systems are: parallel processing of information and learning and generalizing from experience.

2.2.1 Naïve Bayes Classifier

In addition to artificial neural network, Bayesian learning algorithms have been used as a tool of choice for modeling various systems [19]. The Bayesian learning concept consists in inferring a set of parameters of a predefined model from the information contained in some data.

The Naive Bayes classifiers are among the most successful known class of Bayesian learning algorithms, for learning to classify text documents [20; 21]. The Naïve Bayes classifier is also widely used to detect and classify spam [23; 24] and many other unwanted electronic documents.

The Naïve Bayes classifier is directly related to tasks where each instance x is described by a combination of attribute values. The target function $f(x)$ can represent any available value from a finite set V. Given a set of training examples of the target function, the algorithm can classify a new instance. More precisely, given the new instance tuple of attribute values $(a_1, a_2 \dots a_n)$, the classifier indicates the most probable target value V_{MAP} as follows:

$$V_{MAP} = \text{argmax}_{v_j \in V} P\left(v_j | a_1, a_2 \dots\ a_n\right) \qquad (1)$$

By applying the Bayes theorem, Equation 1 can be re-formulated as:

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)}$$

$$= \operatorname{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \qquad (2)$$

Given the training data, an estimation can be made using the two terms $P(a_1, a_2 \dots a_n | v_j)$ and $P(v_j)$. In order to evaluate $P(v_j)$, we calculate the frequency of which the target value v_j appears in the training examples.

The Naïve Bayes classifier is built on the assumption that the tuple of attribute values $(a_1, a_2 \dots a_n)$ is conditionally independent given the target value [20]. Therefore, this naive assumption indicates that $P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$ and hence the Naïve Bayes classifier is expressed as:

$$V_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j) \qquad (3)$$

The Naïve Bayes classifier has a specific characteristic that is different from other learning algorithms. The hypothesis is evaluated by examining the frequency of different data combinations throughout the training examples and without the need of querying. A comprehensive description of this machine learning algorithm can be found in [20]. The procedural steps required to implement the Naïve Bayes classifier is detailed in Chapter 3.

2.2.2 Intrusion Detection Systems in Conjunction with Naïve Bayes Classifier

Typically, the Naïve Bayes classifier is used to classify text documents [25]. However, a successful implementation of a Naïve Bayes classifier in an intrusion detection environment has also been presented [26; 27; 6; 10; 28]. Due to the nature of this learning algorithm, it is natural to employ it in an application where the learning involves strings of text, such as command-lines. A description of procedures involved to implement such machine learning algorithm in an IDS based on learning user command-line sequences is given in Chapter 3.

2.3 Continuous Authentication Based on Learning User Command Sequence

The source of input data used in anomaly detection is normally extracted from different types of user/system input. The most commonly used data source in anomaly based intrusion detection systems involves one of, mouse dynamics [4; 29], keystroke dynamics, system processes [30; 31; 32; 33; 34], and/or command line sequence. In most literatures that involve learning command line sequence, the data is produced using UNIX or UNIX-like operating systems. The most commonly used dataset in this field, is the work of Schonlau et al. [9]. In the work of Schonlau et al., six different methods were used in order to learn and profile [35] user behaviour, based on their given UNIX command line history. These methods include:

- Uniqueness
- Bayes one-step Markov
- Hybrid multi-step Markov
- Compression
- IPAM (Incremental Probabilistic Action Modeling)
- Sequence-match

The *uniqueness* method is established based on the command frequency in the training data. A command line that is not witnessed in the training data is deemed to be malicious. Commands that have a low frequency in the training data will demonstrate a higher indication of malicious behaviour.

The *Bayes one-step Markov* method is based on the concept of single iterations between commands. The system will compare the given sequence of iteration probabilities to previously known iteration tendencies and determine the legitimacy of the given user.

The *hybrid multi-step Markov* method is based on the n^{th}-order Markov chain and a given model that determines the proportionality of commands that were not witnessed in the training data.

The *compression* method is based on generating reversible maps for the data in correspondence to a representation that utilizes less storage than the original. New input from the user is compressed and compared to the given maps and tested for legitimacy based on the compression rates.

The *IPAM* (Incremental Probabilistic Action Modeling) method is based on one-step command iteration probabilities with regards to a given training data, while continuously expanding and updating its arrangement.

The *sequence-matching* method is based on determining the similarities between the ten most recent commands of a given user in comparison with a user's profile. The following table demonstrates the results achieved in each implemented method by Schonlau et al. [9].

Table 1: Results produced by 6 methods to detect masquerades. Schonlau et al

Method	FRR (%)	FAR (%)	DR (%)
Uniqueness	1.4	60.6	39.4
Bayes one-step Markov	6.7	30.7	69.3
Hybrid multistep Markov	3.2	50.7	49.3
Compression	5.0	65.8	34.2
Sequence-Match	3.7	63.2	36.8
IPAM	2.7	58.9	41.1

Looking at the results produced by Schonlau et al., we can clearly observe that the *Uniqueness* method has the lowest False Rejection Rate (FRR), while lacking a convincing False Acceptance Rate (FAR). Although the FRR value is relatively low, the chance of detecting malicious behaviour (DR) is 39.4%.

Other recent work done in this field includes the work of Maxion and Townsend [10]. Using the Naïve Bayes classifier as their learning algorithm, they have produced encouraging results. The following table demonstrates their final results after testing their method against both the Schonlau et al. dataset (typically denoted as SEA) and the Greenberg [36] dataset.

Table 2: Results produced by implementing the Naive Bayes classifier. [10; 26]

Method	FRR (%)	FAR (%)	DR (%)	Dataset
Naïve Bayes (updating)	1.3	38.5	61.5	SEA
Naïve Bayes (no-updating)	4.6	33.8	66.2	SEA
Naïve Bayes (truncated)	4.7	29.1	70.9	Greenberg
Naïve Bayes (enriched)	5.7	17.9	82.1	Greenberg

The hybrid command sequence (HCS) [37] model is another method used in order to detect malicious behaviour based on learning user command sequence. By using a genetic algorithm, the model profiles users based on recorded sessions. It evaluates users considering multiple command sequence fragments in a single session [37].

Other detection methods with regards to learning command sequences include the use of SVM (Support Vector Machine) [7; 38]. SVM is a pattern recognition classifier. It has shown significant results in terms of producing high detection rates [7; 38]. However the FRR rates are still considered to be high.

The following table demonstrates a comprehensive statistical look at the results gained from implementing each of the mentioned methods with respect to a given dataset.

Table 3: A list of detection methods and their relative results

Method	FRR (%)	FAR (%)	DR (%)	Dataset
Naïve Bayes (updating) [10]	1.3	38.5	61.5	SEA
Naïve Bayes (no-updating) [10]	4.6	33.8	66.2	SEA
Uniqueness [9]	1.4	60.6	39.4	SEA
Bayes one-step Markov [9]	6.7	30.7	69.3	SEA
Hybrid multistep Markov [9]	3.2	50.7	49.3	SEA
Compression [9]	5.0	65.8	34.2	SEA
Sequence-Match [9]	3.7	63.2	36.8	SEA
IPAM [9]	2.7	58.9	41.1	SEA
SVM (RBF Kernel) [7]	9.7	19.9	80.1	SEA
SVM (K-gram Kernel) [38]	14.19	10.39	89.61	SEA
SVM (String Kernel) [38]	23.77	2.6	97.40	SEA
HCS [37]	33.9	1.4	98.6	SEA
Naïve Bayes (truncated) [26]	4.7	29.1	70.9	Greenberg
Naïve Bayes (enriched) [26]	5.7	17.9	82.1	Greenberg

These methods can be evaluated and ranked based on certain ranking functions [39; 9; 6; 26]. These ranking functions depend solely on certain predefined criteria. Depending on the application, the significance of the errors produced by each detection method can vary. The ranking functions involved in determining the quality of a detection method is discussed in more detail in further chapters.

The detection rates (DR) in most of the mentioned methods are very low. In cases where the detection rate is above 90% the FRR rates are above 20-30%. The challenge is to develop a system that would notably reduce the FAR and FRR rates.

In this thesis, we propose a novel IDS termed as Continuous Authentication Based on Learning User Command Sequence (CABLUCS). Our approach consists of using the sequential sampling technique (a novel proposed evaluation technique for continuous authentication [1]) in conjunction with the Naïve Bayes learning, applied to user command sequence, to detect masquerade attacks while learning the behavioural tendencies of a given user. More precisely, the user's normal behaviours are recorded and profiled using the Naïve Bayes classifier. The generated profile is used as their signature, while individuals whose behavioural tendency fails to match the given signature are identified as masqueraders.

Chapter 3: Continuous Authentication Based on Learning User Command Sequence (CABLUCS)

This Chapter constitutes the main contribution of this thesis. Here, we describe the design of our proposed Continuous Authentication Based on Learning User Command Sequence (CABLUCS) scheme. The design space, system architecture, and data collection and processing methodologies are described in-depth. A typical intrusion scenario is also introduced to assess the stated design.

3.1 Design Space

Designing an IDS involves a few challenges, including the methods involved in implementing data collectors, detectors and the different responses offered by the intrusion handlers.

3.1.1 Data Collection

Data collection is an important part of the system. Sensors are placed in appropriate locations within the system, in order to listen to the system's activities and collect important data that will determine whether or not an intrusion has taken place. However, depending on the native system, this task can be rather difficult. Learning user command sequences will be a challenge in terms of being able to analyze this data in such a way as to produce the appropriate analysis of the active situation, which will then serve to take the proper actions. In UNIX based systems or other similar systems, one can take advantage of the input provided by the user within a shell (a separate software program that provides direct communication between the user and the operating system). This data is collected by the operating system and is usually defined by the term *shell history*. Although datasets can be controversial in terms of privacy issues [40], in most cases, shell history is readily available.

The shell history can be used to achieve a basic understanding of the user's common patterns. Typically, the *command-line history* is used when attempting to develop an IDS within a UNIX environment [26; 10; 41; 42]. In 1988, Dr. Saul Greenberg of the Department of Computer Science at the University of Calgary, has collected traces of 168 users using the UNIX C shell (*csh*). These traces correspond to command line data executed by each user and the data was intended to be used for research purposes. This dataset [36] was kindly granted to us and we have used it in this thesis. Running this dataset using a slight modification of the C shell (csh) command interpreter has enabled us to duplicate Dr. Greenberg's data collection method, which is crucial in the context of this thesis in order to accomplish our data collection objectives.

Data collection in correspondence to different categories and groups of subjects is of importance. In order to relate different behaviours, it is important to have certain understanding of the given subjects, in which the data is being collected from. In this case, subjects were 168 unpaid volunteers, either students or employees of the University of Calgary. Subjects are divided by Greenberg into 4 different groups, which include:

- Novice Programmers
    - This group consisted of individuals that had no prior programming experience, minimal knowledge of operating systems or UNIX-like command interpreters. These users spent the majority of their time

learning programming techniques and concepts, while familiarizing themselves with the system's facilities [36].

- Experienced Programmers
  - This group consisted of undergraduate Computer Science students completing their senior years. An understanding of the UNIX environment and moderate knowledge of programming languages were expected from this group. [36].
- Computer Scientists
  - This group consisted of Computer Science graduates, including the members of the Faculty, researchers and past graduates from the Department of Computer Science [36].
- Non-programmers
  - This group consisted of members that mostly concentrated on the use of word processing applications. These members had little or no experience in programming languages or no knowledge of the UNIX environment [36].

It should be acknowledged that subjects were assigned as members of these groups given their current agenda at the University of Calgary. Therefore, the assumption that all members fit the given criteria of a particular group cannot be made thoroughly. From the months of February 1987 through June 1987, command line data was continuously collected on site, at Dr. Greenberg's laboratory. The collected data had a specific formatting that includes different annotation for explaining certain situations that had incurred during data collection. These annotations along certain drawbacks to the collected data are discussed next.

3.1.2 Greenberg's Data Organization

The given data was organized through hierarchal folders. The base folder was named as *unix_data*. This base folder was composed of five subfolders. Four of these subfolders corresponded to the groups of subjects, which themselves stored all command trace data of every subject (e.g. novice programmers, experienced programmers, computer scientists, and non-programmers).

The fifth subfolder, *showerrorcode*, included a C program, which was designed to provide explanations for the error codes that were generated by users when executing certain command-lines.

The following tables give a brief description of the attributes used in this dataset.

Table 4: Login session record

Code	Description	Example
S	Start time of the login session	S Thu Sep 20 14:23:32 2008
E	End time of the login session	E Thu Sep 20 19:11:12 2008

Table 5: Command line record

Code	Description	Example
C	The line entered by the user	C gedit document.txt&
D	The current working directory	D /home/user/documents/

A	The alias expansion of the previous command (if any)	A NIL
H	The line entered had a history expansion in it (True or Nil)	H NIL
X	The error detected in the line by csh (if any). A following letter and number code indicates the category and actual error type.	X NIL
T	The time the command line was executed by the command interpreter.	T Thu Sep 20 16:11:43 2008

3.1.3 Reproducing Greenberg's Methodology

In order to understand the data collection mechanism used by Greenberg et al. [36], we had to reproduce it using the software package that was kindly granted to us. C shell is a UNIX command interpreter that introduced new features such as aliases and command history. This justifies (in some sense) why Dr. Greenberg used this shell in order to collect command line history.

In order to achieve a consistent duplication of work, we also use C shell to reproduce Dr. Greenberg's data collection mechanism. Although this tool may be considered inadequate compared to more recent command interpreters, it is important to note that it serves its purpose in the case of collecting command line history. In order to reproduce Dr. Greenberg's data collection mechanism, a copy of the C shell source code was acquired from one of the Ubuntu's available repositories within our laboratory[1].

[1] The Distributed Applications and Broadband Network laboratory (DABNEL), Department of Computer Science, Ryerson University, Toronto, Canada

After making appropriate modifications to the C shell code provided by Greenberg et al. [36], sensors were placed accordingly in order to produce similar results. In addition to Dr. Greenberg's selection of attributes, a new attribute called *Time* (denoted *T*) is introduced. This attribute is used to determine the system time that the command line was executed by the command interpreter. Although Dr. Greenberg had included the attributes S and E which denote the starting and the ending time of each session respectively, it appeared important for us to track the displacement time ΔT of each command line. This is done in order to gain a better understanding of the user's intentions.

The output of our modified *csh* scheme compared to that of the Greenberg dataset scheme is captured in Table 6.

Table 6: Greenberg's reproduced dataset sample

Greenberg Sample Data	Reproduced Sample Data
C *ls* **D** */home/XXXX/documents/* **A** *ls –la* **H** *NIL* **X** *NIL*	**C** *ls* **D** */home/XXXX/documents/* **A** *ls –la* **H** *NIL* **X** *NIL* **T** *Thu Sep 20 16:11:43 2008*

Few drawbacks of the Greenberg's approach for data collection [36] are as follows.

- Given the structure of the implementation, the "details of history directives were not recorded" [36]. However, there are indications of history being used and the command-line that was retrieved.

- It is important to acknowledge that the system was unable to capture all user activity. This mainly relates to software packages that are invoked by the user, where the command line is no longer used (e.g. *emacs* versus *ls*).
- The command line executed does not necessarily determine the program that was actually invoked. Because of the many ways a program can be invoked (e.g. through an alias or a script). Although the records for the alias used are included in the dataset, the dataset fails to compensate for events where an alias is used to invoke another alias.

3.2 System Architecture

Similarly to many existing intrusion detection systems, our architecture is composed of sensors, detectors and intrusion handlers as depicted in Figure 1.

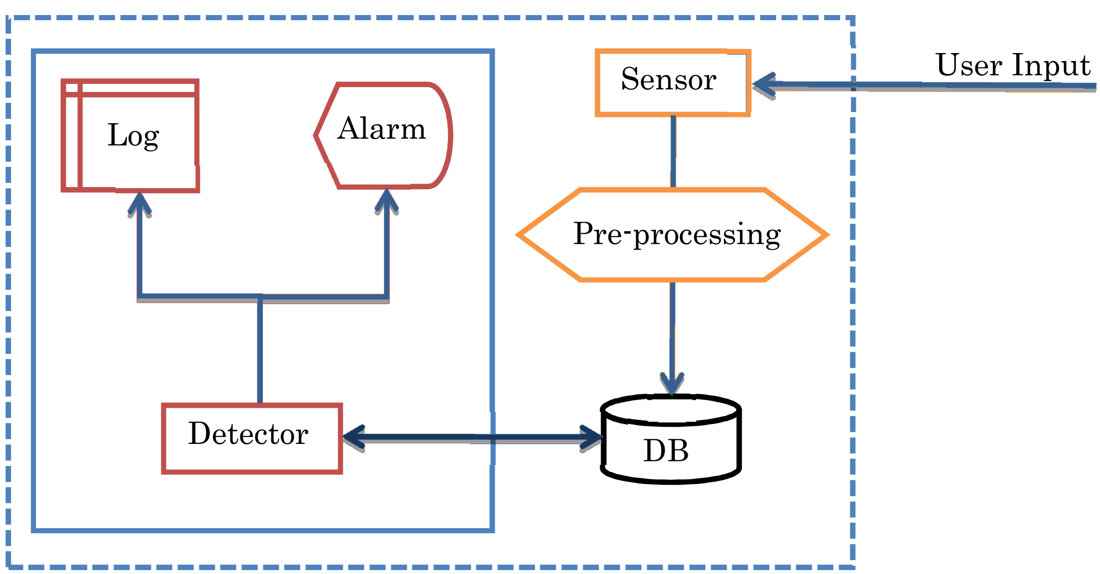

Figure 1: System architecture

The user input is captured by a sensor and restructured into a desired format (by undergoing a pre-processing step). The output of the pre-processing step is then placed in an incoming pool in the database and made ready for use. The detection mechanism (so-called Detector) will then deploy its intrinsic evaluation algorithm (in the form of a Naïve Bayes classifier) and a decision of an acceptance or a rejection will be made. If the input is rejected, the system will be alarmed and appropriate actions will be taken. Meantime, the system will keep maintaining a Log file that stores all the activities that have been running the system's operations. In case of an acceptance, the users will continue to use their concessions while the system will continue to authenticate their behaviours.

3.3 Implementation

This section describes the implementation of the Continuous Authentication Based on Learning User Command Sequence (CABLUCS) design approach. The Greenberg dataset [36] is used as the source for generating user profiles and user inputs. The Naïve Bayes classifier is used in conjunction with a new evaluation technique based on continuous authentication [1], namely the sequential sampling technique, to classify and evaluate a given user.

3.3.1 Data Structure

The Greenberg dataset is used as the source of data for this implementation, for training and testing purposes.

The data is in its raw format. It consists of multiple folders, each representing a separate category of subjects. The categories are defined by each user's level of experience or position (i.e. novice programmers, experienced programmers, computer scientists, and non-programmers) within the set of test subjects. For the purpose of our implementation, the category under which the user falls into is ignored. This information may be useful for the implementations of certain IDSs. However, due to the nature of our approach, the user categories are deemed to be extraneous. A discussion on future works that can incorporate a primary and secondary levels of classification in a multi-category based user environment is given in the Conclusion Chapter.

Each user is separated with a text file that contains the recorded command line history of the user. Each session is separated by a start and end time stamp. Figure 2 illustrates a single user session.

```
S Wed Feb 18 16:37:25 1987

E Wed Feb 18 16:56:22 1987


C date

D /user/srdg/xxxxx

A NIL

H NIL

X NIL


C mail

D /user/srdg/xxxxx

A NIL

H NIL

X NIL


C p audio.mail

D /user/srdg/xxxxx

A page audio.mail

H NIL
```

Figure 2: Greenberg's dataset user session sample

The filename for each user is constructed using the name of the category that the user is part of, and is concatenated with a numerical digit. An example is given in Table 7.

Table 7: Filename structure used in the dataset

User Category	Filenames
Computer Scientist	scientist-1, scientist-2, … , scientist-52

Experienced Programmers	experienced-1, experienced-2, … , experienced-36
Non-Programmers	non-1, non-2, … , non-25
Novice Programmers	novice-1, novice-2, … , novice-55

In this dataset (Table 7), there are 168 users, among which 52 are Computer Scientists, 36 are Experienced Programmers, 25 are Non-Programmers and 55 are Novice Programmers.

3.3.2 Pre-processing the Dataset

Due to the difficulty encountered in using the data in its current raw format, we had to restructure the data into a more manageable configuration. To this effect, the data was pre-processed and restructured into a relational database. This step is vital in order to maintain the relations between the command-lines, sessions, users and the user categories while this process is being completed. During the pre-processing progression, the complete structure and integrity of the data was preserved and tested.

The dataset was restructured into four database tables, referred to as user_types, users, sessions and data (as shown in Tables 8 to 14).

- The *user_types* table (Table 8) is created in order to maintain the different user categories involved in the dataset. This table can also be used to merge other datasets of similar nature into Greenberg's dataset by introducing new sets of categories.

Table 8: User types (user_types) database table schema

Field Name	Description
utid	This field maintains the id for each user category. This id is used in other tables to indicate which category a particular user belongs to.
type	This field defines a two character identification of a user type. (i.e. 'cs' for Computer Scientist)
description	This field is used to maintain a brief description of each user type. It is used for the purpose of describing the types in English for individuals who are new to using the Greenberg dataset.

