Spring 2012

DEFEATING MASQUERADE DETECTION

Avani Kothari
San Jose State University

Recommended Citation

Kothari, Avani, "DEFEATING MASQUERADE DETECTION" (2012). *Master's Projects*. 239.
DOI: https://doi.org/10.31979/etd.hu29-kn4y
https://scholarworks.sjsu.edu/etd_projects/239

DEFEATING MASQUERADE DETECTION

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Avani Kothari

May 2012

The Designated Project Committee Approves the Project Titled

DEFEATING MASQUERADE DETECTION

by

Avani Kothari

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2012

Dr. Mark Stamp	Department of Computer Science
Dr. Chris Pollettt	Department of Computer Science
Dr. Teng Moh	Department of Computer Science

ABSTRACT

Defeating Masquerade Detection

by Avani Kothari

A masquerader is an attacker who has obtained access to a legitimate user's computer and is pretending to be that user. The masquerader's goal is to conduct an attack while remaining undetected.

Hidden Markov models (HMM) are well-known machine learning techniques that have been used successfully in a wide variety of fields, including speech recognition, malware detection, and intrusion detection systems. Previous research has shown that HMM trained on a user's UNIX commands can provide an effective means of masquerade detection. Naïve Bayes is a simple classifier based on Bayes Theorem, which relies on the command frequency. In this project we empirically test various masquerade mimicry strategies, that is, strategies for evading masquerade detection. We develop and analyze four distinct masquerade mimicry strategies and in each case, we give empirical results for their effectiveness at evading Naïve Bayes and HMM-based masquerade detection.

ACKNOWLEDGMENTS

I would like to thank Dr. Mark Stamp for all the support, help and guidance throughout the research and implementation of the project. His suggestions and guidance were very valuable at all times. This project would not have been possible without him.

I would also like to thank my committee members Dr. Teng Moh and Dr. Chris Pollett for their insightful suggestions and guidance.

Finally, I would like to express my appreciation towards Lin Huang, for letting me use his research work on Hidden Markov Models as a reference for this project.

TABLE OF CONTENTS

CHAPTER

APPENDIX

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure Page

Figure **Page**

CHAPTER 1

Introduction

A masquerader is an attacker who has obtained access to a legitimate user's computer and is pretending to be that user [26]. The masquerader's goal is to conduct an attack while remaining undetected. Detecting masquerade attacks is an important challenge in the field of security.

In spite of efforts to prevent intrusions [24], attackers with malicious intentions often find a way to attack a system. We rely on Intrusion Detection Systems (IDS) to detect such attacks. There are two broad categories of IDS, namely, signature-based and anomaly-based. Signature-based IDS relies on attack patterns or signatures. In contrast, anomaly-based detection relies on differences from expected behaviors of a user or system.

The masquerade detection strategies considered here are based on anomaly detection. Specifically, we focus on user-issued UNIX Commands. In this paper, first we consider two previously studied masquerade detection techniques; one technique relies on Hidden Markov models (HMM) and the other is known as One-Class Naïve Bayes (OCNB).

Hidden Markov models are well-known machine learning techniques that have been used successfully in a wide variety of applications, including speech recognition [15], malware detection [27], and intrusion detection [8]. Research has shown that an HMM that has been trained based on a users UNIX commands can be used as an effective means of masquerade detection [8]. One-Class Naïve Bayes is a simple and efficient technique that has also been successfully applied to the masquerade detection problem [26].

In this project, we consider several types of attacks, and we apply these attacks to a particular data setthe Schonlau dataset [19]. This dataset has been used in at least forty published papers focused on intrusion detection [15]. We empirically analyze the effect of each attack on both HMM-based and OCNB-based masquerade detection.

CHAPTER 2

Background

2.1 Intrusion Detection Techniques

Intrusion detection is an essential security feature. This is because the violation of security by an insider is the most dangerous attack in the field of security [10], as the deployed security such as the firewalls and passwords will not detect the intrusion. According to Tavallaee, Stakhanova and Ghorbani in [27] intrusion detection is defined as, a variety of techniques for detecting malicious and unauthorized activities commonly known as attacks. Intrusion detection techniques are mainly classified into two categories namely, signature-based and anomaly-based techniques. Signature-based IDS relies on pre-defined signatures, whereas, anomaly-based IDS depends on normal patterns, classifying a deviation from normal as an attack [27].