- The *user_types* table (*Table 9*) maintains the four records that directly correspond to the four different categories involved in the dataset.

Table 9: User types (user_types) database table sample data

utid	type	description
1	cs	Computer Scientist
2	ep	Experienced Programmer
3	np	Non-Programmer
4	nv	Novice Programmer

These values in Table 9 are static. They are used as a reference point to indicate which category a user belongs to.

- In order to maintain the identity of each user, the *users'* table (Table 10) is introduced. This table holds the basic records of all 168 users involved in the dataset. The *uid* field is used consistently throughout the database in order to maintain the data integrity.

Table 10: Users (users) database table schema

Field Name	Description
uid	This field maintains the id for each user. This id is used in other tables to identify each user.
utid	This field is used to indicate which category a particular user belongs to within the *user_type* table.
greenberg_name	This field maintains the filename used in the Greenberg dataset that

	corresponds to the given user. (i.e. scientist-1)

- The *users* table (Table 11) can also be used as a quick reference in order to distinguish between the different users while manually traversing through the database. This table plays a major role in keeping the integrity of the restructured dataset. Manipulation of this table can compromise the integrity of the overall dataset and hence it is only used as a reference.

Table 11: Users (users) database table sample data

Uid	Utid	greenberg_name
1	1	scientist-1
2	1	scientist-2
3	2	experienced-1
4	3	programmer-1
5	4	non-1

- Sessions were handled with two lines at the beginning of each of the sessions. Each session's start and end time was parsed and converted into a UNIX Timestamp, and then inserted into the *sessions* table (Table 12). It is important to notice that once the newly converted timestamp is made available, date, time calculations and manipulations become simpler.

Table 12: Sessions (sessions) database table schema

Field Name	Description
sid	This field maintains the session id for each session within a user's stored data. This id is used in other tables to identify a session.
uid	This field is used to indicate the user that this session belongs to.
start	This field indicates the start time of a session. The values are kept in UNIX Timestamp format.
end	This field indicates the start time of a session. The values are kept in UNIX Timestamp format.

A UNIX Timestamp is an integer which indicates the number of seconds elapsed since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970 [43]. For example, the UNIX Timestamp of 540682645 is the equivalent of February 18, 1987 at 3:37:25 pm. Every data item (command line) belongs to a particular session. Using the *sessions* table, we can immediately identify such session's start and end time/date as shown in Table 13.

Table 13: User sessions (sessions) database table sample data

sid	Uid	start	end
1	1	540682645	540683782
2	1	540742586	540744368
3	2	544203016	544203669
4	2	544424264	544438849
5	2	544449748	544459510

- The *data* table (along with its relations with the *user_types*, *users* and *sessions* tables) holds the entire dataset. It contains 303,628 data items (command-lines) from 168 different users. Its main fields are shown in Table 14.

Table 14: Data items (data) database table schema

Field Name	Description
did	This field maintains the data id for each command line in the Greenberg dataset. This id is used in other tables to identify a command line.
sid	This field indicates which session this command line belongs to. Using this id we can indicate the start and end time of the session.

uid	This field is used to indicate the user that this command line belongs to.
order	This field is used to maintain the order in which command lines in a session were entered.
command	This field contains the entire command line.
directory	This field contains the current working directory in which the command line was executed.
alias	This field will indicate if the command line was in fact an alias to execute another program. It will contain the command in which the alias is executing otherwise a NIL value will be given.
history	This field indicates whether or not History was used to execute this command line.
error	This field indicates if an error occurred during the executing of this command line.

- The *data* table (Table 15) encompasses the completely restructured dataset. In its new format, the data can be searched, manipulated and tested at a higher rate of efficiency. Furthermore, this higher rate of efficiency can also be transferred onto any available platforms.

Table 15: Sample command lines in the 'data' table from User 1

did	sid	uid	order	command	directory	alias	history	error
1	1	1	1	Date	/user/srdg/xxxxx	NIL	NIL	NIL
2	1	1	2	Mail	/user/srdg/xxxxx	NIL	NIL	NIL
3	1	1	3	p audio.mail	/user/srdg/xxxxx	page audio.mail	NIL	NIL
4	2	1	1	Ls	/user/srdg/xxxxx	/bin/ls –Fs	NIL	NIL

In order to gain a better understanding of the simple statistical features of the dataset, the *insert_report* table (Table 16) is introduced. This table gives a general understanding on the number of command-lines, aliases, use of history and errors, which are involved in the dataset.

Table 16: Data report (insert_report) database table schema

Field Name	Description
id	This field contains the report id.
filename	This field indicates the filename for which this report was generated.
uid	This field is used to indicate the

	user that this report belongs to.
commands	This field indicates the number of command lines that were executed by the given user.
history	This field indicates the number of times the user resorted to using its history database.
errors	This field indicates the number of times an error occurred while a user executed its command lines.
aliases	This field indicates the number of times the user resorted to using an alias.
lines	This field indicates the number of lines in the filename.

Primarily, the generated reports (Table 17) allow us to test the integrity of the database by comparing the results to its raw counterparts. The general statistical understanding of the dataset will also allow us to plan our implementation in a more meaningful way.

Our new knowledge of the data allows us to make better choices for the future. For example, based on this, we can determine which users will be beneficial for our testing purposes. Hence, if a user does not have sufficient amount of data items, then it becomes difficult to process a complete set of tasks in most training environments. The reports will also be helpful to index users who have made use of their history, aliases or are disposed to make errors in their command lines or vice versa. This type of information can become crucial in many research related tasks, particularly tasks that involve datasets being used in a controlled environment. The generated statistics for the Greenberg's dataset is made available in Appendix A: Generated Statistics for the Greenberg Dataset (Ordered by Commands).

Table 17: Sample reports in the 'insert_report' table from 5 different users

id	filename	uid	commands	history	errors	aliases	lines
1	scientist-1	1	1856	54	111	761	11792
2	scientist-10	2	2024	77	120	730	12658
3	scientist-11	3	205	0	13	0	1380
4	scientist-12	4	2499	53	52	1162	15412
5	scientist-13	5	3593	357	118	204	21988

3.3.3 Naïve Bayes Classifier

The Naïve Bayes classifier is used as a learning mechanism (Detector box of Figure 1) in order to understand the available sample data.

The sample data is used in order to train the system and to familiarize it with possible outcomes. In principle, the available sample dataset will determine our expectations in anticipating accurate results in detecting legitimate versus illegitimate user sessions. The quality and the scale of available training data to the system will dictate our confidence in its results.

The basic algorithm involved in recognizing different predefined classifications with the use of the Naïve Bayes classifier involves two procedures, namely the Naïve Bayes learning mechanism and the Naïve Bayes classification. We first train our classifier with the available training data by using the Naïve Bayes learning mechanism. Once our learning procedure is completed, we classify new sets of input using our trained system.

In order to proceed with the learning mechanism, we first determine certain attributes that directly influence the Naïve Bayes learning algorithm. As previously stated, the Naïve Bayes classifier is typically used to classify text documents such as electronic news articles [44; 45] or to classify spam, websites, documents, to name a few [23; 24]. But in the case of intrusion detection and profiling of legitimate users (that we deal with in this thesis), the classification must be achieved differently.

Here, our approach for classification consists in considering the possible outcomes of an IDS, i.e. the detection of a masquerader (illegitimate user) or the detection of a legitimate user. Therefore, we determine that our target value is either *legitimate* or *illegitimate*. Based on our training data, we can then determine the characteristics of a legitimate user. Yet, we do not have a direct understanding of what characterizes an illegitimate user. Naturally if a legitimate user is not detected, then the user must be considered as illegitimate. However the Naïve Bayes classifier requires us to have certain understanding on all defined classifications in our dataset prior to the detection. The absence of the training data for illegitimate users prevents us from gaining any understanding on the behavioural tendencies of a potential masquerader. Therefore, due to this shortcoming, we are obliged to practice the common adaptation [10] of using any training data available to us that does not belong to the potential legitimate user and consider this as the training data for an illegitimate user. Certainly, such an assumption may have certain consequences that may skew the final classification results. Depending on the scale of the dataset, an immediate consequence based on this assumption is as follows. Due to the nature of the Naïve Bayes classifier algorithm, the number of incidences in any classification is of importance. For instance, in our dataset of 168 users, the available training data for one legitimate user compared to its illegitimate counterpart (167 users) can potentially disrupt the classification. The reasoning behind this claim lies solely in the nature of the Naïve Bayes classifier algorithm.

Once the target values for the classification have been decided, we need to traverse through the training data and identify each element as a member of each target value. We then introduce the set V to represent all the possible target values v_j. In order to begin the learning process we also introduce the set $Vocabulary$. This set includes all the distinct words (command lines) that are available within the training data. The Vocabulary set can be regarded as our dataset dictionary.

Two probability terms $P(v_j)$ and $P(w_k|v_j)$ are used as the driving forces of the learning mechanism within the Naïve Bayes classifier. The term $P(v_j)$ also known as the prior probability, represents the probability of the target value v_j occurring within the available training data. The term $P(w_k|v_j)$ represents the conditional probability, that a randomly selected word (command-line) from the training data belonging to the target value v_j will be the word w_k.

Once learning is completed, we can then classify new sets of input using the following Equation:

$$V_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i|v_j) \quad (4)$$

The procedural steps that are required to train (hence produce the profile of a user) and classify user command sequences using the Naïve Bayes classifier are discussed next.

3.3.3.1 Profiling Users

In order to test the legitimacy of a user session, we must first develop an understanding of what constitutes a legitimate user session. To this effect, it is required to make use of the available training data which corresponds to the normal working state of any particular user within the system, then, develop a profile that accurately represents such user. This can be achieved by using a Naïve Bayes learning mechanism. The detailed procedural steps involved in implementing our Naïve Bayes learning mechanism with our dataset, in order to create a set of independent and distinct user profiles, is described as follows.

Our target value set V is defined as as:

$$V = \{legitimate, illegitimate\} \quad (5)$$

As previously mentioned, the Naïve Bayes classifier requires the evaluation of the two probability terms $P(v_j)$ and $P(w_k|v_j)$ with regards to the training data, in order to successfully classify the new input.

To calculate $P(v_j)$ with respect to our new target value set V, we evaluate the following:

$$|data| = \sum_{j \in V} |data_j|$$

$$= |data_{legitimate}| + |data_{illegitimate}| \quad (6)$$

where $data$ is our training data.

$$P(v_{legitimate}) = \frac{|data_{legitimate}|}{|data|} \quad (7)$$

$$P(v_{illegitimate}) = \frac{|data_{illegitimate}|}{|data|}$$

$$= 1 - P(v_{legitimate}) \quad (8)$$

We estimate the conditional probability $P(w_k|v_j)$ the same way as done in [20], i.e.

$$P(w_k|v_j) = \frac{n_k+1}{n+|Vocabulary|} \quad (9)$$

where $w_k \in Vocabulary$,

n_k *is the number of times the command line* w_k *occurs in* $data_j$ and

n *is the total number of distinct command lines in* $data_j$

Having the preceding algorithms outlined, we can begin to document and *profile* every user within our training dataset.

In order to implement a successful training session, it is required to have a sufficient amount of data for each user's profile. Lack of sufficient data will directly contribute to inaccurate results. Our initial confidence in the system relies on the quality and the availability of a rich representation of a user's behavioural tendencies in a form of a dataset. By investigating the general statistical information (See Appendix A) regarding the available dataset, we can make certain decisions in regards to possible usability and suitability for each user's data and their potential candidacy for our training sessions.

Retrieving the generated statistical information allows us to consider each user as a potential candidate for a training session. As a rule of thumb, we consider each user that has equal or greater than 1500 command lines in its data pool as such candidate. After applying this rule to our 168 user dataset, we witness that 75 of the users meet the requirements.

These 75 users will be denoted as *victims.* We then use the available data associated with each victim to build our profiles. To this effect, we have considered the first 1000 command lines of each victim as the source for our training data. Once the victims have been identified and their designated training data has been extracted, we proceed to gain a more detailed understanding on the overall commuted training data. Based on the final commuted training data, a vocabulary is built, which consists of all distinct command lines used by all victims along with their number of occurrences. This vocabulary is used as a reference to build each victim's profile.

Along with the vocabulary, each user's distinct command lines are identified and recorded in a separate table. This table contains all the distinct command lines witnessed in the training data that belong to each user, along with their number of occurrences.

In order to build a profile for every user, all command lines witnessed in the vocabulary are coupled with each user. A user profile consists of all the terms (command-lines) within the vocabulary along with the probabilities $p_{legitimate}$ and $p_{illegitamate}$ associated with the term with respect to the user. These probabilities are the representations of the results gained from evaluating the probability term $P(w_k|v_j)$, where w_k represents each command line in the vocabulary. $p_{legitimate}$ indicates the probability that the given command line belongs to the user, where $p_{illegitimate}$ indicates the probability that the given command line belongs to other users within the training data.

The vocabulary associated with our training data consisted of 17,982 terms (command lines). Therefore, each user's profile consists of the same number of terms along with their associated probabilities. After the completion of the learning mechanism, 1,348,650 records were generated, representing the profile information for the 75 victims. The following table (Table 18) represents a small segment of a profile belonging to one of the victims.

Table 18: Small segment of a given profile

uid	Command Line	$p_{legitimate}$	$p_{illegitimate}$
78	ls	0.40874959414524	0.43698392003477
78	fg	0.12359203459912	0.12143113580107
78	e	0.11171047003469	0.052685310845613
78	lpq	0.092699966731619	0.046794399410464
78	bye	0.059431585951238	0.023954223940219
78	myada	0.052302647212585	5.5574636180644e-07
78	e conq.a	0.052302647212585	5.5574636180644e-07
78	ada –m conq.a	0.049926334299701	5.5574636180644e-07
78	a.out	0.042797395561047	0.018062312505071
78	who	0.03329214390951	0.040570040158232
78	purge	0.03329214390951	0.0016122201956005
78	rwho | more	0.035668456822394	0.0043909520046327
78	e queens.a	0.016657953519319	5.5574636180644e-07

In Table 18, each command line is represented with its associated $p_{legitimate}$ and $p_{illegitimate}$. In the next section, we discuss the classification process involved in determining whether a command line is classified as *legitimate* or *illegitimate*.

3.3.3.2 Classifying Users

Once we have established the probability values $p_{legitimate}$ and $p_{illegitimate}$ for all the terms within the vocabulary with respect to each victim, we can proceed to classify new terms (command-lines). As previously mentioned, ideally $p_{illegitimate}$ should represent the probability that the command line belongs to masqueraders. However, due to the absence of such data, it is required to build the probability from other sources. Although it may seem that the data is not authentic, it will nevertheless give a fair representation of what a legitimate user is not, which is the sole purpose of creating the counterpart classification to the target value *legitimate*. As a result of this assumption,

$P(v_{illegitimate})$ will always dominate $P(v_{legitimate})$, since the available data ratio is 74:1. Consequently, the probability term $P(v_j)$ will dominate the Naïve Bayes classification, which in turn, will always classify inputs as $illegitimate$. In order to compensate for the dominating factor of the probability term $P(v_{illegitimate})$, we have to make the assumption that the likelihood of a masquerader will be equal to that of the legitimate user (victim). Thus, the consideration of the term $P(v_j)$ can be eliminated from the classification process.

As an example, given the sequence of command lines described in the set $\{ls, cd\ classes, cd\ cps511, pico\ deadlines, exit\}$, the Naïve Bayes classifier will determine the classification based on each command-line's predetermined probability. For instance, if the probability distribution for an arbitrary user who has entered the command-lines within the given set is captured in Table 19, the Naïve Bayes classifier can be used to classify the set as whether it is $legitimate$ or $illegitimate$ compared to the given sample user profile. The Naïve Bayes classifier makes the assumption that each element in the set is independent, which explains the naïve nature of the classifier.

Table 19: Probability distribution for an arbitrary user

Command Line	$\boldsymbol{p_{legitimate}}$	$\boldsymbol{p_{illegitimate}}$
ls	0.3121	0.4351
cd classes	0.3123	0.0922
cd cps511	0.0422	0.0311
pico deadlines	0.0911	0.0022
exit	0.0332	0.0021

In this example, to determine the classification, we perform the following calculations:

$$V_{NB} = \text{argmax}_{v_j \in V} \prod_{i \in positions} P(a_i | v_j) \quad (10)$$

$$\prod_{\text{i} \in \text{positions}} \text{P}(\text{a}_\text{i} | \text{v}_{\text{legitimate}}) = \text{P}(\text{ls} | \text{v}_{\text{legitimate}}) \text{P}(\text{cd classes} | \text{v}_{\text{legitimate}}) \ldots \text{P}(\text{exit} | \text{v}_{\text{legitimate}})$$

$$= (0.3121)(0.3123)(0.0422)(0.0911)(0.0332)$$

$$= 1.24 \times 10^{-5} \quad (11)$$

$$\prod_{\text{i} \in \text{positions}} \text{P}(\text{a}_\text{i} | \text{v}_{\text{illegitimate}}) = \text{P}(\text{ls} | \text{v}_{\text{illegitimate}}) \text{P}(\text{cd classes} | \text{v}_{\text{illegitimate}}) \ldots \text{P}(\text{exit} | \text{v}_{\text{illegitimate}})$$

$$= (0.4351)(0.0922)(0.0311)(0.0022)(0.0021)$$

$$= 5.76 \times 10^{-9} \quad (12)$$

Since $5.76 \times 10^{-9} < 1.24 \times 10^{-5}$, it is concluded that the set is classified as *legitimate*, i.e. the given set is recognized as legitimate input produced by the owner of our sample user profile.

In our implementation, we use the Naïve Bayes classifier to classify each new input that is witnessed in our evaluation. However, the final decision to accept or reject a user is made by using the sequential sampling method, as described in the next section.

3.3.3.3 Continuous Authentication and the Sequential Sampling Technique

In order to appropriately evaluate the legitimacy of a particular user against a certain profile, an evaluation technique is required, where the crucial characteristics of the system are recognized. In an IDS, decisions can be made using any number of input values, regardless of the relevancy of the data in question. However, the accuracy of the decisions can be questionable. In order to maintain a certain confidence rate in our system, we have adopted a recently proposed evaluation technique [1], which can be considered as a pioneer method for achieving continuous authentication based on learning biometrics data. In this thesis, this technique has been adapted for use in the case of user command sequence.

The term continuous authentication [46; 47; 48] refers to a system where the authentication process is continuously active throughout the session. Typically, the behavioural patterns of the user are tested and evaluated against a predefined signature. The data rates for the input streams are unknown. Hence, the authenticity of the user is tested continuously as data becomes available.

Due to the nature of our application, the input data flow rate is also unknown; hence an evaluation technique is required that can systematically adapt itself to the rate at which new input data are available. In order to evaluate the legitimacy of a user, we use the above-mentioned evaluation technique (sequential sampling technique [1]). This method has previously been used to evaluate the legitimacy of a user based on mouse dynamics. This is the first time that the method is being used for evaluating command line sequences.

The sequential sampling technique is a dynamic sample size decision technique. Typically, a classical sampling is used to make the decision on the validity of a particular user [1]. In classical sampling, the sample size is predetermined and decisions are only made when the end of the data collection procedure is met. The sequential sampling method was introduced to compensate for certain shortcomings of the classical sampling approach when dealing with continuous authentication. In a classical sampling approach, a decision cannot be made, until there is a sufficient amount of data available. Hence, the system can be vulnerable during that time, and thus the system's TTA (Time-to-Alarm) can be significantly influenced by such a method. By using the sequential sampling method, the decision making is active during data collection. Therefore, decisions can be made as new input is presented, and data collection and analysis can be done simultaneously [1].

In order to utilize the sequential sampling technique, it is required to first model a sampling plan which consists of three regions, namely, *Accept*, *Reject* and *Continue*. While the data is being collected continuously (as shown in Figure 3), the sample size is incremented accordingly until a decision is made.

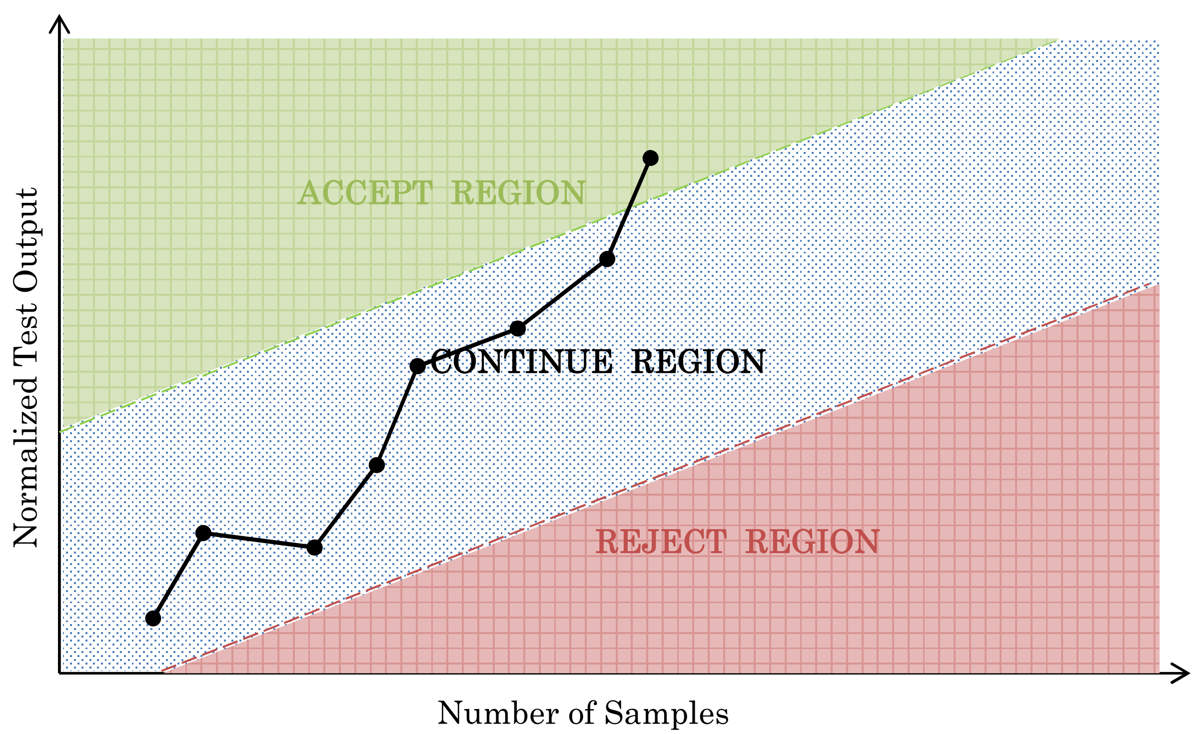

Figure 3: General sampling plan

The sequential sampling technique will continuously test the null hypothesis as the number of test inputs increases. The amount of collected data within each iteration is based on a predetermined sample size, while the sample size itself is determined based on the application in which the sequential sampling technique is used. Depending on the application the sample size can range from collecting only a single data item to collecting a large set of data items.

The sampling plan is developed in accordance with the following parameters [1]:

α = acceptable type I error (false positive)

β = acceptable type II error (false negative)

p_1 = lower threshold limit (as proportion)

p_2 = higher threshold limit (as proportion)

A decision is only made when a normalized confidence rate value enters one of the decision regions (i.e. accept or reject). The acceptance and rejection lines are expressed as follows [1]:

$$N_{CR} = h_2 + sn \qquad (13)$$

$$N_{CR} = -h_1 + sn \qquad (14)$$

where N_{CR} is the normalized value of the confidence ratio computed for test number n.

$$N_{CR} = CR_n \times \frac{n}{100} \qquad (15)$$

The parameters h_1, h_2 and s can be expressed as follows:

$$h_1 = ln\frac{1-\alpha}{\beta} \div ln\frac{p_2(1-p_1)}{p_1(1-p_2)} \qquad (16)$$

$$h_2 = ln\frac{1-\beta}{\alpha} \div ln\frac{p_2(1-p_1)}{p_1(1-p_2)} \qquad (17)$$

$$s = ln\frac{1-p_1}{1-p_2} \div ln\frac{p_2(1-p_1)}{p_1(1-p_2)} \qquad (18)$$

It can be observed that equation 13 and 14, follow the basic principles of a straight line. Therefore, the variables $h_1$and h_2 influence the distance between the two lines and s represents their slope [1].

Chapter 4: Experimental Evaluation

This Chapter discusses the performance evaluation of the Continuous Authentication Based on Learning User Command Sequence (CABLUCS) scheme proposed in this thesis. This includes the evaluation approach, the experiments setup and operational aspects, and finally a description of the results obtained.

4.1 Challenges

The goal of a IDS is to accurately determine the legitimacy of a given user in a timely fashion. To this effect, several parameters can been predefined and then used to measure the efficiency and accuracy of the system in terms of detection rate. In this thesis, we have considered the following parameters.

- The Detection Rate (DR): this is the rate at which the system can successfully detect an intrusion, in the event of a masquerade attack.
- False Reject Rate (FRR) and False Accept Rate (FAR): Typically, two types of errors arose when decisions are made using experimental data in an IDS. They are symbolized as Type I and Type II errors, respectively
    - A Type I error implies that a reject decision has inappropriately been made, indicating that a false rejection has occurred. In this case, the rejection of a legitimate user has happened. The FRR represents the percentage in which the system has falsely rejected a legitimate user.
    - A Type II error implies that an accept decision has inappropriately been made. This type of error indicates that a false acceptance has occurred. In this case, the acceptance of a masquerader as a legitimate user has happened. The FAR represents the percentage in which the system has falsely accepted a

masquerader as a legitimate user. It is also an indication of the rate at which the system has been compromised.

- Time-to-Alarm (TTA) and Mean-Time-to-Alarm (MTTA): The accuracy of the detection mechanism is an important factor. However, if the decision is not made in a timely fashion, the system can be jeopardised. The TTA indicates the time elapsed until the masquerader was detected and the MTTA represents its responsiveness.

In the design of an IDS, attempting to minimize the FRR, FAR, and the MTTA rates is a difficult task in the sense that these attributes are loosely related to each other. In order to reduce the MTTA value, decisions must be made faster. However, a quick decision may not be appropriate since this may lead to increased FAR and FRR. The challenge is thus to minimize all attributes while maintaining an efficient and operational system.

4.2 Evaluation Approach and Setup

In order to setup the working environment for our experiments, we have acquired the LAMP software bundle. LAMP was installed on a Dell Workstation, Quad Xeon Processors at 1.86 GHz with 8 GB RAM. LAMP is an open source software bundle that consists of **L**inux, **A**pache HTTP Server, **M**ySQL relational database management system and **P**HP. This combination is generally used to create dynamic, database driven, web applications. LAMP projects tend to be platform independent, hence once developed, they can be executed on most operating systems.

PHP is a scripting language with syntax similar to the C programming language, which is simple to deploy and execute. It is generally bundled for the use of dynamic web programming in relations with MySQL. The setup was configured to be sensitive to errors, while disabling caching and timeouts. Caching is typically used in PHP for the purpose of enhancing the processing time for the re-runs of the same code. Because we are interested in monitoring and differentiating the processing time (in seconds) of the different test cases, we have set this feature to 'disabled'. PHP processing timeouts are usually set to 30 seconds, which is a reasonable time if we are concerned with executing a program that results in producing a simple web page. However, due to the nature of our experiment, we anticipate a much higher of processing time. Therefore, timeouts have also been disabled. In our experiments, we have also configured PHP warning and error settings, to make all potential warnings and errors visible within our apparatus.

Along with PHP, the MySQL server and the Apache HTTP Server also utilize caching and other performance enhancing features as the system adapts itself to its environment. Therefore, for every test case, the servers are reset to their original status and restarted accordingly. As a result, certain tests could not be done simultaneously and longer testing times are required to test different parameters. This procedure is followed to maintain a fair comparison between the different test cases in comparative processing times (measured in seconds). These precautionary steps do not influence the FRR, FAR and MTTA values since these values do not incorporate time (in seconds) as their unit. The MTTA value is based on the number of actions required to make a decision, thus is not based on the processing time (in seconds) since different workstations can produce different processing times while testing the same IDS, but in contrast, they will all output the same MTTA value.

4.2.1 Extracting the Data from Its Raw Format

A PHP script was written in order to extract the data from its raw format into a MySQL database. The script traverses through the different folders looking for files that matched the required criteria. The criteria are set based on the provided Greenberg dataset structure. Once the script has determined that a file meets the criteria, it detects the user and its type. Using the discovered information regarding the user, the script then creates a user record in the database, while documenting the related information.

Once the user has been determined, the script traverses through its given data, scanning for user sessions. Each user session that is found is recorded in the database. The related information on this user is documented, which include the session's start and end time. A function is then used to convert the start and end time to a UNIX timestamp value.

4.2.2 Extracting the Command Line of a Session

Once a session has been determined, the data collection proceeds to extract command lines relative to the given session. Each command line is extracted from the session and is given an order number, which represents the order in which the command line is seen within that session.

Certain string values (i.e. command-line, working directory and alias) are required to be character-escaped in order to meet certain PHP and MySQL compatibility issues. Due to the nature of the Greenberg dataset, each command line is known to be coupled with certain attributes (such as working directory, history, alias and error). In order to retrieve the attributes with respect to the command line, the script is parsed through each item and the necessary information is collected. An audit is also kept on the general statistical information of each user (i.e. the number of command lines, the errors, to name a few).

Due to certain hidden characters within the Greenberg dataset, several string comparisons in each user session tended to fail. By trimming whitespaces and other unknown hidden characters, this issue has been fixed. Binary data comparison is used in all tests, in order to represent a perfect match. The script finally traverses through every session within every 168 users available in the Greenberg dataset and records their entire data, while maintaining complete data integrity. The resulting data in its new format is highly accessible, easy to use, flexible and customizable.

4.2.3 Deciding the Victims and Masqueraders

After the above-mentioned data extraction, 31 masqueraders and 75 victims are decided by examining the general statistical information acquired. A PHP script is written in order to train the 75 profiles using our Naïve Bayes learning algorithm. The profiling process took approximately 5 hours to complete, yielding a total of 1,348,650 data items.

4.2.4 Calculating the Confidence Ratio

In order to calculate the confidence ratio (CR), a set of command lines are tested against a given profile. For instance, considering the following test input set of five command-lines $\{ls, cd\ classes, vi\ hello.txt, whoami, exit\}$, we test each command line against user x's profile. Assuming that after each command line's independent classification based on the Naïve Bayes classifier, 4 command lines are classified as legitimate and 1 is classified as illegitimate. In this example, our CR for the five command lines belonging to user x is obtained as $\frac{4}{5} \times 100 = 80$. Although initially it may seem that the tested input set belongs to user x given a CR of 80%, this conclusion is based on only five command lines. If the next five command lines produces a CR of 5%, we can immediately sense that our original hypothesis may be faulty.

4.2.5 Determining the Legitimacy of Users

We use the sequential sampling technique in order to complement the nature of continuous authentication systems. The sequential sampling technique allows us to make better decisions as the size of our input set increases based on a given sample size. Different applications require different sample sizes.

The sequential sampling technique has been used to determine the legitimacy of users based on mouse dynamics, where sample sizes ranged from 25 to 100 [1]. The sample size used in a mouse dynamics application differs from that of a command-line based application. For instance, five command lines may differ in significance than five mouse gestures or clicks.

In order to use the sequential sampling technique, we have built different sampling plans that will allow us to test the legitimacy of a given user. To build a sampling plan, we have to determine the *accept*, *continue* and *reject* regions. In order to determine these regions, we have established the parameters involved in producing the two lines that separate the three regions. Depending on the application, the values of the required parameters are different. In order to determine suitable values for the required parameters p_1 and p_2, which represent the lower and higher thresholds respectively, we have conducted a simple test. This test consisted in determining values that will adjust the distance between the accept/reject lines in such a way that it will satisfy the overall range of our CR values. After several trials and error cases, three sampling plans are selected. The values chosen for each sampling plan are recorded in Table 20.

Table 20: Three selected sampling plans

Sampling Plan	$\boldsymbol{p_1}$	$\boldsymbol{p_2}$	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$
A	0.29	0.71	0.01	0.01
B	0.30	0.70	0.01	0.01
C	0.31	0.69	0.01	0.01

In Table 20, α and β represent the acceptable type I and the acceptable type II errors respectively. In all our test cases, the values for these two parameters are set to 0.01. The three selected sampling plans are illustrated in Figure 4, Figure 5 and Figure 6.

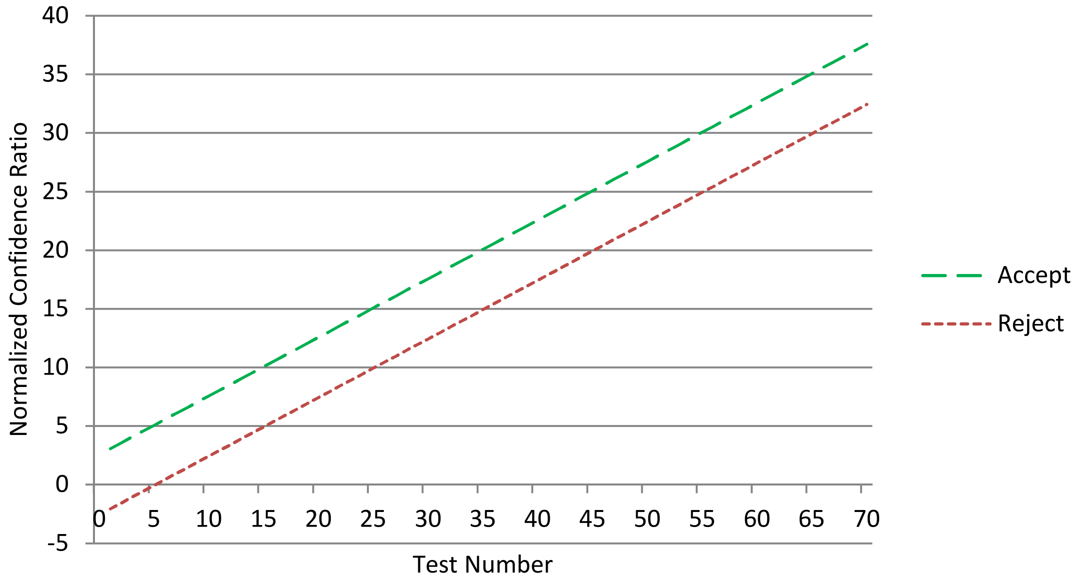

Figure 4: Sampling Plan A ($p_1 = 0.29$, $p_2 = 0.71, \alpha = 0.01$, $\beta = 0.01$).

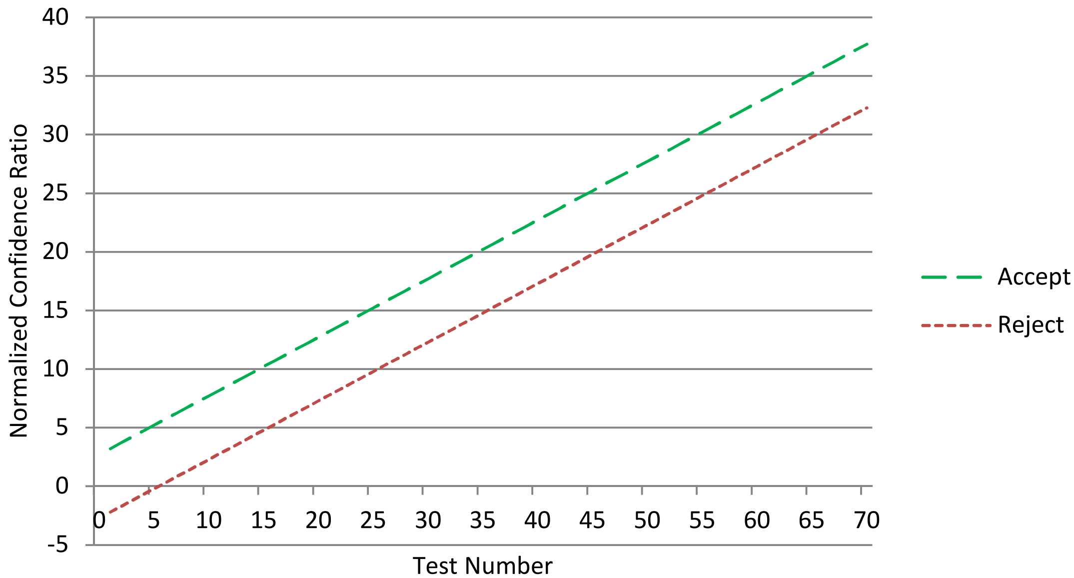

Figure 5: Sampling Plan B ($p_1 = 0.30$ $p_2 = 0.70, \alpha = 0.01$, $\beta = 0.01$).

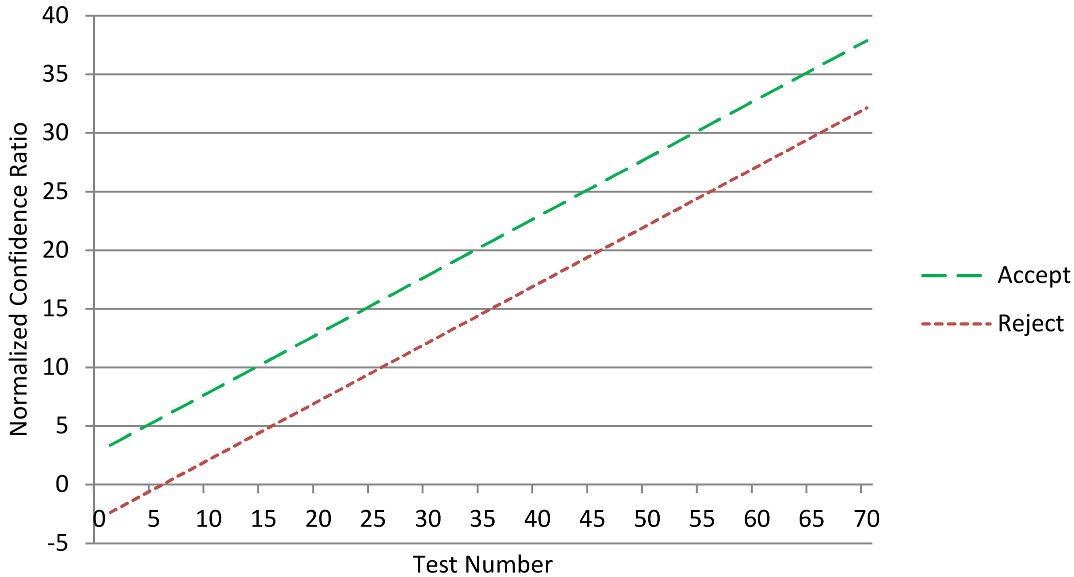

Figure 6: Sampling Plan C ($p_1 = 0.31$ $p_2 = 0.69, \alpha = 0.01$, $\beta = 0.01$)

4.2.6 Calculating the MTTA, FAR, and FRR

The p_1 and p_2 values determine the sensitivity of the system to decision making. As p_1 is decremented, the *continue* region becomes smaller and the system becomes more susceptible to making faster decisions. Hence, Sampling Plan A is expected to have a lower Mean-Time-to-Alarm (MTTA) value than Sampling Plan B and C. As p_1 is incremented, the MTTA value is expected to climb, hence, decisions are made slower. A lower MTTA does not always suggest a better system. The challenge is to lower the MTTA while making accurate decisions.

In order to calculate the False Rejection Rate (FRR), every victim is tested against its own profile. As previously mentioned, the first 1,000 command lines of a user are utilized in order to build its relative profile. It is crucial not to use the same data source as the new input for testing purposes; otherwise the results will not carry great weight in our conclusion. The first command-line used to test the user against its own profile is the 1,001st element in the given user's data pool. There are 75 victims in total, hence, 75 tests are completed for each sampling plan in order to calculate the relative FRR.

Calculating the FAR value involves testing all masqueraders against all victims for each sampling plan. All 31 masqueraders are contributing in attacking each of the 75 victims. The expected results in this test are rejections. In the case where an acceptance has been issued to a masquerader, the result is recorded and the FAR value is updated accordingly.

A maximum TTA of 1000 command-lines is set in order to compensate for system halts. System halts can occur when a decision cannot be made using the available input data against the sampling plan. This can occur if the normalized confidence ratio continues to be in the *continue region* of the sampling plan without penetrating the final decision regions (i.e. accept or reject). System halts can also be triggered as a result of insufficient input data that can cause the prevention of a decision from being made.

4.2.7 Comparing a User against a Given Profile

In order to utilize the sequential sampling technique, for each sampling plan, a sample size must be set before any testing can begin. The sample size determines the number of command lines to sample before attempting to make a decision. For instance, a sample size of 5 means that the iterator will sample and accumulate every 5 command-lines as they are made available. Every iteration is recognized by a number, which is denoted as a *test number*. With a sample size of 5 command-lines, *test number 3* indicates that 15 command lines have been collected and tested.

A PHP script is written in order to simulate the testing process of a user against a given profile. This script incorporates the sequential sampling technique in order to make decisions on the legitimacy of the test user. The three different sampling plans are tested using different sample sizes, while the results are recorded in two different MySQL tables. The *engine_report* MySQL table (Table 21) is used to detail the final results made by the algorithm as a user is tested against a profile.

Table 21: engine_report MySQL table schema

Field Name	Description
id	This field is used as the primary key for this table. It is used to identify the given report (record).
uid_input	This field identifies the user that is being tested against a given profile.
uid_profile	This field indicates the user-profile.
iteration	This field indicates the number of iterations required to make a decision.
decision_expected	This field indicates the expected decision to be made. (i.e. *Accept* if

	uid_input is equal to *uid_profile*)
decisioin_made	is field indicates the final decision ıde after the testing was completed.
seconds	This field indicates the number of seconds required to make a decision.
sample_size	This field indicates the sample size used in the sequential sampling technique.
num_commands	This field indicates the number of command-lines that were required in order to make a decision.
num_trained_items	This field indicates the number of command-lines that were used to train the profile. In all of the tested cases, this number remained at 1,000.
p1	This field indicates the lower threshold used in the sampling plan.
p2	This field indicates the higher threshold used in the sampling plan.
alpha	This field represents the acceptable type I error (false positive)
beta	This field represents the acceptable

	type II error (false negative)
max_iterations	This field indicates the maximum allowed number of iterations.

The details of each decision are recorded in the *engine_output* MySQL table (Table 22). This table outlined the successive progression of the decision making process of the sequential sampling technique (as shown in Table 23 and Table 24).

Table 22: engine_output MySQL table schema

Field Name	Description
report_id	This field identifies the report that this record belongs to.
test_num	This field indicates the test number for the given iteration. (i.e. 1,2,3..)
accept_limit	This field represents the accept value, given the test number,p1, p2, alpha and beta.
reject_limit	This field represents the reject value, given the test number, p1, p2, alpha and beta.
normalized_cr	This field represents the normalized confidence ratio after testing n*N command lines, where n is the test

	number and N is the sample size.
cr	This field represents the confidence ratio after testing n*N command lines, where n is the test number and N is the sample size.
num_commands	This field represents the number of command-lines used to calculate the confidence ratio.

Table 23: Details of the final results made by using CABLUCS

Id	Uid Input	Uid Profile	Iteration	Decision Expected	Decision Made	Seconds	Sample Size	Number of Commands	Number of Trained Commands	P1	P2	Alpha	Beta	Max Iteration
1	86	86	12	Accept	Accept	20	5	55	1000	0.3	0.7	0.01	0.01	1000
2	30	30	67	Accept	Reject	138	5	330	1000	0.3	0.7	0.01	0.01	1000
3	89	55	6	Reject	Reject	9	3	15	1000	0.31	0.69	0.01	0.01	1000
4	24	24	9	Accept	Accept	77	20	160	1000	0.29	0.71	0.01	0.01	1000

Table 24: Successive progression of the decision making process

Report id	Test Number	Accept Limit	Reject Limit	Normalized CR	CR	Number of Commands
3	1	3.3715216902596	-2.3715216902596	0	0	3
3	2	3.8715216902596	-1.8715216902596	0	0	6
3	3	4.3715216902596	-1.3715216902596	0	0	9
3	4	4.8715216902596	-0.8715216902596	0	0	12
3	5	5.3715216902596	-0.3715216902596	0	0	15

3	6	5.8715216902596	0.12847830974036	0	0	18

4.2. Performance Results

The following table describes the annotations used to describe each completed test. Here, TID is used to indicate the given test. The list of prepared sampling plans and tests are shown in Table 25.

Table 25: List of prepared sampling plans and tests

TID	Sample Size	Sampling Plan	p_1	p_2	α	β
1_A	3	A	0.29	0.71	0.01	0.01
1_R	3	A	0.29	0.71	0.01	0.01
2_A	3	B	0.30	0.70	0.01	0.01
2_R	3	B	0.30	0.70	0.01	0.01
3_A	3	C	0.31	0.69	0.01	0.01
3_R	3	B	0.31	0.69	0.01	0.01
4_A	3	-	0.32	0.68	0.01	0.01
5_A	3	-	0.33	0.67	0.01	0.01
6_A	3	-	0.34	0.66	0.01	0.01
7_A	3	-	0.35	0.65	0.01	0.01
8_A	5	A	0.29	0.71	0.01	0.01
4_R	5	A	0.29	0.71	0.01	0.01
9_A	5	B	0.30	0.70	0.01	0.01
9_R	5	B	0.30	0.70	0.01	0.01
10_A	5	C	0.31	0.69	0.01	0.01
10_R	5	C	0.31	0.69	0.01	0.01
11_A	10	A	0.29	0.71	0.01	0.01
11_R	10	A	0.29	0.71	0.01	0.01
12_A	10	B	0.30	0.70	0.01	0.01
12_R	10	B	0.30	0.70	0.01	0.01
13_A	10	C	0.31	0.69	0.01	0.01
13_R	10	C	0.31	0.69	0.01	0.01
14_R	15	A	0.29	0.71	0.01	0.01
15_R	15	B	0.30	0.70	0.01	0.01
16_R	15	C	0.31	0.69	0.01	0.01
17_R	20	A	0.29	0.71	0.01	0.01
18_R	20	B	0.30	0.70	0.01	0.01
19_R	20	C	0.31	0.69	0.01	0.01

Figure 7 and Figure 8 illustrate a visualization of the sequential sampling technique using actual data taken from our experiment. Figure 7 shows a user's test input on its own profile based on TID 9_A. TID values with the subscript letter *A*, represent a test case where the expected decision is an acceptance, and values with the subscript letter *R,* represent a test case where the expected decision is a rejection.

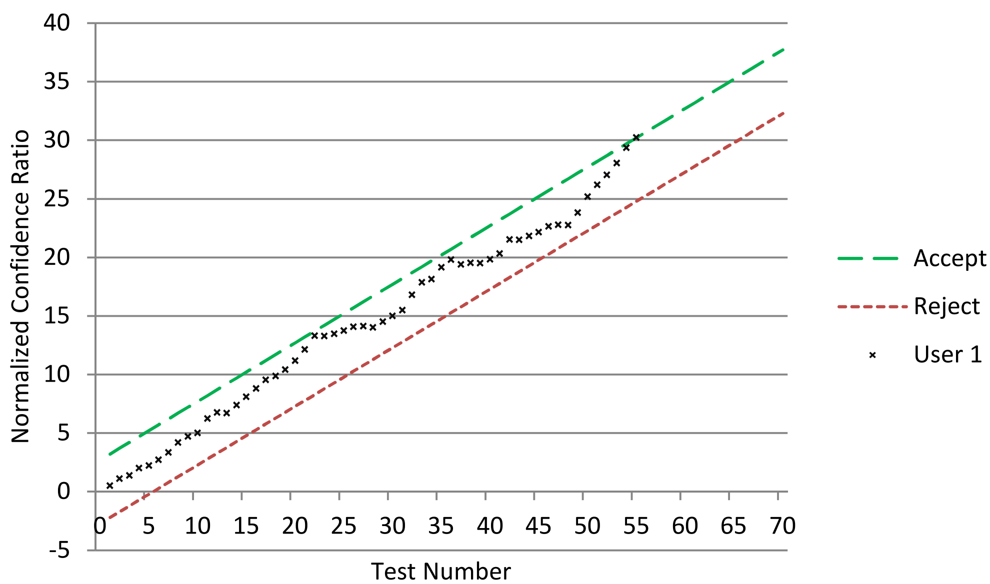

Figure 7: User 1's new input tested on User 1's profile

The user's normalized confidence ratio (CR) navigates through the *continue* region, until a decision has been made. We can witness a decision being made at *test number 55*, where the user's normalized confidence ratio crosses the acceptance line and hence the user is accepted. This is an example of a successful trial. Figure 8 demonstrates a test (TID 9_R) on the same profile, however this time, the profile does not belong to the user. By looking at this figure, we can witness that the system comes close to detecting the masquerader near *test number 24*, however the confidence ratio climbs as new data is made available. Eventually, this user is rejected at test number 70, where the confidence ratio penetrates the reject region.

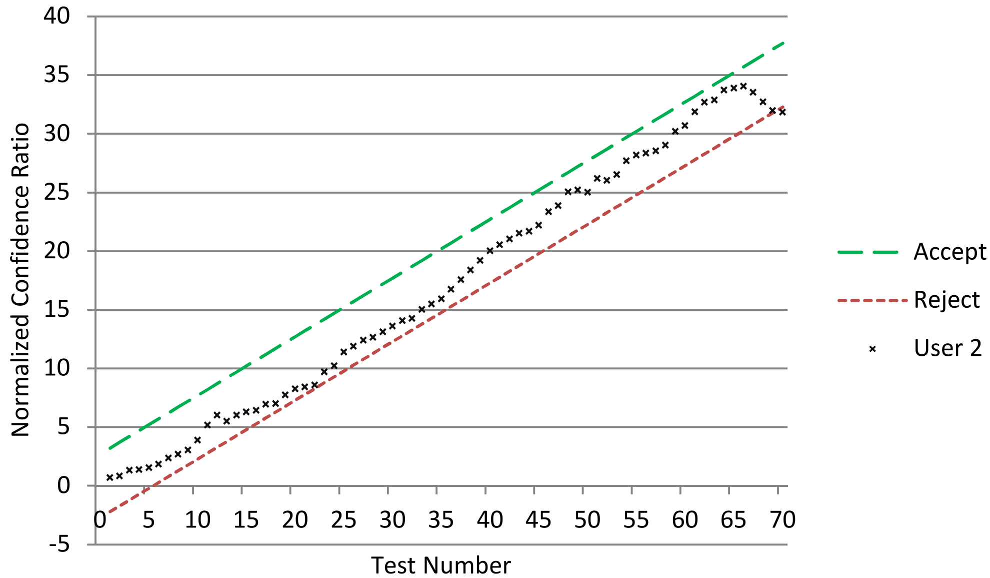

Figure 8: User 2's input tested on User 1's profile

During the course of the system's analysis of the user, we can witness that the decision can go either way. Depending on the sensitivity of the system, decisions can be altered dramatically. For instance, if the *continue* region was reduced in size, a faster decision would have been made at test number 24, which is more than 50% faster than the latter. The challenge lies in determining a sampling plan that will be ideal to the given application. It is important to mention that a sampling plan must be built, such that it compensates for the entire training range.

The following table (Table 26) illustrates the decision making process of the system for the previous two examples.

Table 26: Decision making process of the sequential sampling technique

Test Number	Number of Commands	Accept Limit	Reject Limit	User 1's Normalized CR	User 2's Normalized CR	Decision
1	5	3.211632	-2.21163	0.5	0.666667	Continue
2	10	3.711632	-1.71163	1.111111	0.8	Continue
3	15	4.211632	-1.21163	1.384615	1.285714	Continue
4	20	4.711632	-0.71163	2	1.333333	Continue
5	25	5.211632	-0.21163	2.222222	1.5	Continue
6	30	5.711632	0.288368	2.7	1.8	Continue
7	35	6.211632	0.788368	3.333333	2.333333	Continue
8	40	6.711632	1.288368	4.173913	2.666667	Continue
9	45	7.211632	1.788368	4.695652	3	Continue
10	50	7.711632	2.288368	5	3.846154	Continue
11	55	8.211632	2.788368	6.233333	5.133333	Continue
12	60	8.711632	3.288368	6.75	6	Continue
13	65	9.211632	3.788368	6.685714	5.473684	Continue
14	70	9.711632	4.288368	7.388889	6	Continue
15	75	10.21163	4.788368	8.076923	6.25	Continue
16	80	10.71163	5.288368	8.8	6.4	Continue
17	85	11.21163	5.788368	9.536585	6.925926	Continue
18	90	11.71163	6.288368	9.857143	6.967742	Continue
19	95	12.21163	6.788368	10.40476	7.71875	Continue
20	100	12.71163	7.288368	11.16279	8.235294	Continue
21	105	13.21163	7.788368	12.13333	8.4	Continue
22	110	13.71163	8.288368	13.29167	8.555556	Continue
23	115	14.21163	8.788368	13.26923	9.684211	Continue
24	120	14.71163	9.288368	13.47368	10.2	Continue
25	125	15.21163	9.788368	13.75	11.36364	Continue
26	130	15.71163	10.28837	14.06557	11.86957	Continue
27	135	16.21163	10.78837	14.12308	12.375	Continue
28	140	16.71163	11.28837	14	12.62745	Continue
29	145	17.21163	11.78837	14.5	13.07843	Continue
30	150	17.71163	12.28837	15	13.58491	Continue
31	155	18.21163	12.78837	15.5	14.03774	Continue
32	160	18.71163	13.28837	16.8	14.22222	Continue
33	165	19.21163	13.78837	17.85882	15	Continue
34	170	19.71163	14.28837	18.13333	15.45455	Continue
35	175	20.21163	14.78837	19.15789	15.90909	Continue
36	180	20.71163	15.28837	19.8	16.71429	Continue
37	185	21.21163	15.78837	19.38095	17.52632	Continue
38	190	21.71163	16.28837	19.53271	18.34483	Continue

39	195	22.21163	16.78837	19.5	19.16949	Continue
40	200	22.71163	17.28837	19.82301	20	Continue
41	205	23.21163	17.78837	20.32479	20.5	Continue
42	210	23.71163	18.28837	21.52066	21	Continue
43	215	24.21163	18.78837	21.5	21.5	Continue
44	220	24.71163	19.28837	21.824	21.66154	Continue
45	225	25.21163	19.78837	22.14286	22.16418	Continue
46	230	25.71163	20.28837	22.64063	23.33333	Continue
47	235	26.21163	20.78837	22.78788	23.84058	Continue
48	240	26.71163	21.28837	22.75556	25.01408	Continue
49	245	27.21163	21.78837	23.8	25.18056	Continue
50	250	27.71163	22.28837	25.17241	25	Continue
51	255	28.21163	22.78837	26.18	26.17105	Continue
52	260	28.71163	23.28837	27.02632	26	Continue
53	265	29.21163	23.78837	28.03871	26.5	Continue
54	270	29.71163	24.28837	29.3625	27.65854	Continue
55	275	30.21163	24.78837	30.21605	28.15476	Accept User 1
56	280	30.71163	25.28837		28.32941	Continue
57	285	31.21163	25.78837		28.5	Continue
58	290	31.71163	26.28837		29	Continue
59	295	32.21163	26.78837		30.17045	Continue
60	300	32.71163	27.28837		30.66667	Continue
61	305	33.21163	27.78837		31.82609	Continue
62	310	33.71163	28.28837		32.66667	Continue
63	315	34.21163	28.78837		32.84043	Continue
64	320	34.71163	29.28837		33.68421	Continue
65	325	35.21163	29.78837		33.85417	Continue
66	330	35.71163	30.28837		34.02062	Continue
67	335	36.21163	30.78837		33.5	Continue
68	340	36.71163	31.28837		32.69231	Continue
69	345	37.21163	31.78837		31.94444	Continue
70	350	37.71163	32.28837		31.81818	Reject User 2

The two types of tests that were conducted in this experiment include the testing for rejection and the testing for acceptance. Acceptance tests involved the testing of a user against its own profile. The purpose of this test is to calculate the FRR of our intrusion detection system. Given that there are 75 victims, this type of testing did not require vast amount of computational time. Depending on the sample size, the acceptance tests did not require more than an hour to complete a single trial on our workstation. However, in the case of a rejection test, 31 masqueraders are used as input to 75 profiles. Depending on the sample size, this type of test can take up to 10 hours to complete a single trial on our workstation. The purpose of a rejection test is to calculate the FAR of our intrusion detection system.

The following table (Table 27) illustrates the final results that we have achieved by implementing a Naïve Bayes learning mechanism in conjunction with the decision making of the sequential sampling technique.

Table 27: Results achieved based on different parameters and sampling sizes

Test ID	Sample Size N (Actions)	p_1	p_2	α	β	FAR (%)	FRR (%)	DR (%)	Min TTA (Commands)	Max TTA (Actions)	Mean TTA (Actions)	CPU Usage (Seconds)
	3	0.27	0.73	0.01	0.01	03.225		96.78	12	183	16.28	
	3	0.28	0.72	0.01	0.01	03.183		96.82	12	186	16.61	
1	3	0.29	0.71	0.01	0.01	02.968	12.00	97.03	15	198	19.35	6.97
2	3	0.30	0.70	0.01	0.01	02.882	12.00	97.12	15	237	20.09	7.20
3	3	0.31	0.69	0.01	0.01	02.882	12.00	97.12	15	252	20.60	7.35
4	3	0.32	0.68	0.01	0.01	02.796	12.00	97.20	18	252	23.63	-
5	3	0.33	0.67	0.01	0.01	02.581	12.00	97.42	18	300	24.46	-
6	3	0.34	0.66	0.01	0.01	02.581	12.00	97.42	18	546	25.25	-
7	3	0.35	0.65	0.01	0.01	02.581	12.00	97.42	21	552	29.13	-
8	5	0.29	0.71	0.01	0.01	02.237	12.00	97.76	25	555	34.26	10.42
9	5	0.30	0.70	0.01	0.01	02.237	12.00	97.76	25	575	35.17	10.47
10	5	0.31	0.69	0.01	0.01	02.237	12.00	97.76	25	575	36.16	10.59
11	10	0.29	0.71	0.01	0.01	01.464	13.33	98.41	50	910	67.94	17.61
12	10	0.30	0.70	0.01	0.01	01.421	13.33	98.45	50	910	70.22	18.10
13	10	0.31	0.69	0.01	0.01	01.421	13.33	98.41	50	920	72.59	18.75
14	15	0.29	0.71	0.01	0.01	-	12.00		-	-	-	-
15	15	0.30	0.70	0.01	0.01	-	12.00		-	-	-	-
16	15	0.31	0.69	0.01	0.01	-	12.00		-	-	-	-
17	20	0.30	0.70	0.01	0.01	-	13.33		-	-	-	-
18	20	0.31	0.69	0.01	0.01	-	13.33		-	-	-	-
19	20	0.29	0.71	0.01	0.01	-	13.33		-	-	-	-

Looking at the results, a few patterns are visible. We can witness that as the sample size increases the FAR rate is decreased. A larger sample size allows the system to gain a better understanding of the given user, before making a decision. Figure 9 illustrates the three different sampling plans used in testing the system. We can clearly notice that as the sample size is increased, the FAR rate decreases.

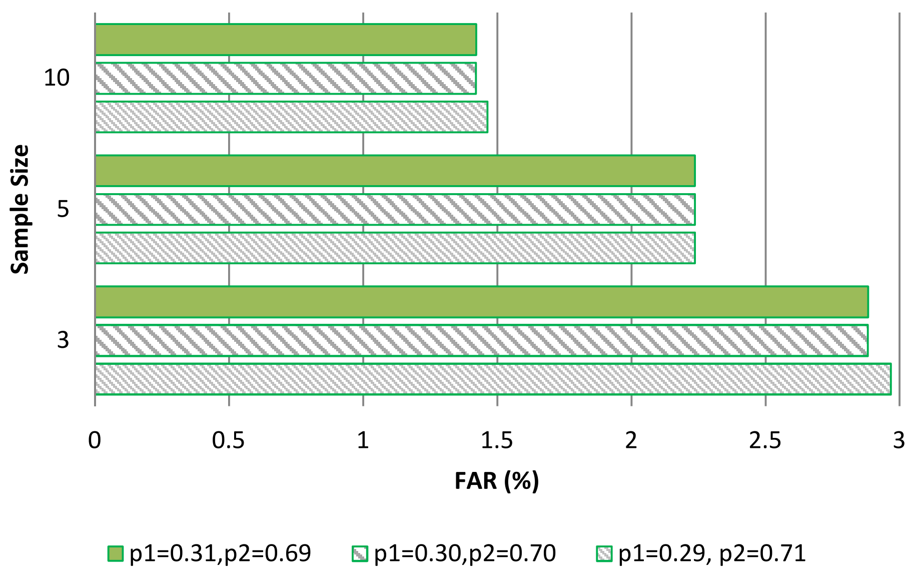

Figure 9: Analysing the system's FAR rate versus the sample size

Figure 10 illustrates the analysis of the FRR rate as the sample size increases. We can witness that as the sample size increases the FRR rate also increased.

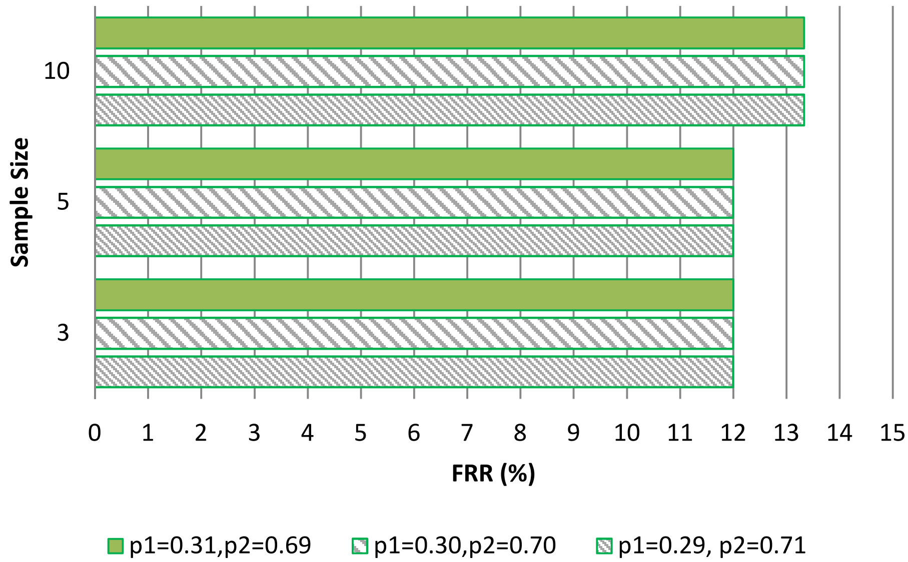

Figure 10: Analysing the system's FRR rate versus the sample size

Figure 11 illustrates the pattern of the MTTA value as the sample size is increased. It is observed that the Time-to-Alarm (TTA) is increased as the sample size is increased.

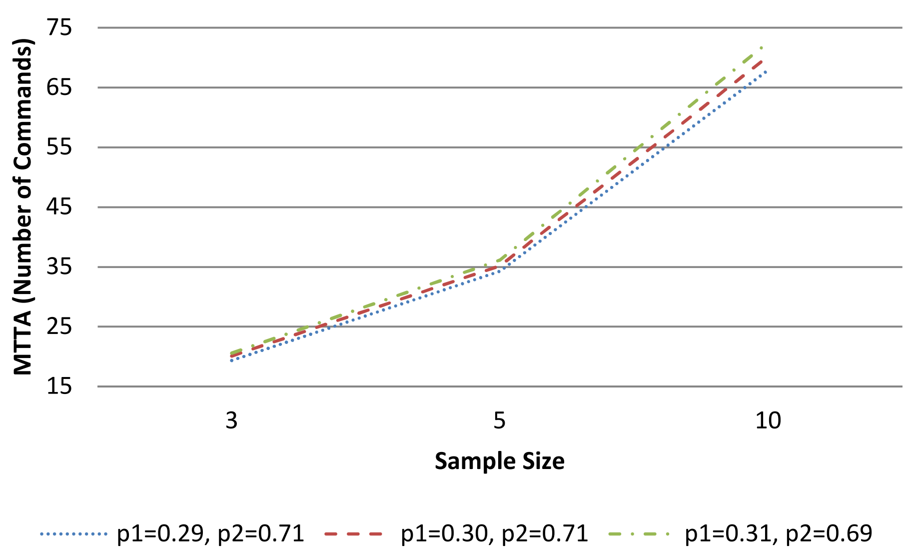

Figure 11: Analysis of the system's MTTA value as the sample size is increased

Figure 12 shows a closer view at the effects of the sample size on the MTTA value. We can observed that depending on the sampling plan, the MTTA is also affected. This is understandable due to the fact that as the lower threshold is increased within the sampling plan, the *continue* region is also increased in size. Therefore, the time spent in the *continue* region is increased.

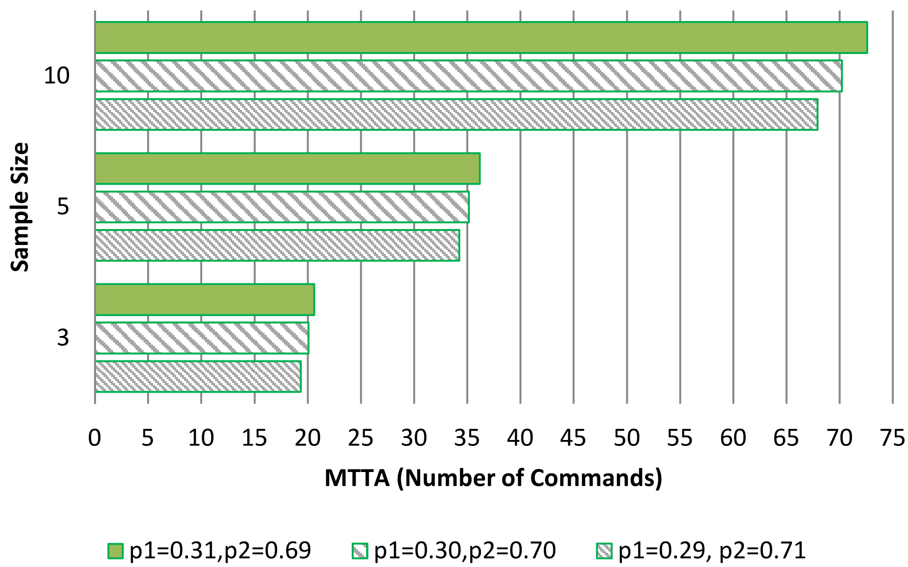

Figure 12: Analysing the system's MTTA value versus the sample size

Figure 13 illustrates the pattern between the CPU usages and the different sample sizes. Naturally, the CPU usage follows the same pattern as the MTTA value. Depending on the workstation used, the values in seconds are different. However, the illustrated pattern should remain the same.

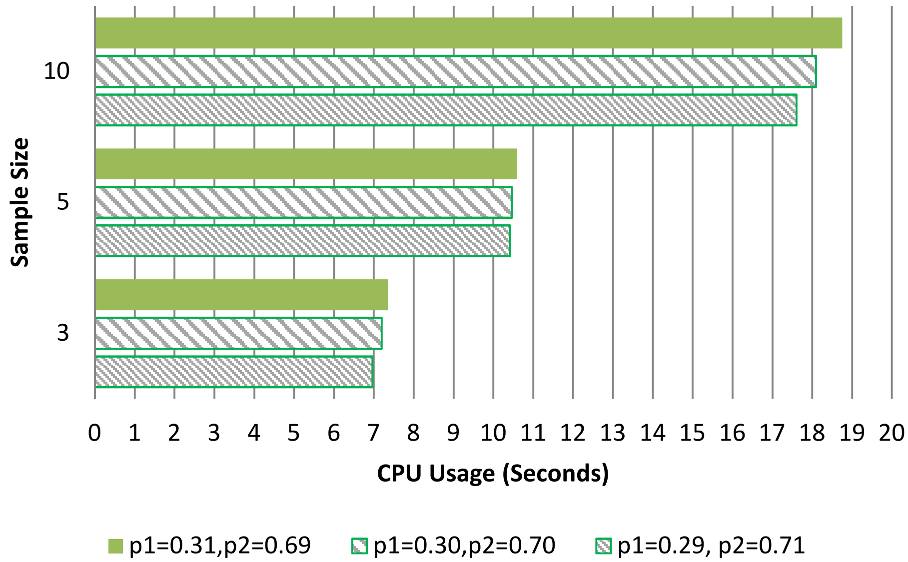

Figure 13: Analyzing the systems CPU Usage versus the sample size

A brief discussion on the given results will be given in the next section.

4.3. Discussion

As stated earlier, it is the goal of any intrusion detection system to reduce the FAR, FRR and the MTTA values. The challenge lies in finding suitable approaches that can accomplish this task in an efficient way. Looking at our results, we can witness a common trend, the more accurate our results are, the more time is consumed. Figure 14, Figure 15 and Figure 16 show the relationship between the FRR and the FAR value in three different sampling plans, using three different sample sizes.

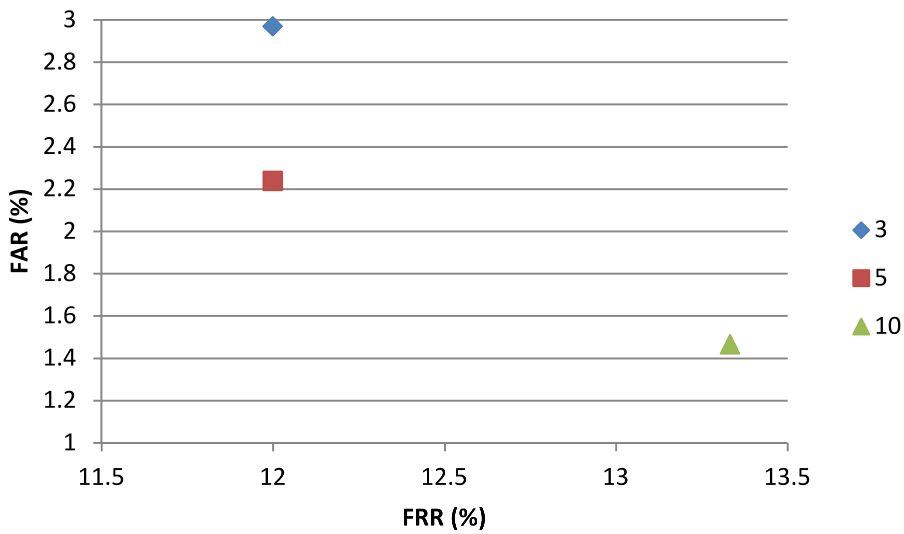

Figure 14: Relationship between sample size and FRR/FAR using sampling plan A

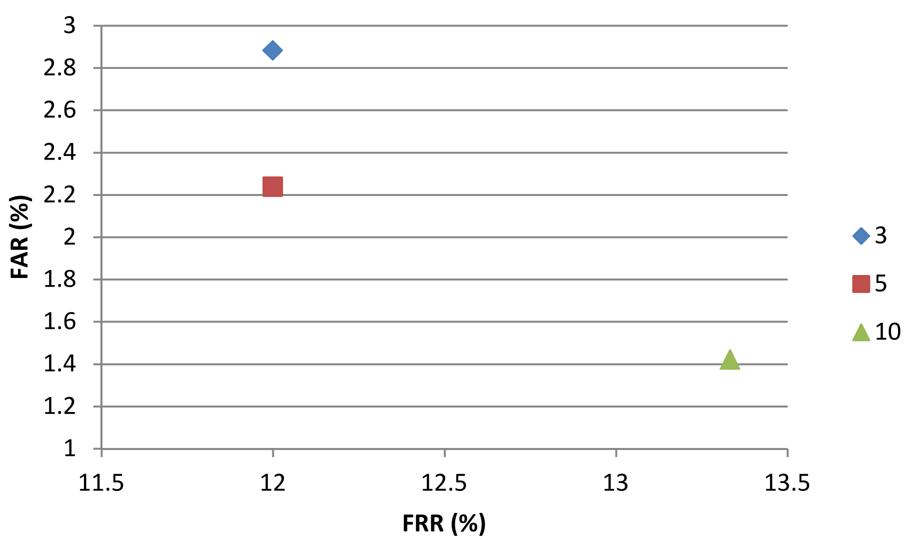

Figure 15: Relationship between sample size and FRR/FAR using sampling plan B

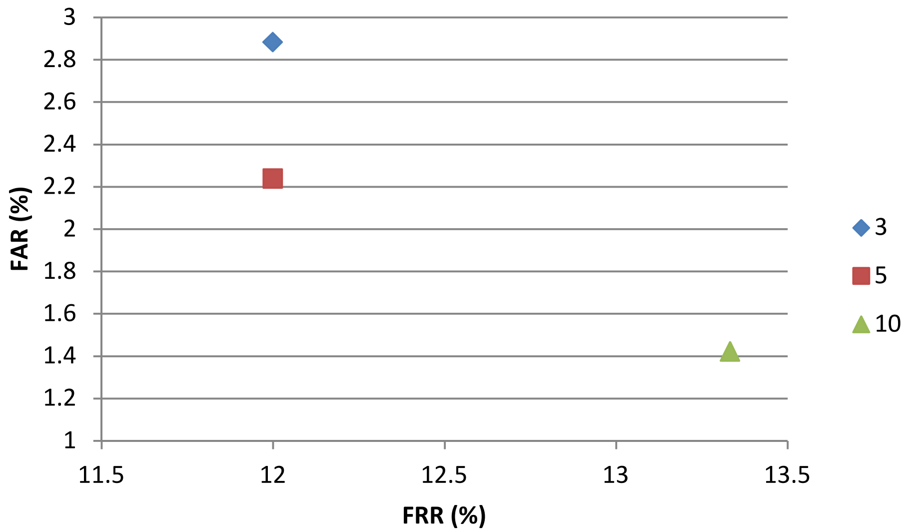

Figure 16: Relationship between sample size and FRR/FAR using sampling plan C

It is important to acknowledge that the data used to calculate the FRR rate as opposed to the FAR rate was minimal. In order to calculate the FRR rate, 75 cases were tested, whereas the FAR rate was calculated by testing 3,235 cases. The FRR rate may become quite different if a more extensive testing procedure is applied.

The results show a promising range of detection rates. Depending on the sampling plan and sample size, the detection rate ranged from 96.78% to 98.41%. These values represent a promising system compared to other proposed intrusion detection systems [15]. Although there are not many literatures in the intrusion detection field where the Greenberg dataset has been used used, comparing these results solely on the basis of the outcome achieved, we can confirm that the results produced are quite competitive.

Depending on the ranking function used to evaluate the integrity of the system, the ranking results can vary. Hence, the emphasis can be set either on type I errors or type II errors. The following ranking function [10] represents the foundation for determining such emphasis.

$$Cost = \alpha(FAR) + \beta(FRR) \quad (19)$$

If there is no preference to the type of error considered, the α and β attributes can be ignored, i.e. set to 1. Therefore, in order to calculate the cost of a given detection algorithm, we add the FAR and FRR rates. Table 28 shows comparison comparative study of numerous detection methods based on a ranking function that does not emphasize on a particular type of error.

Table 28: Result comparison based on Cost = (FAR) + (FRR)

Method	Cost	DR (%)	FAR (%)	FRR (%)	Sample Size	Trained	Data
N. Bayes Classifier	33.8	70.9	29.1	4.7	10	1000	Greenberg
N. Bayes Classifier	23.6	82.1	17.9	5.7	10	1000	Greenberg
CABLUCS	14.97	97.3	2.97	12	3	1000	Greenberg
CABLUCS	14.24	97.76	2.24	12	5	1000	Greenberg
CABLUCS	14.75	98.45	1.42	13.33	10	1000	Greenberg
Customized Grammars	14.1	93.1	6.9	7.2	-	-	SEA
Customized Grammars	30.6	71.0	29.0	1.6	-	-	SEA
Self Signature with Uniqueness	14.6	91.3	8.7	5.9	-	-	SEA
Self Signature with Uniqueness	33.9	67.5	32.5	1.4	-	-	SEA
Boosting Decision Stumps	20.9	89.2	10.8	10.1	-	-	SEA
SVM	29.6	80.1	19.9	9.7	-	-	SEA
ECM	30.2	72.3	27.7	2.5	-	-	SEA
N. Bayes (no updating)	38.4	66.2	33.8	4.6	-	-	SEA
N. Bayes (updating)	39.8	61.5	38.5	1.3	-	-	SEA
Uniqueness	62	39.4	60.6	1.4	-	-	SEA
IPAM	61.3	41.4	58.6	2.7	-	-	SEA
Hybrid Markov	53.9	49.3	50.7	3.2	-	-	SEA

Sequence match	66.9	36.8	63.2	3.7	-	-	SEA
Compression	70.8	34.2	65.8	5.0	-	-	SEA
Bayes one-step Markov	37.4	69.3	30.7	6.7	-	-	SEA

Using this ranking function, we can witness that our system ranks 1st in the detection methods that use the Greenberg dataset, and ranks 2nd overall, regardless of the dataset used.

Table 29: Result ranking comparison based on Cost = (FAR) + (FRR)

Rank	**Method**	**Cost**	**Dataset**
1	Customized Grammars	14.1	SEA
2	CABLUCS	14.24	Greenberg
3	Self Signature with Uniqueness	14.6	SEA
4	CABLUCS	14.75	Greenberg
5	CABLUCS **S**	14.97	Greenberg
6	Boosting Decision Stumps	20.9	SEA
7	Naïve Bayes Classifier	23.6	Greenberg
8	SVM	29.6	SEA
9	ECM	30.2	SEA
10	Customized Grammars	30.6	SEA
11	Naïve Bayes Classifier	33.8	Greenberg
12	Self Signature with Uniqueness	33.9	SEA
13	Bayes one-step Markov	37.4	SEA
14	Naïve Bayes (no updating)	38.4	SEA
15	Naïve Bayes (updating)	39.8	SEA
16	Hybrid Markov	53.9	SEA
17	IPAM	61.3	SEA
18	Uniqueness	62	SEA
19	Sequence match	66.9	SEA
20	Compression	70.8	SEA

The SEA dataset is the work of Schonlau et al. [9], which is a more commonly used benchmark dataset. In their work, the emphasis was set based on achieving a 1% FRR rate [10]. After the completion of their tests, the only successful method to achieve the given FRR rate was *Uniqueness.* In order to rank *Uniqueness* as the best detection method based on the given criteria, the β (type II error) emphasis was set to 6. Table 30 shows the results in comparison to the Schonlau et al. ranking function.

Table 30: Result comparison based on Cost = (FAR) + 6(FRR)

Method	**Cost**	**DR (%)**	**FAR (%)**	**FRR (%)**	**Sample Size**	**Trained**	**Dataset**
N. Bayes Classifier	57.3	70.9	29.1	4.7	10	1000	Greenberg
Naïve Bayes Classifier	52.1	82.1	17.9	5.7	10	1000	Greenberg
CABLUCS	74.97	97.3	2.97	12	3	1000	Greenberg
CABLUCS	74.24	97.76	2.24	12	5	1000	Greenberg
CABLUCS	81.4	98.45	1.42	13.33	10	1000	Greenberg
Customized Grammars	14.1	93.1	6.9	7.2	-	-	SEA
Customized Grammars	30.6	71.0	29.0	1.6	-	-	SEA
Self Signature with Uniqueness	14.6	91.3	8.7	5.9	-	-	SEA
Self Signature with Uniqueness	33.9	67.5	32.5	1.4	-	-	SEA
Boosting Decision Stumps	20.9	89.2	10.8	10.1	-	-	SEA
SVM	29.6	80.1	19.9	9.7	-	-	SEA
ECM	30.2	72.3	27.7	2.5	-	-	SEA
N. Bayes (no updating)	38.4	66.2	33.8	4.6	-	-	SEA
N. Bayes (updating)	39.8	61.5	38.5	1.3	-	-	SEA
Uniqueness	62.0	39.4	60.6	1.4	-	-	SEA
IPAM	61.3	41.4	58.6	2.7	-	-	SEA
Hybrid Markov	53.9	49.3	50.7	3.2	-	-	SEA
Sequence match	66.9	36.8	63.2	3.7	-	-	SEA
Compression	70.8	34.2	65.8	5.0	-	-	SEA
Bayes one-step Markov	37.4	69.3	30.7	6.7	-	-	SEA

Table 31 shows the ranking comparison of the detection methods based on the new criteria. We can witness that the results have dramatically changed given that the emphasis is now based on the type II error. Considering the relatively high FRR rate of our system in comparison to other mentioned detection methods, it comes as no surprise that our system is now ranked the lowest.

Table 31: Result ranking comparison based on Cost = (FAR) + 6(FRR)

Rank	Method	Cost	Dataset
1	Customized Grammars	14.1	SEA
2	Self Signature with Uniqueness	14.6	SEA
3	Boosting Decision Stumps	20.9	SEA
4	SVM	29.6	SEA
5	ECM	30.2	SEA
6	Customized Grammars	30.6	SEA
7	Self Signature with Uniqueness	33.9	SEA
8	Bayes one-step Markov	37.4	SEA
9	N. Bayes (no updating)	38.4	SEA
10	N. Bayes (updating)	39.8	SEA
11	Naïve Bayes Classifier	52.1	Greenberg
12	Hybrid Markov	53.9	SEA
13	N. Bayes Classifier	57.3	Greenberg
14	IPAM	61.3	SEA
15	Uniqueness	62	SEA
16	Sequence match	66.9	SEA
17	Compression	70.8	SEA
18	CABLUCS	74.24	Greenberg
19	CABLUCS	74.97	Greenberg
20	CABLUCS	81.4	Greenberg

Depending on the favouritism of the ranking function, each detection method can be ranked and used differently, in contingent with the application in question. It can be said that the FAR value of our system shows a more accurate representation of our detection method than the FRR rate. As previously stated, the FAR value is determined after numerous testing for each given trial (3,235 cases per trial), while the FRR value is tested using only 75 cases per trial. Hence, a single incident of a false rejection can significantly skew the overall results.

Looking at the cases where the false rejections were witnessed, we can gain a better understanding for the reasoning behind our high FRR rate. Table 32 shows the 13 profiles that were consistently rejected throughout 21 different test cases. Our high FRR rate is based on 17.33% of our victims that have an average probability of 72.16% in producing a false rejection. Further investigation of the relative sampling plans and the sequential progression in the decision making process of these 13 profiles can demonstrate the reasoning for such consistently high rejection rates.

Table 32 :Users that were falsely rejected in 21 different test cases

Uid	FRR (%)
10	100
13	4.76
19	100
20	33.33
30	76.19
40	100
43	100
45	100
71	52.38
126	14.29
128	33.33
129	76.19
154	47.62
157	100

Chapter 5: Conclusion

In this thesis, we have proposed a hybrid approach based on learning user command sequence for detecting classical masquerade attacks. Our approach (so-called CABLUCS) consisted of two methods, the Naïve Bayes classifier and the sequential sampling technique, used to enhance the capability of a continuous authentication mechanism within an intrusion detection system. In addition, a newly structured dataset was formed using the Greenberg raw dataset, in such a way as to maximize its usability and efficiency. Using this newly structured dataset, a general statistical analysis of the given data was produced, which can be quite helpful to future researchers using the Greenberg dataset. Through experimental evaluation, we found that our scheme achieves a significant improvement over the Maxion and Townsend scheme in terms of accuracy detection.

We believe that this performance is largely attributed to the contribution of the part of our approach that deal with sequential sampling technique for continuous authentication, which constitutes the core of the decision making regarding the legitimacy of a user.

Departing from the results achieved in this thesis, we can infer that our technique can provide significant advancement to the field of masquerade detection, by opening the possibility of exploring the method to other areas of anomaly detection. This can be classified as future work.

Appendix A: Generated Statistics for the Greenberg Dataset (Ordered by Commands)

User	Uid	Commands	History	Errors	Aliases	Lines
scientist-36	20	12056	488	566	3161	73434
scientist-52	15	7705	231	299	717	47280
scientist-42	10	6068	6	644	3243	37598
experienced-7	55	5857	67	612	2926	35896
scientist-18	48	5584	6	240	1816	34258
non-4	106	5050	18	161	3296	30830
scientist-40	38	4605	0	98	628	29020
experienced-20	53	4556	435	370	1646	28054
scientist-4	23	4507	178	320	789	27992
experienced-35	78	4272	28	169	2504	26290
scientist-37	32	4187	121	83	1866	25604
novice-46	146	4163	112	372	1909	26080
scientist-9	26	4067	224	65	665	25424
non-20	99	4042	165	124	2122	24798
experienced-5	62	4015	35	222	910	25220
experienced-28	74	3893	78	60	2516	25116
scientist-27	37	3817	102	85	0	23344
experienced-4	75	3776	2	123	1329	23258
scientist-38	24	3775	48	168	1312	23172
experienced-1	80	3714	174	298	1906	22830
scientist-13	42	3593	357	118	204	21988
scientist-25	30	3508	7	122	379	22706
scientist-14	47	3433	178	183	2	21404
novice-19	164	3401	7	363	0	20816
scientist-23	16	3360	52	135	481	21454
experienced-24	86	3331	222	228	1456	20440
novice-36	142	3213	0	137	0	19840
novice-14	126	3194	0	208	0	19786
novice-33	122	3127	0	106	0	19556
scientist-43	39	3106	0	101	546	19066
scientist-2	25	2954	236	149	1656	18514
experienced-8	68	2930	67	265	625	18114
scientist-19	43	2831	106	112	1330	17560
experienced-22	73	2814	325	122	560	17478
scientist-20	45	2697	74	189	804	17080
scientist-29	17	2683	20	243	530	16632
scientist-34	52	2639	15	88	910	16648
scientist-46	3	2551	80	110	495	16480
scientist-12	28	2499	53	52	1162	15412
novice-1	151	2457	37	213	1381	14960
novice-12	118	2436	0	210	0	16366
experienced-21	59	2394	157	83	974	14762
experienced-9	71	2351	86	136	502	14500
experienced-17	79	2343	0	144	102	14396
novice-3	150	2337	0	93	0	15400
novice-41	115	2317	0	51	1000	14244
experienced-23	81	2306	189	119	1004	14214
novice-28	136	2221	0	120	0	13816
experienced-29	84	2214	59	133	1072	13566
novice-23	129	2138	0	72	0	13186
scientist-30	40	2129	186	123	409	13492
novice-31	157	2073	0	102	18	12692
novice-25	155	2066	2	217	0	13070
scientist-41	49	2037	0	36	0	13036
experienced-30	66	2028	82	110	624	12686
scientist-10	27	2024	77	120	730	12658
novice-37	154	1949	0	57	0	12044

novice-4	131	1919	0	123	0	11758
novice-22	166	1893	1	51	547	11844
experienced-34	76	1869	206	218	598	11676
scientist-1	13	1856	54	111	761	11792
non-11	98	1848	0	61	0	12210
novice-8	153	1822	0	19	0	11298
experienced-14	88	1810	23	153	996	11270
experienced-19	58	1807	163	88	829	11328
experienced-12	77	1763	106	92	889	10840
scientist-21	19	1762	50	134	586	10894
scientist-39	41	1753	173	77	530	10992
experienced-27	67	1693	77	54	741	11032
novice-55	128	1662	6	40	0	10218
non-1	90	1622	0	59	410	10110
experienced-36	63	1580	56	116	781	9718
non-22	112	1567	48	56	0	10004
scientist-5	21	1563	18	78	558	10164
scientist-44	44	1543	12	84	394	9544
scientist-50	12	1496	219	225	387	9526
scientist-24	29	1494	0	55	1217	9250
experienced-25	83	1465	69	89	346	9072
novice-10	114	1464	0	40	872	9038
experienced-11	70	1456	21	86	927	9126
scientist-49	51	1448	138	97	179	9106
novice-35	135	1444	0	54	50	9022
scientist-15	50	1429	200	81	175	9216
non-18	111	1403	0	64	0	8804
novice-47	148	1316	0	78	0	8118
non-23	89	1294	0	48	636	8118
experienced-33	82	1292	83	65	649	7986
novice-44	158	1277	0	40	0	7896
novice-34	147	1276	4	46	0	8146
novice-2	160	1267	0	58	0	8072
non-3	108	1265	9	15	209	7928
non-7	101	1231	3	54	792	7704
novice-29	120	1230	0	44	0	7754
scientist-47	11	1229	9	81	618	7672
novice-27	139	1195	1	63	414	7452
novice-17	132	1194	0	59	0	7702
novice-15	141	1139	0	48	0	7148
novice-26	137	1120	0	60	0	7066
experienced-13	85	1109	25	160	446	6848
novice-39	163	1107	0	51	9	6936
scientist-6	14	1103	33	49	278	7196
novice-18	167	1088	0	38	0	6710
novice-42	119	1068	0	33	5	6774
scientist-35	22	1049	23	29	594	6612
novice-7	145	1039	98	51	36	6608
novice-53	124	1028	0	41	51	6558
novice-50	117	985	0	92	0	6100
scientist-26	6	983	0	70	231	6388
scientist-3	36	978	1	69	255	6398
experienced-32	54	974	47	87	303	6102
novice-40	165	967	0	24	722	6032
novice-30	149	946	0	28	0	5986
experienced-3	87	915	88	42	356	5600
scientist-51	34	910	0	67	358	5754
novice-6	123	871	0	44	0	5520
scientist-45	35	862	17	59	223	5330
novice-9	134	853	0	63	0	5292

novice-21	127	849	1	42	0	5268
novice-24	130	849	48	53	118	5436
non-17	110	848	0	65	0	5330
scientist-8	2	842	0	51	79	5294
novice-38	159	839	0	17	0	5468
non-16	105	821	144	26	0	5108
scientist-48	9	819	0	43	0	5216
experienced-16	60	795	24	22	245	4932
scientist-28	18	765	64	26	235	5032
experienced-6	57	757	0	32	69	4752
scientist-22	7	750	0	39	20	5026
novice-49	156	723	0	31	0	4428
novice-54	138	683	0	56	0	4248
experienced-31	61	683	19	38	454	4368
experienced-26	72	679	0	66	59	4192
novice-13	168	652	0	49	0	4106
novice-45	116	651	0	16	0	4120
novice-52	140	650	0	38	0	4174
novice-43	125	608	0	26	45	3778
scientist-32	5	601	0	20	0	3916
novice-5	121	593	1	67	0	3804
experienced-18	69	575	5	21	114	3548
non-15	94	571	0	28	0	3736
scientist-17	46	569	0	38	0	3792
non-24	96	542	0	34	0	3390
non-10	93	495	0	20	0	3096
non-13	100	487	0	5	0	3072
novice-51	143	480	0	20	0	3046
non-2	113	454	0	15	63	2934
experienced-10	56	446	2	26	170	2774
novice-20	144	418	5	19	0	2722
novice-32	133	385	0	37	60	2512
scientist-7	33	366	0	28	169	2246
non-9	102	357	4	23	45	2432
non-25	104	327	3	18	48	2264
scientist-16	8	326	0	29	38	2250
scientist-33	4	325	0	12	0	2044
novice-48	152	269	0	9	0	1704
novice-11	162	256	2	21	0	1770
novice-16	161	256	0	25	0	1598
scientist-31	1	250	9	20	12	1758
non-5	103	244	0	11	0	1770
non-8	97	239	28	13	18	1524
experienced-15	65	225	0	12	85	1404
experienced-2	64	219	6	11	33	1414
non-12	107	216	0	26	0	1390
scientist-11	31	205	0	13	0	1380
non-14	109	201	1	4	0	1272
non-6	95	177	0	7	0	1152
non-19	92	175	0	7	116	1356
non-21	91	132	0	7	0	890

Appendix B: Generated Statistics for the Greenberg Dataset (Ordered by History)

User	Uid	Commands	History	Errors	Aliases	Lines
scientist-36	20	12056	488	566	3161	73434
experienced-20	53	4556	435	370	1646	28054
scientist-13	42	3593	357	118	204	21988
experienced-22	73	2814	325	122	560	17478
scientist-2	25	2954	236	149	1656	18514
scientist-52	15	7705	231	299	717	47280
scientist-9	26	4067	224	65	665	25424
experienced-24	86	3331	222	228	1456	20440
scientist-50	12	1496	219	225	387	9526
experienced-34	76	1869	206	218	598	11676
scientist-15	50	1429	200	81	175	9216
experienced-23	81	2306	189	119	1004	14214
scientist-30	40	2129	186	123	409	13492
scientist-4	23	4507	178	320	789	27992
scientist-14	47	3433	178	183	2	21404
experienced-1	80	3714	174	298	1906	22830
scientist-39	41	1753	173	77	530	10992
non-20	99	4042	165	124	2122	24798
experienced-19	58	1807	163	88	829	11328
experienced-21	59	2394	157	83	974	14762
non-16	105	821	144	26	0	5108
scientist-49	51	1448	138	97	179	9106
scientist-37	32	4187	121	83	1866	25604
novice-46	146	4163	112	372	1909	26080
scientist-19	43	2831	106	112	1330	17560
experienced-12	77	1763	106	92	889	10840
scientist-27	37	3817	102	85	0	23344
novice-7	145	1039	98	51	36	6608
experienced-3	87	915	88	42	356	5600
experienced-9	71	2351	86	136	502	14500
experienced-33	82	1292	83	65	649	7986
experienced-30	66	2028	82	110	624	12686
scientist-46	3	2551	80	110	495	16480
experienced-28	74	3893	78	60	2516	25116
experienced-27	67	1693	77	54	741	11032
scientist-10	27	2024	77	120	730	12658
scientist-20	45	2697	74	189	804	17080
experienced-25	83	1465	69	89	346	9072
experienced-7	55	5857	67	612	2926	35896
experienced-8	68	2930	67	265	625	18114
scientist-28	18	765	64	26	235	5032
experienced-29	84	2214	59	133	1072	13566
experienced-36	63	1580	56	116	781	9718
scientist-1	13	1856	54	111	761	11792
scientist-12	28	2499	53	52	1162	15412
scientist-23	16	3360	52	135	481	21454
scientist-21	19	1762	50	134	586	10894
scientist-38	24	3775	48	168	1312	23172
non-22	112	1567	48	56	0	10004
novice-24	130	849	48	53	118	5436
experienced-32	54	974	47	87	303	6102
novice-1	151	2457	37	213	1381	14960
experienced-5	62	4015	35	222	910	25220
scientist-6	14	1103	33	49	278	7196
experienced-35	78	4272	28	169	2504	26290
non-8	97	239	28	13	18	1524
experienced-13	85	1109	25	160	446	6848

experienced-16	*60*	*795*	*24*	*22*	*245*	*4932*
scientist-35	*22*	*1049*	*23*	*29*	*594*	*6612*
experienced-14	*88*	*1810*	*23*	*153*	*996*	*11270*
experienced-11	*70*	*1456*	*21*	*86*	*927*	*9126*
scientist-29	*17*	*2683*	*20*	*243*	*530*	*16632*
experienced-31	*61*	*683*	*19*	*38*	*454*	*4368*
non-4	*106*	*5050*	*18*	*161*	*3296*	*30830*
scientist-5	*21*	*1563*	*18*	*78*	*558*	*10164*
scientist-45	*35*	*862*	*17*	*59*	*223*	*5330*
scientist-34	*52*	*2639*	*15*	*88*	*910*	*16648*
scientist-44	*44*	*1543*	*12*	*84*	*394*	*9544*
non-3	*108*	*1265*	*9*	*15*	*209*	*7928*
scientist-47	*11*	*1229*	*9*	*81*	*618*	*7672*
scientist-31	*1*	*250*	*9*	*20*	*12*	*1758*
scientist-25	*30*	*3508*	*7*	*122*	*379*	*22706*
novice-19	*164*	*3401*	*7*	*363*	*0*	*20816*
experienced-2	*64*	*219*	*6*	*11*	*33*	*1414*
novice-55	*128*	*1662*	*6*	*40*	*0*	*10218*
scientist-18	*48*	*5584*	*6*	*240*	*1816*	*34258*
scientist-42	*10*	*6068*	*6*	*644*	*3243*	*37598*
novice-20	*144*	*418*	*5*	*19*	*0*	*2722*
experienced-18	*69*	*575*	*5*	*21*	*114*	*3548*
non-9	*102*	*357*	*4*	*23*	*45*	*2432*
novice-34	*147*	*1276*	*4*	*46*	*0*	*8146*
non-25	*104*	*327*	*3*	*18*	*48*	*2264*
non-7	*101*	*1231*	*3*	*54*	*792*	*7704*
novice-25	*155*	*2066*	*2*	*217*	*0*	*13070*
novice-11	*162*	*256*	*2*	*21*	*0*	*1770*
experienced-4	*75*	*3776*	*2*	*123*	*1329*	*23258*
experienced-10	*56*	*446*	*2*	*26*	*170*	*2774*
non-14	*109*	*201*	*1*	*4*	*0*	*1272*
novice-27	*139*	*1195*	*1*	*63*	*414*	*7452*
novice-21	*127*	*849*	*1*	*42*	*0*	*5268*
scientist-3	*36*	*978*	*1*	*69*	*255*	*6398*
novice-22	*166*	*1893*	*1*	*51*	*547*	*11844*
novice-5	*121*	*593*	*1*	*67*	*0*	*3804*
novice-9	*134*	*853*	*0*	*63*	*0*	*5292*
novice-17	*132*	*1194*	*0*	*59*	*0*	*7702*
novice-23	*129*	*2138*	*0*	*72*	*0*	*13186*
novice-40	*165*	*967*	*0*	*24*	*722*	*6032*
scientist-26	*6*	*983*	*0*	*70*	*231*	*6388*
novice-4	*131*	*1919*	*0*	*123*	*0*	*11758*
novice-15	*141*	*1139*	*0*	*48*	*0*	*7148*
novice-32	*133*	*385*	*0*	*37*	*60*	*2512*
novice-54	*138*	*683*	*0*	*56*	*0*	*4248*
novice-26	*137*	*1120*	*0*	*60*	*0*	*7066*
novice-28	*136*	*2221*	*0*	*120*	*0*	*13816*
novice-35	*135*	*1444*	*0*	*54*	*50*	*9022*
novice-52	*140*	*650*	*0*	*38*	*0*	*4174*
novice-51	*143*	*480*	*0*	*20*	*0*	*3046*
scientist-32	*5*	*601*	*0*	*20*	*0*	*3916*
novice-49	*156*	*723*	*0*	*31*	*0*	*4428*
novice-31	*157*	*2073*	*0*	*102*	*18*	*12692*
novice-44	*158*	*1277*	*0*	*40*	*0*	*7896*
novice-38	*159*	*839*	*0*	*17*	*0*	*5468*
novice-2	*160*	*1267*	*0*	*58*	*0*	*8072*
novice-16	*161*	*256*	*0*	*25*	*0*	*1598*
novice-39	*163*	*1107*	*0*	*51*	*9*	*6936*
novice-18	*167*	*1088*	*0*	*38*	*0*	*6710*
novice-36	*142*	*3213*	*0*	*137*	*0*	*19840*

novice-37	*154*	*1949*	*0*	*57*	*0*	*12044*
scientist-33	*4*	*325*	*0*	*12*	*0*	*2044*
novice-14	*126*	*3194*	*0*	*208*	*0*	*19786*
novice-47	*148*	*1316*	*0*	*78*	*0*	*8118*
novice-30	*149*	*946*	*0*	*28*	*0*	*5986*
novice-3	*150*	*2337*	*0*	*93*	*0*	*15400*
scientist-8	*2*	*842*	*0*	*51*	*79*	*5294*
novice-48	*152*	*269*	*0*	*9*	*0*	*1704*
novice-8	*153*	*1822*	*0*	*19*	*0*	*11298*
novice-13	*168*	*652*	*0*	*49*	*0*	*4106*
experienced-15	*65*	*225*	*0*	*12*	*85*	*1404*
scientist-24	*29*	*1494*	*0*	*55*	*1217*	*9250*
non-23	*89*	*1294*	*0*	*48*	*636*	*8118*
non-1	*90*	*1622*	*0*	*59*	*410*	*10110*
non-21	*91*	*132*	*0*	*7*	*0*	*890*
non-19	*92*	*175*	*0*	*7*	*116*	*1356*
non-10	*93*	*495*	*0*	*20*	*0*	*3096*
non-15	*94*	*571*	*0*	*28*	*0*	*3736*
non-6	*95*	*177*	*0*	*7*	*0*	*1152*
scientist-11	*31*	*205*	*0*	*13*	*0*	*1380*
scientist-7	*33*	*366*	*0*	*28*	*169*	*2246*
experienced-6	*57*	*757*	*0*	*32*	*69*	*4752*
scientist-41	*49*	*2037*	*0*	*36*	*0*	*13036*
experienced-26	*72*	*679*	*0*	*66*	*59*	*4192*
scientist-17	*46*	*569*	*0*	*38*	*0*	*3792*
scientist-43	*39*	*3106*	*0*	*101*	*546*	*19066*
scientist-40	*38*	*4605*	*0*	*98*	*628*	*29020*
experienced-17	*79*	*2343*	*0*	*144*	*102*	*14396*
scientist-51	*34*	*910*	*0*	*67*	*358*	*5754*
non-24	*96*	*542*	*0*	*34*	*0*	*3390*
non-11	*98*	*1848*	*0*	*61*	*0*	*12210*
scientist-48	*9*	*819*	*0*	*43*	*0*	*5216*
novice-45	*116*	*651*	*0*	*16*	*0*	*4120*
novice-50	*117*	*985*	*0*	*92*	*0*	*6100*
novice-12	*118*	*2436*	*0*	*210*	*0*	*16366*
novice-42	*119*	*1068*	*0*	*33*	*5*	*6774*
novice-29	*120*	*1230*	*0*	*44*	*0*	*7754*
novice-33	*122*	*3127*	*0*	*106*	*0*	*19556*
novice-6	*123*	*871*	*0*	*44*	*0*	*5520*
novice-53	*124*	*1028*	*0*	*41*	*51*	*6558*
novice-41	*115*	*2317*	*0*	*51*	*1000*	*14244*
novice-10	*114*	*1464*	*0*	*40*	*872*	*9038*
non-13	*100*	*487*	*0*	*5*	*0*	*3072*
non-5	*103*	*244*	*0*	*11*	*0*	*1770*
scientist-16	*8*	*326*	*0*	*29*	*38*	*2250*
non-12	*107*	*216*	*0*	*26*	*0*	*1390*
non-17	*110*	*848*	*0*	*65*	*0*	*5330*
non-18	*111*	*1403*	*0*	*64*	*0*	*8804*
scientist-22	*7*	*750*	*0*	*39*	*20*	*5026*
non-2	*113*	*454*	*0*	*15*	*63*	*2934*
novice-43	*125*	*608*	*0*	*26*	*45*	*3778*

Appendix C: Generated Statistics for the Greenberg Dataset (Ordered by Errors)

User	Uid	Commands	History	Errors	Aliases	Lines
scientist-42	10	6068	6	644	3243	37598
experienced-7	55	5857	67	612	2926	35896
scientist-36	20	12056	488	566	3161	73434
novice-46	146	4163	112	372	1909	26080
experienced-20	53	4556	435	370	1646	28054
novice-19	164	3401	7	363	0	20816
scientist-4	23	4507	178	320	789	27992
scientist-52	15	7705	231	299	717	47280
experienced-1	80	3714	174	298	1906	22830
experienced-8	68	2930	67	265	625	18114
scientist-29	17	2683	20	243	530	16632
scientist-18	48	5584	6	240	1816	34258
experienced-24	86	3331	222	228	1456	20440
scientist-50	12	1496	219	225	387	9526
experienced-5	62	4015	35	222	910	25220
experienced-34	76	1869	206	218	598	11676
novice-25	155	2066	2	217	0	13070
novice-1	151	2457	37	213	1381	14960
novice-12	118	2436	0	210	0	16366
novice-14	126	3194	0	208	0	19786
scientist-20	45	2697	74	189	804	17080
scientist-14	47	3433	178	183	2	21404
experienced-35	78	4272	28	169	2504	26290
scientist-38	24	3775	48	168	1312	23172
non-4	106	5050	18	161	3296	30830
experienced-13	85	1109	25	160	446	6848
experienced-14	88	1810	23	153	996	11270
scientist-2	25	2954	236	149	1656	18514
experienced-17	79	2343	0	144	102	14396
novice-36	142	3213	0	137	0	19840
experienced-9	71	2351	86	136	502	14500
scientist-23	16	3360	52	135	481	21454
scientist-21	19	1762	50	134	586	10894
experienced-29	84	2214	59	133	1072	13566
non-20	99	4042	165	124	2122	24798
experienced-4	75	3776	2	123	1329	23258
novice-4	131	1919	0	123	0	11758
scientist-30	40	2129	186	123	409	13492
scientist-25	30	3508	7	122	379	22706
experienced-22	73	2814	325	122	560	17478
scientist-10	27	2024	77	120	730	12658
novice-28	136	2221	0	120	0	13816
experienced-23	81	2306	189	119	1004	14214
scientist-13	42	3593	357	118	204	21988
experienced-36	63	1580	56	116	781	9718
scientist-19	43	2831	106	112	1330	17560
scientist-1	13	1856	54	111	761	11792
experienced-30	66	2028	82	110	624	12686
scientist-46	3	2551	80	110	495	16480
novice-33	122	3127	0	106	0	19556
novice-31	157	2073	0	102	18	12692
scientist-43	39	3106	0	101	546	19066
scientist-40	38	4605	0	98	628	29020
scientist-49	51	1448	138	97	179	9106
novice-3	150	2337	0	93	0	15400
novice-50	117	985	0	92	0	6100
experienced-12	77	1763	106	92	889	10840

experienced-25	83	1465	69	89	346	9072
scientist-34	52	2639	15	88	910	16648
experienced-19	58	1807	163	88	829	11328
experienced-32	54	974	47	87	303	6102
experienced-11	70	1456	21	86	927	9126
scientist-27	37	3817	102	85	0	23344
scientist-44	44	1543	12	84	394	9544
scientist-37	32	4187	121	83	1866	25604
experienced-21	59	2394	157	83	974	14762
scientist-47	11	1229	9	81	618	7672
scientist-15	50	1429	200	81	175	9216
scientist-5	21	1563	18	78	558	10164
novice-47	148	1316	0	78	0	8118
scientist-39	41	1753	173	77	530	10992
novice-23	129	2138	0	72	0	13186
scientist-26	6	983	0	70	231	6388
scientist-3	36	978	1	69	255	6398
novice-5	121	593	1	67	0	3804
scientist-51	34	910	0	67	358	5754
experienced-26	72	679	0	66	59	4192
scientist-9	26	4067	224	65	665	25424
experienced-33	82	1292	83	65	649	7986
non-17	110	848	0	65	0	5330
non-18	111	1403	0	64	0	8804
novice-27	139	1195	1	63	414	7452
novice-9	134	853	0	63	0	5292
non-11	98	1848	0	61	0	12210
novice-26	137	1120	0	60	0	7066
experienced-28	74	3893	78	60	2516	25116
novice-17	132	1194	0	59	0	7702
scientist-45	35	862	17	59	223	5330
non-1	90	1622	0	59	410	10110
novice-2	160	1267	0	58	0	8072
novice-37	154	1949	0	57	0	12044
non-22	112	1567	48	56	0	10004
novice-54	138	683	0	56	0	4248
scientist-24	29	1494	0	55	1217	9250
non-7	101	1231	3	54	792	7704
experienced-27	67	1693	77	54	741	11032
novice-35	135	1444	0	54	50	9022
novice-24	130	849	48	53	118	5436
scientist-12	28	2499	53	52	1162	15412
novice-22	166	1893	1	51	547	11844
scientist-8	2	842	0	51	79	5294
novice-41	115	2317	0	51	1000	14244
novice-39	163	1107	0	51	9	6936
novice-7	145	1039	98	51	36	6608
novice-13	168	652	0	49	0	4106
scientist-6	14	1103	33	49	278	7196
non-23	89	1294	0	48	636	8118
novice-15	141	1139	0	48	0	7148
novice-34	147	1276	4	46	0	8146
novice-6	123	871	0	44	0	5520
novice-29	120	1230	0	44	0	7754
scientist-48	9	819	0	43	0	5216
experienced-3	87	915	88	42	356	5600
novice-21	127	849	1	42	0	5268
novice-53	124	1028	0	41	51	6558
novice-10	114	1464	0	40	872	9038
novice-44	158	1277	0	40	0	7896

novice-55	128	1662	6	40	0	10218
scientist-22	7	750	0	39	20	5026
experienced-31	61	683	19	38	454	4368
scientist-17	46	569	0	38	0	3792
novice-18	167	1088	0	38	0	6710
novice-52	140	650	0	38	0	4174
novice-32	133	385	0	37	60	2512
scientist-41	49	2037	0	36	0	13036
non-24	96	542	0	34	0	3390
novice-42	119	1068	0	33	5	6774
experienced-6	57	757	0	32	69	4752
novice-49	156	723	0	31	0	4428
scientist-16	8	326	0	29	38	2250
scientist-35	22	1049	23	29	594	6612
scientist-7	33	366	0	28	169	2246
novice-30	149	946	0	28	0	5986
non-15	94	571	0	28	0	3736
non-12	107	216	0	26	0	1390
scientist-28	18	765	64	26	235	5032
novice-43	125	608	0	26	45	3778
non-16	105	821	144	26	0	5108
experienced-10	56	446	2	26	170	2774
novice-16	161	256	0	25	0	1598
novice-40	165	967	0	24	722	6032
non-9	102	357	4	23	45	2432
experienced-16	60	795	24	22	245	4932
novice-11	162	256	2	21	0	1770
experienced-18	69	575	5	21	114	3548
non-10	93	495	0	20	0	3096
novice-51	143	480	0	20	0	3046
scientist-32	5	601	0	20	0	3916
scientist-31	1	250	9	20	12	1758
novice-20	144	418	5	19	0	2722
novice-8	153	1822	0	19	0	11298
non-25	104	327	3	18	48	2264
novice-38	159	839	0	17	0	5468
novice-45	116	651	0	16	0	4120
non-3	108	1265	9	15	209	7928
non-2	113	454	0	15	63	2934
scientist-11	31	205	0	13	0	1380
non-8	97	239	28	13	18	1524
experienced-15	65	225	0	12	85	1404
scientist-33	4	325	0	12	0	2044
non-5	103	244	0	11	0	1770
experienced-2	64	219	6	11	33	1414
novice-48	152	269	0	9	0	1704
non-19	92	175	0	7	116	1356
non-6	95	177	0	7	0	1152
non-21	91	132	0	7	0	890
non-13	100	487	0	5	0	3072
non-14	109	201	1	4	0	1272

Appendix D: Generated Statistics for the Greenberg Dataset (Ordered by Aliases)

User	Uid	Commands	History	Errors	Aliases	Lines
non-4	106	5050	18	161	3296	30830
scientist-42	10	6068	6	644	3243	37598
scientist-36	20	12056	488	566	3161	73434
experienced-7	55	5857	67	612	2926	35896
experienced-28	74	3893	78	60	2516	25116
experienced-35	78	4272	28	169	2504	26290
non-20	99	4042	165	124	2122	24798
novice-46	146	4163	112	372	1909	26080
experienced-1	80	3714	174	298	1906	22830
scientist-37	32	4187	121	83	1866	25604
scientist-18	48	5584	6	240	1816	34258
scientist-2	25	2954	236	149	1656	18514
experienced-20	53	4556	435	370	1646	28054
experienced-24	86	3331	222	228	1456	20440
novice-1	151	2457	37	213	1381	14960
scientist-19	43	2831	106	112	1330	17560
experienced-4	75	3776	2	123	1329	23258
scientist-38	24	3775	48	168	1312	23172
scientist-24	29	1494	0	55	1217	9250
scientist-12	28	2499	53	52	1162	15412
experienced-29	84	2214	59	133	1072	13566
experienced-23	81	2306	189	119	1004	14214
novice-41	115	2317	0	51	1000	14244
experienced-14	88	1810	23	153	996	11270
experienced-21	59	2394	157	83	974	14762
experienced-11	70	1456	21	86	927	9126
scientist-34	52	2639	15	88	910	16648
experienced-5	62	4015	35	222	910	25220
experienced-12	77	1763	106	92	889	10840
novice-10	114	1464	0	40	872	9038
experienced-19	58	1807	163	88	829	11328
scientist-20	45	2697	74	189	804	17080
non-7	101	1231	3	54	792	7704
scientist-4	23	4507	178	320	789	27992
experienced-36	63	1580	56	116	781	9718
scientist-1	13	1856	54	111	761	11792
experienced-27	67	1693	77	54	741	11032
scientist-10	27	2024	77	120	730	12658
novice-40	165	967	0	24	722	6032
scientist-52	15	7705	231	299	717	47280
scientist-9	26	4067	224	65	665	25424
experienced-33	82	1292	83	65	649	7986
non-23	89	1294	0	48	636	8118
scientist-40	38	4605	0	98	628	29020
experienced-8	68	2930	67	265	625	18114
experienced-30	66	2028	82	110	624	12686
scientist-47	11	1229	9	81	618	7672
experienced-34	76	1869	206	218	598	11676
scientist-35	22	1049	23	29	594	6612
scientist-21	19	1762	50	134	586	10894
experienced-22	73	2814	325	122	560	17478
scientist-5	21	1563	18	78	558	10164
novice-22	166	1893	1	51	547	11844
scientist-43	39	3106	0	101	546	19066
scientist-29	17	2683	20	243	530	16632
scientist-39	41	1753	173	77	530	10992
experienced-9	71	2351	86	136	502	14500

scientist-46	*3*	*2551*	*80*	*110*	*495*	*16480*
scientist-23	*16*	*3360*	*52*	*135*	*481*	*21454*
experienced-31	*61*	*683*	*19*	*38*	*454*	*4368*
experienced-13	*85*	*1109*	*25*	*160*	*446*	*6848*
novice-27	*139*	*1195*	*1*	*63*	*414*	*7452*
non-1	*90*	*1622*	*0*	*59*	*410*	*10110*
scientist-30	*40*	*2129*	*186*	*123*	*409*	*13492*
scientist-44	*44*	*1543*	*12*	*84*	*394*	*9544*
scientist-50	*12*	*1496*	*219*	*225*	*387*	*9526*
scientist-25	*30*	*3508*	*7*	*122*	*379*	*22706*
scientist-51	*34*	*910*	*0*	*67*	*358*	*5754*
experienced-3	*87*	*915*	*88*	*42*	*356*	*5600*
experienced-25	*83*	*1465*	*69*	*89*	*346*	*9072*
experienced-32	*54*	*974*	*47*	*87*	*303*	*6102*
scientist-6	*14*	*1103*	*33*	*49*	*278*	*7196*
scientist-3	*36*	*978*	*1*	*69*	*255*	*6398*
experienced-16	*60*	*795*	*24*	*22*	*245*	*4932*
scientist-28	*18*	*765*	*64*	*26*	*235*	*5032*
scientist-26	*6*	*983*	*0*	*70*	*231*	*6388*
scientist-45	*35*	*862*	*17*	*59*	*223*	*5330*
non-3	*108*	*1265*	*9*	*15*	*209*	*7928*
scientist-13	*42*	*3593*	*357*	*118*	*204*	*21988*
scientist-49	*51*	*1448*	*138*	*97*	*179*	*9106*
scientist-15	*50*	*1429*	*200*	*81*	*175*	*9216*
experienced-10	*56*	*446*	*2*	*26*	*170*	*2774*
scientist-7	*33*	*366*	*0*	*28*	*169*	*2246*
novice-24	*130*	*849*	*48*	*53*	*118*	*5436*
non-19	*92*	*175*	*0*	*7*	*116*	*1356*
experienced-18	*69*	*575*	*5*	*21*	*114*	*3548*
experienced-17	*79*	*2343*	*0*	*144*	*102*	*14396*
experienced-15	*65*	*225*	*0*	*12*	*85*	*1404*
scientist-8	*2*	*842*	*0*	*51*	*79*	*5294*
experienced-6	*57*	*757*	*0*	*32*	*69*	*4752*
non-2	*113*	*454*	*0*	*15*	*63*	*2934*
novice-32	*133*	*385*	*0*	*37*	*60*	*2512*
experienced-26	*72*	*679*	*0*	*66*	*59*	*4192*
novice-53	*124*	*1028*	*0*	*41*	*51*	*6558*
novice-35	*135*	*1444*	*0*	*54*	*50*	*9022*
non-25	*104*	*327*	*3*	*18*	*48*	*2264*
non-9	*102*	*357*	*4*	*23*	*45*	*2432*
novice-43	*125*	*608*	*0*	*26*	*45*	*3778*
scientist-16	*8*	*326*	*0*	*29*	*38*	*2250*
novice-7	*145*	*1039*	*98*	*51*	*36*	*6608*
experienced-2	*64*	*219*	*6*	*11*	*33*	*1414*
scientist-22	*7*	*750*	*0*	*39*	*20*	*5026*
non-8	*97*	*239*	*28*	*13*	*18*	*1524*
novice-31	*157*	*2073*	*0*	*102*	*18*	*12692*
scientist-31	*1*	*250*	*9*	*20*	*12*	*1758*
novice-39	*163*	*1107*	*0*	*51*	*9*	*6936*
novice-42	*119*	*1068*	*0*	*33*	*5*	*6774*
scientist-14	*47*	*3433*	*178*	*183*	*2*	*21404*
novice-30	*149*	*946*	*0*	*28*	*0*	*5986*
novice-47	*148*	*1316*	*0*	*78*	*0*	*8118*
novice-3	*150*	*2337*	*0*	*93*	*0*	*15400*
novice-34	*147*	*1276*	*4*	*46*	*0*	*8146*
scientist-17	*46*	*569*	*0*	*38*	*0*	*3792*
novice-20	*144*	*418*	*5*	*19*	*0*	*2722*
novice-51	*143*	*480*	*0*	*20*	*0*	*3046*
novice-36	*142*	*3213*	*0*	*137*	*0*	*19840*
novice-15	*141*	*1139*	*0*	*48*	*0*	*7148*

novice-52	*140*	*650*	*0*	*38*	*0*	*4174*
scientist-27	*37*	*3817*	*102*	*85*	*0*	*23344*
scientist-32	*5*	*601*	*0*	*20*	*0*	*3916*
novice-48	*152*	*269*	*0*	*9*	*0*	*1704*
novice-8	*153*	*1822*	*0*	*19*	*0*	*11298*
novice-13	*168*	*652*	*0*	*49*	*0*	*4106*
novice-18	*167*	*1088*	*0*	*38*	*0*	*6710*
scientist-41	*49*	*2037*	*0*	*36*	*0*	*13036*
scientist-33	*4*	*325*	*0*	*12*	*0*	*2044*
novice-19	*164*	*3401*	*7*	*363*	*0*	*20816*
novice-11	*162*	*256*	*2*	*21*	*0*	*1770*
novice-16	*161*	*256*	*0*	*25*	*0*	*1598*
novice-2	*160*	*1267*	*0*	*58*	*0*	*8072*
novice-38	*159*	*839*	*0*	*17*	*0*	*5468*
novice-44	*158*	*1277*	*0*	*40*	*0*	*7896*
novice-49	*156*	*723*	*0*	*31*	*0*	*4428*
novice-25	*155*	*2066*	*2*	*217*	*0*	*13070*
novice-37	*154*	*1949*	*0*	*57*	*0*	*12044*
novice-54	*138*	*683*	*0*	*56*	*0*	*4248*
novice-26	*137*	*1120*	*0*	*60*	*0*	*7066*
novice-28	*136*	*2221*	*0*	*120*	*0*	*13816*
novice-12	*118*	*2436*	*0*	*210*	*0*	*16366*
novice-50	*117*	*985*	*0*	*92*	*0*	*6100*
novice-45	*116*	*651*	*0*	*16*	*0*	*4120*
non-6	*95*	*177*	*0*	*7*	*0*	*1152*
non-24	*96*	*542*	*0*	*34*	*0*	*3390*
non-22	*112*	*1567*	*48*	*56*	*0*	*10004*
non-18	*111*	*1403*	*0*	*64*	*0*	*8804*
non-17	*110*	*848*	*0*	*65*	*0*	*5330*
non-14	*109*	*201*	*1*	*4*	*0*	*1272*
non-11	*98*	*1848*	*0*	*61*	*0*	*12210*
non-12	*107*	*216*	*0*	*26*	*0*	*1390*
non-13	*100*	*487*	*0*	*5*	*0*	*3072*
non-16	*105*	*821*	*144*	*26*	*0*	*5108*
non-5	*103*	*244*	*0*	*11*	*0*	*1770*
novice-29	*120*	*1230*	*0*	*44*	*0*	*7754*
novice-5	*121*	*593*	*1*	*67*	*0*	*3804*
scientist-11	*31*	*205*	*0*	*13*	*0*	*1380*
novice-9	*134*	*853*	*0*	*63*	*0*	*5292*
novice-17	*132*	*1194*	*0*	*59*	*0*	*7702*
novice-4	*131*	*1919*	*0*	*123*	*0*	*11758*
non-21	*91*	*132*	*0*	*7*	*0*	*890*
novice-23	*129*	*2138*	*0*	*72*	*0*	*13186*
novice-55	*128*	*1662*	*6*	*40*	*0*	*10218*
novice-21	*127*	*849*	*1*	*42*	*0*	*5268*
novice-14	*126*	*3194*	*0*	*208*	*0*	*19786*
non-10	*93*	*495*	*0*	*20*	*0*	*3096*
non-15	*94*	*571*	*0*	*28*	*0*	*3736*
novice-6	*123*	*871*	*0*	*44*	*0*	*5520*
novice-33	*122*	*3127*	*0*	*106*	*0*	*19556*
scientist-48	*9*	*819*	*0*	*43*	*0*	*5216*

Appendix E: Generated Statistics for the Greenberg Dataset (Ordered by Lines)

User	Uid	Commands	History	Errors	Aliases	Lines
scientist-36	20	12056	488	566	3161	73434
scientist-52	15	7705	231	299	717	47280
scientist-42	10	6068	6	644	3243	37598
experienced-7	55	5857	67	612	2926	35896
scientist-18	48	5584	6	240	1816	34258
non-4	106	5050	18	161	3296	30830
scientist-40	38	4605	0	98	628	29020
experienced-20	53	4556	435	370	1646	28054
scientist-4	23	4507	178	320	789	27992
experienced-35	78	4272	28	169	2504	26290
novice-46	146	4163	112	372	1909	26080
scientist-37	32	4187	121	83	1866	25604
scientist-9	26	4067	224	65	665	25424
experienced-5	62	4015	35	222	910	25220
experienced-28	74	3893	78	60	2516	25116
non-20	99	4042	165	124	2122	24798
scientist-27	37	3817	102	85	0	23344
experienced-4	75	3776	2	123	1329	23258
scientist-38	24	3775	48	168	1312	23172
experienced-1	80	3714	174	298	1906	22830
scientist-25	30	3508	7	122	379	22706
scientist-13	42	3593	357	118	204	21988
scientist-23	16	3360	52	135	481	21454
scientist-14	47	3433	178	183	2	21404
novice-19	164	3401	7	363	0	20816
experienced-24	86	3331	222	228	1456	20440
novice-36	142	3213	0	137	0	19840
novice-14	126	3194	0	208	0	19786
novice-33	122	3127	0	106	0	19556
scientist-43	39	3106	0	101	546	19066
scientist-2	25	2954	236	149	1656	18514
experienced-8	68	2930	67	265	625	18114
scientist-19	43	2831	106	112	1330	17560
experienced-22	73	2814	325	122	560	17478
scientist-20	45	2697	74	189	804	17080
scientist-34	52	2639	15	88	910	16648
scientist-29	17	2683	20	243	530	16632
scientist-46	3	2551	80	110	495	16480
novice-12	118	2436	0	210	0	16366
scientist-12	28	2499	53	52	1162	15412
novice-3	150	2337	0	93	0	15400
novice-1	151	2457	37	213	1381	14960
experienced-21	59	2394	157	83	974	14762
experienced-9	71	2351	86	136	502	14500
experienced-17	79	2343	0	144	102	14396
novice-41	115	2317	0	51	1000	14244
experienced-23	81	2306	189	119	1004	14214
novice-28	136	2221	0	120	0	13816
experienced-29	84	2214	59	133	1072	13566
scientist-30	40	2129	186	123	409	13492
novice-23	129	2138	0	72	0	13186
novice-25	155	2066	2	217	0	13070
scientist-41	49	2037	0	36	0	13036
novice-31	157	2073	0	102	18	12692
experienced-30	66	2028	82	110	624	12686
scientist-10	27	2024	77	120	730	12658
non-11	98	1848	0	61	0	12210

novice-37	*154*	*1949*	*0*	*57*	*0*	*12044*
novice-22	*166*	*1893*	*1*	*51*	*547*	*11844*
scientist-1	*13*	*1856*	*54*	*111*	*761*	*11792*
novice-4	*131*	*1919*	*0*	*123*	*0*	*11758*
experienced-34	*76*	*1869*	*206*	*218*	*598*	*11676*
experienced-19	*58*	*1807*	*163*	*88*	*829*	*11328*
novice-8	*153*	*1822*	*0*	*19*	*0*	*11298*
experienced-14	*88*	*1810*	*23*	*153*	*996*	*11270*
experienced-27	*67*	*1693*	*77*	*54*	*741*	*11032*
scientist-39	*41*	*1753*	*173*	*77*	*530*	*10992*
scientist-21	*19*	*1762*	*50*	*134*	*586*	*10894*
experienced-12	*77*	*1763*	*106*	*92*	*889*	*10840*
novice-55	*128*	*1662*	*6*	*40*	*0*	*10218*
scientist-5	*21*	*1563*	*18*	*78*	*558*	*10164*
non-1	*90*	*1622*	*0*	*59*	*410*	*10110*
non-22	*112*	*1567*	*48*	*56*	*0*	*10004*
experienced-36	*63*	*1580*	*56*	*116*	*781*	*9718*
scientist-44	*44*	*1543*	*12*	*84*	*394*	*9544*
scientist-50	*12*	*1496*	*219*	*225*	*387*	*9526*
scientist-24	*29*	*1494*	*0*	*55*	*1217*	*9250*
scientist-15	*50*	*1429*	*200*	*81*	*175*	*9216*
experienced-11	*70*	*1456*	*21*	*86*	*927*	*9126*
scientist-49	*51*	*1448*	*138*	*97*	*179*	*9106*
experienced-25	*83*	*1465*	*69*	*89*	*346*	*9072*
novice-10	*114*	*1464*	*0*	*40*	*872*	*9038*
novice-35	*135*	*1444*	*0*	*54*	*50*	*9022*
non-18	*111*	*1403*	*0*	*64*	*0*	*8804*
novice-34	*147*	*1276*	*4*	*46*	*0*	*8146*
novice-47	*148*	*1316*	*0*	*78*	*0*	*8118*
non-23	*89*	*1294*	*0*	*48*	*636*	*8118*
novice-2	*160*	*1267*	*0*	*58*	*0*	*8072*
experienced-33	*82*	*1292*	*83*	*65*	*649*	*7986*
non-3	*108*	*1265*	*9*	*15*	*209*	*7928*
novice-44	*158*	*1277*	*0*	*40*	*0*	*7896*
novice-29	*120*	*1230*	*0*	*44*	*0*	*7754*
non-7	*101*	*1231*	*3*	*54*	*792*	*7704*
novice-17	*132*	*1194*	*0*	*59*	*0*	*7702*
scientist-47	*11*	*1229*	*9*	*81*	*618*	*7672*
novice-27	*139*	*1195*	*1*	*63*	*414*	*7452*
scientist-6	*14*	*1103*	*33*	*49*	*278*	*7196*
novice-15	*141*	*1139*	*0*	*48*	*0*	*7148*
novice-26	*137*	*1120*	*0*	*60*	*0*	*7066*
novice-39	*163*	*1107*	*0*	*51*	*9*	*6936*
experienced-13	*85*	*1109*	*25*	*160*	*446*	*6848*
novice-42	*119*	*1068*	*0*	*33*	*5*	*6774*
novice-18	*167*	*1088*	*0*	*38*	*0*	*6710*
scientist-35	*22*	*1049*	*23*	*29*	*594*	*6612*
novice-7	*145*	*1039*	*98*	*51*	*36*	*6608*
novice-53	*124*	*1028*	*0*	*41*	*51*	*6558*
scientist-3	*36*	*978*	*1*	*69*	*255*	*6398*
scientist-26	*6*	*983*	*0*	*70*	*231*	*6388*
experienced-32	*54*	*974*	*47*	*87*	*303*	*6102*
novice-50	*117*	*985*	*0*	*92*	*0*	*6100*
novice-40	*165*	*967*	*0*	*24*	*722*	*6032*
novice-30	*149*	*946*	*0*	*28*	*0*	*5986*
scientist-51	*34*	*910*	*0*	*67*	*358*	*5754*
experienced-3	*87*	*915*	*88*	*42*	*356*	*5600*
novice-6	*123*	*871*	*0*	*44*	*0*	*5520*
novice-38	*159*	*839*	*0*	*17*	*0*	*5468*
novice-24	*130*	*849*	*48*	*53*	*118*	*5436*

scientist-45	*35*	*862*	*17*	*59*	*223*	*5330*
non-17	*110*	*848*	*0*	*65*	*0*	*5330*
scientist-8	*2*	*842*	*0*	*51*	*79*	*5294*
novice-9	*134*	*853*	*0*	*63*	*0*	*5292*
novice-21	*127*	*849*	*1*	*42*	*0*	*5268*
scientist-48	*9*	*819*	*0*	*43*	*0*	*5216*
non-16	*105*	*821*	*144*	*26*	*0*	*5108*
scientist-28	*18*	*765*	*64*	*26*	*235*	*5032*
scientist-22	*7*	*750*	*0*	*39*	*20*	*5026*
experienced-16	*60*	*795*	*24*	*22*	*245*	*4932*
experienced-6	*57*	*757*	*0*	*32*	*69*	*4752*
novice-49	*156*	*723*	*0*	*31*	*0*	*4428*
experienced-31	*61*	*683*	*19*	*38*	*454*	*4368*
novice-54	*138*	*683*	*0*	*56*	*0*	*4248*
experienced-26	*72*	*679*	*0*	*66*	*59*	*4192*
novice-52	*140*	*650*	*0*	*38*	*0*	*4174*
novice-45	*116*	*651*	*0*	*16*	*0*	*4120*
novice-13	*168*	*652*	*0*	*49*	*0*	*4106*
scientist-32	*5*	*601*	*0*	*20*	*0*	*3916*
novice-5	*121*	*593*	*1*	*67*	*0*	*3804*
scientist-17	*46*	*569*	*0*	*38*	*0*	*3792*
novice-43	*125*	*608*	*0*	*26*	*45*	*3778*
non-15	*94*	*571*	*0*	*28*	*0*	*3736*
experienced-18	*69*	*575*	*5*	*21*	*114*	*3548*
non-24	*96*	*542*	*0*	*34*	*0*	*3390*
non-10	*93*	*495*	*0*	*20*	*0*	*3096*
non-13	*100*	*487*	*0*	*5*	*0*	*3072*
novice-51	*143*	*480*	*0*	*20*	*0*	*3046*
non-2	*113*	*454*	*0*	*15*	*63*	*2934*
experienced-10	*56*	*446*	*2*	*26*	*170*	*2774*
novice-20	*144*	*418*	*5*	*19*	*0*	*2722*
novice-32	*133*	*385*	*0*	*37*	*60*	*2512*
non-9	*102*	*357*	*4*	*23*	*45*	*2432*
non-25	*104*	*327*	*3*	*18*	*48*	*2264*
scientist-16	*8*	*326*	*0*	*29*	*38*	*2250*
scientist-7	*33*	*366*	*0*	*28*	*169*	*2246*
scientist-33	*4*	*325*	*0*	*12*	*0*	*2044*
non-5	*103*	*244*	*0*	*11*	*0*	*1770*
novice-11	*162*	*256*	*2*	*21*	*0*	*1770*
scientist-31	*1*	*250*	*9*	*20*	*12*	*1758*
novice-48	*152*	*269*	*0*	*9*	*0*	*1704*
novice-16	*161*	*256*	*0*	*25*	*0*	*1598*
non-8	*97*	*239*	*28*	*13*	*18*	*1524*
experienced-2	*64*	*219*	*6*	*11*	*33*	*1414*
experienced-15	*65*	*225*	*0*	*12*	*85*	*1404*
non-12	*107*	*216*	*0*	*26*	*0*	*1390*
scientist-11	*31*	*205*	*0*	*13*	*0*	*1380*
non-19	*92*	*175*	*0*	*7*	*116*	*1356*
non-14	*109*	*201*	*1*	*4*	*0*	*1272*
non-6	*95*	*177*	*0*	*7*	*0*	*1152*
non-21	*91*	*132*	*0*	*7*	*0*	*890*

Appendix F: List of User Profiles (Victims)

Uid	*Number of Distinct Commands*	*Uid*	*Number of Distinct Commnds*
3	*427*	*73*	*234*
10	*427*	*74*	*350*
13	*425*	*75*	*193*
15	*316*	*76*	*244*
16	*273*	*77*	*336*
17	*507*	*78*	*241*
19	*412*	*79*	*338*
20	*376*	*80*	*241*
21	*345*	*81*	*188*
23	*351*	*84*	*230*
24	*426*	*86*	*420*
25	*286*	*88*	*274*
26	*311*	*90*	*341*
27	*312*	*98*	*472*
28	*314*	*99*	*262*
30	*462*	*106*	*322*
32	*320*	*112*	*277*
37	*275*	*115*	*252*
38	*329*	*118*	*138*
39	*349*	*122*	*138*
40	*389*	*126*	*358*
41	*342*	*128*	*137*
42	*222*	*129*	*126*
43	*307*	*131*	*255*
44	*367*	*136*	*195*
45	*422*	*142*	*157*
47	*334*	*146*	*339*
48	*314*	*150*	*119*
49	*267*	*151*	*284*
52	*214*	*153*	*107*
53	*168*	*154*	*230*
55	*339*	*155*	*339*
58	*321*	*157*	*129*
59	*195*	*164*	*206*
62	*267*	*166*	*208*
63	*274*		
66	*204*		
67	*341*		
68	*335*		
71	*341*		

Appendix G: Undecided Results

Uid Input	Uid Profile	Decision Expected	Interval Size	p_1	p_2	α	β
70	52	Reject	10	0.29	0.71	0.01	0.01
22	52	Reject	10	0.29	0.71	0.01	0.01
82	52	Reject	10	0.29	0.71	0.01	0.01
70	52	Reject	10	0.30	0.70	0.01	0.01
22	52	Reject	10	0.30	0.70	0.01	0.01
82	52	Reject	10	0.30	0.70	0.01	0.01
108	84	Reject	10	0.31	0.69	0.01	0.01
70	84	Reject	10	0.31	0.69	0.01	0.01
22	84	Reject	10	0.31	0.69	0.01	0.01
82	84	Reject	10	0.31	0.69	0.01	0.01
13	13	Accept	15	0.29	0.71	0.01	0.01
13	13	Accept	20	0.31	0.69	0.01	0.01
13	13	Accept	15	0.30	0.7	0.01	0.01
13	13	Accept	15	0.30	0.7	0.01	0.01
13	13	Accept	15	0.31	0.69	0.01	0.01
13	13	Accept	20	0.29	0.71	0.01	0.01
13	13	Accept	10	0.31	0.69	0.01	0.01
13	13	Accept	10	0.30	0.7	0.01	0.01
13	13	Accept	20	0.30	0.7	0.01	0.01
30	30	Accept	15	0.31	0.69	0.01	0.01
30	30	Accept	5	0.45	0.55	0.01	0.01
30	30	Accept	20	0.29	0.71	0.01	0.01
30	30	Accept	15	0.30	0.7	0.01	0.01
30	30	Accept	20	0.30	0.7	0.01	0.01
30	30	Accept	20	0.31	0.69	0.01	0.01
30	30	Accept	15	0.30	0.7	0.01	0.01
30	30	Accept	15	0.29	0.71	0.01	0.01
55	55	Accept	20	0.30	0.7	0.01	0.01
55	55	Accept	20	0.29	0.71	0.01	0.01
55	55	Accept	15	0.31	0.69	0.01	0.01
55	55	Accept	15	0.30	0.7	0.01	0.01
55	55	Accept	10	0.31	0.69	0.01	0.01
55	55	Accept	10	0.29	0.71	0.01	0.01
55	55	Accept	20	0.31	0.69	0.01	0.01
55	55	Accept	15	0.30	0.7	0.01	0.01
55	55	Accept	15	0.29	0.71	0.01	0.01
55	55	Accept	10	0.30	0.7	0.01	0.01
58	58	Accept	20	0.31	0.69	0.01	0.01
58	58	Accept	20	0.30	0.7	0.01	0.01
78	78	Accept	20	0.31	0.69	0.01	0.01
98	98	Accept	15	0.31	0.69	0.01	0.01
98	98	Accept	20	0.31	0.69	0.01	0.01
98	98	Accept	20	0.30	0.7	0.01	0.01
98	98	Accept	20	0.29	0.71	0.01	0.01
136	136	Accept	20	0.31	0.69	0.01	0.01
136	136	Accept	20	0.30	0.7	0.01	0.01
136	136	Accept	20	0.29	0.71	0.01	0.01

References

[1] Ahmed, A. A., & Traore, I. (2008). Performance Metrics and Model for Continuous Authentication Systems. *submitted to the IEEE Transactions on Dependable and.*

[2] O'Gorman, L. (2003). Comparing Passwords, Tokens, and Biometrics for User Authentication. *IEEE, Dec. 03 Vol. 91*, (pp. 2021-2040).

[3] Shepherd, S. (1995). Continuous authentication by analysis of keyboard typing characteristics. *Security and Detection, European Convention*, (pp. 111-114). Brighton.

[4] Ahmed, A., & Traore, I. (2007). A New Biometric Technology Based on Mouse Dynamics. *Dependable and Secure Computing, IEEE Transactions*, (pp. 165-179).

[5] Loeb, V. (2001). Spy Case Prompts Computer Search. *Washington Post, DC* , A01-00.

[6] Maxion, R., & Townsend, T. (2004). Masquerade detection augmented with error analysis. *Reliability, IEEE Transactions* (pp. 124-147). Pittsburgh: IEEE Reliability Society.

[7] Kim, H., & Cha, S. (2005). Empirical evaluation of SVM-based masquerade detection using UNIX commands. *Computer & Security , 24*, 160-168.

[8] Wang, K., & S, S. (2003). One-class training for masquerade detection. *3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security.*

[9] Schonlau, M., DuMouchel, W., Ju, W., Karr, A. F., Theus, M., & Vardi, Y. (2001). Computer Intrusion: Detecting masquerades. *Statistical Science , I* (16), 58-74.

[10] Maxion, R., & Townsend, T. (2002). Masquerade detection using truncated command lines. *Dependable Systems and Networks. DSN 2002. Proceedings. International Conference*, (pp. 219-228). Pittsburgh.

[11] Axelsson, S. (2000, March 14). Intrusion Detection Systems: A Survey and Taxonomy. 1-27. Sweden.

[12] Armstrong, D., & Peile, C. (2005). Perimeter intruder detection systems performance standard. *Security Technology, 2005. CCST '05. 39th Annual 2005 International Carnahan Conference*, (pp. 33-36). Las Palmas.

[13] Cannady, J. (1998). *Artificial Neural Networks for Misuse Detection.* Retrieved from http://csrc.nist.gov/nissc/1998/proceedings/paperF13.pdf

[14] Ghosh, A. K., & Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8* , 12-12.

[15] Jali, K. A., Kamarudin, M. H., & Masrek, M. N. (2010). Comparison of Machine Learning algorithms performance in detecting network intrusion. *Networking and Information Technology (ICNIT), International Conference*, (pp. 221-226). Manila, Philippines.

[16] Kemmerer, R. A., & Vigna, G. (2002). Intrusion detection: A brief history and overview. *Computer 35 (SUPPL)* , 27-30.

[17] Lin, M., Miikkulainen, R., & Ryan, J. (1998, June). Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems* .

[18] Liu, Z., Florez, G., & Bridges, S. (2002). A comparison of input representations in neural networks: a case study in intrusion detection. *Neural Networks. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, (pp. 1708-1713). Honolulu, HI.

[19] Zhang, P. (2005). Neural Networks. In *Data Mining and Knowledge Discovery Handbook* (pp. 487-507). Springer US.

[20] Mitchell, T. (1997). *Machine Learning.* The McGraw-Hill Companies, Inc.

[21] Kim, S.-B., Han, K.-S., Rim, H.-C., & Myaeng, S. H. (2006). Some Effective Techniques for Naive Bayes Text Classification. *Knowledge and Data Engineering, IEEE Transactions*, (pp. 1457-1466). Los Angeles.

[22] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification.* New York: Ellis Horwood.

[23] Almeida, T., Yamakami, A., & Almeida, J. (2009). Evaluation of Approaches for Dimensionality Reduction Applied with Naive Bayes Anti-Spam Filters. *Machine Learning and Applications. ICMLA '09. International Conference*, (pp. 517-522). Miami Beach, FL.

[24] Yang, Z., Nie, X., Xu, W., & Guo, J. (2006). An Approach to Spam Detection by Naive Bayes Ensemble Based on Decision Induction. *Intelligent Systems Design and Applications. ISDA '06. Sixth International Conference*, (pp. 861-866). Jinan.

[25] Joachims, T. (1996). *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization.* Carnegie Mellon University: Computer Science Technical Report CMU-CS-96-118.

[26] Maxion, R. (2003). Masquerade detection using enriched command lines. *Dependable Systems and Networks. Proceedings. 2003 International Conference*, (pp. 5-14).

[27] Maxion, R., & Tan, K. (2000). Benchmarking anomaly-based detection systems. *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference*, (pp. 623-630). New York, NY.

[28] Jiang, L., Zhang, H., & Cai, Z. (2009). A Novel Bayes Model: Hidden Naive Bayes. *Knowledge and Data Engineering, IEEE Transactions*, (pp. 1361-1371).

[29] Garg, A., Rahalkar, R., Upadhyaya, S., & Kwiat, K. (2006). Profiling Users in GUI Based Systems for Masquerade Detection. West Point: Information Assurance Workshop IEEE.

[30] Ghosh, A. K., Schwartzbard, A., & Schatz, M. (1999). Learning Program Behavior Profiles for Intrusion Detection.

[31] Ghosh, A. K., Wanken, J., & Charron, F. (1998). Detecting anomalous and unknown intrusions against programs. *In Proceedings of the 1998 Annual Computer Security Applications Conference* .

[32] Heywood, M., Lichodzijewski, P., & Zincir-Heywood, N. (2001). *Host-Based Intrusion Detection Using Self-Organizing Maps.* Retrieved from http://users.cs.dal.ca/~mheywood/X-files/Publications/PeterSOM-ids.pdf

[33] Hoglund, A. J., Hatonen, K., & Sorvari, A. S. (2000). A computer host-based user anomaly detection system using theself-organizing map. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on* , 411 - 416.

[34] Wang, W., Guan, X., & Zhan, X. (2004). Profiling program and user behaviors for anomaly intrusion detection based on non-negative matrix factorization. *Decision and Control, 2004. CDC. 43rd IEEE Conference*, (pp. 99 - 104 Vol.1). Atlantis.

[35] Marin, J., Ragsdale, D., & Sirdu, J. (2001). A hybrid approach to the profile creation and intrusion detection. *DARPA Information Survivability Conference & Exposition II. DISCEX '01. Proceedings*, (pp. 69-76). Anaheim.

[36] Greenberg, S. (1988). *Using Unix: collected traces of 168 users.* Calgary: Research report 88/333/45, Department of Computer Science, University of Calgary.

[37] Jian, Z., Shirai, H., Takahashi, I., Kuroiwa, J., Odaka, T., & Ogura, H. (2007). A Hybrid Command Sequence Model for Anomaly Detection. *Advances In Knowledge Discovery and Data Mining , 4426* (1), 108-118.

[38] Seo, J., & Cha, S. (2007). Masquerade detection based on SVM and sequence-based user commands profile. *2nd ACM symposium on Information, computer and communications security*, (pp. 398-400). Singapore.

[39] Zhang, H., & Sheng, S. (2004). Learning weighted naive Bayes with accurate ranking. *Data Mining. ICDM '04. Fourth IEEE International Conference*, (pp. 567-570).

[40] Masrom, M., & Ismail, Z. (2008). Computer security and computer ethics awareness: A component of management information system. *Information Technology. ITSim 2008. International Symposium*, (pp. 1-7). Kuala Lumpur.

[41] Yurcik, W., & Liu, C. (2005). A first step toward detecting SSH identity theft in HPC cluster environments: discriminating masqueraders based on command behavior. *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium*, (pp. 111 - 120). Cardiff, UK.

[42] Bishop, M. (2009). Reflections on UNIX Vulnerabilities. *Computer Security Applications Conference. ACSAC '09. Annual* , (pp. 161-184). Honolulu, HI.

[43] Robbins, A. (2005). *Unix in a Nutshell* (4th ed.). (M. Loukides, Ed.) Sebastopol, United States of America: O'Reilly Media Inc.

[44] Lang, K. (1995). Newsweeder: Learning to filter netnews. *12th International Conference on Machine Learning* (pp. 331-339). San Francisco: Morgan Kaufmann Publishers.

[45] Leng, C., Wang, S., & Wang, H. (2009). Learning Naive Bayes Classifiers with Incomplete Data. *Artificial Intelligence and Computational Intelligence. AICI '09. International Conference*, (pp. 350-353). Shanghai.

[46] Brosso, I., Ferreira, F., Bressan, G., & Ruggiero, W. (2010). Known User Continuous Authentication System. *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, (pp. 1-2). Las Vegas, NV.

[47] Ferro, M., Pioggia, G., Tognetti, A., Dalle Mura, G., & De Rossi, D. (2009). Event Related Biometrics: Towards an Unobtrusive Sensing Seat System for Continuous Human Authentication. *Intelligent Systems Design and Applications. ISDA '09. Ninth International Conference on*, (pp. 679-682). Pisa.

[48] Liu, J., Yu, F., Lung, C.-H., & Tang, H. (2008). A Framework of Combining Intrusion Detection and Continuous Authentication in Mobile Ad Hoc Networks. *Communications. ICC '08. IEEE International Conference on* , (pp. 1515-1519). Beijing.

[49] Kohonen, T. (2001). *Self-Organizing Maps* (3rd ed.). New York, United States of America: Springer.

[50] Lane, T., & Brodley, C. E. (2003). An Empirical Study of Two Approaches to Sequence Learning for Anomaly Detection. *Machine Learning Vol. 51* , 73-107.

[51] Johnson, R. A. (2005). *Miller & Freund's Probability and Statistics for Engineers.* Toronto: Pearson Education, Inc.

[52] Wan, M. D., Wu, H.-C., Kuo, Y.-W., Marshall, J., & Huang, S.-H. (2008). Detecting Masqueraders Using High Frequency Commands as Signatures. *Advanced Information Networking and Applications - Workshops. AINAW 2008. 22nd International Conference*, (pp. 596-601). Okinawa.