2.1.1 Signature-based Detection

In signature-based IDS, the signature refers to a byte sequence used to detect malicious activities [22]. Signature-based IDS examine the incoming data for a pattern of attack to match from the collection of attack signatures [6]. It is used to detect known attacks. One of the most important advantages of a signature-based IDS is that, no rigorous training is required that is knowledge of normal behavior is not essential. Signature-based IDS are used by various commercial products, for example malware detection and virus detection [6]. The predictability of this type of IDS is very high, and the false positive or miss rate is very low, as signature-based IDS is used to detect known attacks. Signature-based IDS are very simple and efficient in case of known signatures [24].

The biggest challenge in signature-based IDS is that every signature must have an entry in the database of attack signatures. As each incoming sequence has to be compared with each entry in the database, it slows down the system. Signature-based IDS are vulnerable in case of unknown signatures, that is, if a new attack is generated with a new signature, signature-based IDS will fail to detect the attack[22]. Signatures of newly developed attacks would have to be entered in the database in order to be detected. As we know, attacks are developed at lightening pace; hence the database of signature-based IDS needs to be updated often. We can conclude that accuracy of signature-based IDS is high with no false positives when a signature is known, but it does not perform well with unknown attacks.

2.1.2 Anomaly-based Detection

Anomaly-based IDS separates normal behavior from abnormal and attempts to define a baseline or a profile behavior [24]. So, a sufficiently large deviation from the profile behavior is evidence of intrusion and the IDS provides a warning. In anomaly-based IDS, abnormal is a synonym for an attack or intrusion [20]. An anomaly-based IDS works on the assumption that the attack behavior significantly varies from normal user behavior [13]. Therefore, it is comprised of two important steps: first, to profile a normal behavior and second, to calculate the deviation from normal behavior.

Masquerade detection is a type of anomaly-based IDS. Anomaly-based IDS refer to identifying pattern that does not fit normal behavior. Various techniques have been used to develop anomaly-based IDS, for example statistical analysis-based [20], Hidden Markov Model [8], Naïve Bayes [26] and Support Vector Machine [10] to name a few. These are a few conditions that make anomaly-based IDS difficult to detect:

1. Defining normal behavior that includes all possible normal instances

2. Malicious actions adapting themselves to normal behavior to avoid getting detected

3. Adapting to evolving normal behavior

4. Obtaining sufficient training data so as to define normal behavior [2]

Anomaly-based IDS can detect a previously unknown attack. We can say anomaly-based IDS can thus, over come the disadvantage of signature-based IDS.

2.2 Masquerade Detection

A masquerade attack is an attack by a malicious user who has gained access to a legitimate users credential and tries to impersonate the legitimate user. An IDS used to detect this masquerade attack is called masquerade detection. We focus on a UNIX command domain dataset in this project. Substantial work has been done in this domain in the past few years. For masquerade detection in UNIX command domain, various techniques have been developed to detect masquerade attacks. A survey was conducted by Beracchini and Fierens in [1], to analyze all the techniques used for masquerade detection in this domain. Below are several of the techniques mentioned in [1].

1. Information Theoretic: This is a simple compression technique based on the assumption that, commands by user will tend to compress more than the intruders commands. The results of this technique were disappointing [20].

2. Text Mining: This is a very popular technique. It detects repetitive sequence by the user. The results are very accurate, but the computational cost is high [3].

3. Hidden Markov Model Hidden Markov Model (HMM) is a state machine [32] based machine learning technique. HMM is trained using a legitimate users data. The probability of transition from one state to another is provided. The results are very good in case of HMM [8].

4. Naïve Bayes Naïve Bayes is a very simple and efficient technique using a probability classifier. But in the UNIX command domain, the masquerade detection could not perform substantially well [9]. As a result, a modified version known as One-Class Naïve Bayes (OCNB), was developed in [26] to obtain better results. The results obtained in [26] are exceptionally good.

5. Sequence and Bioinformatics Techniques based on sequence, generally used in bioinformatics to identify a pattern, were applied in the UNIX command domain. The entire focus here is on sequence related information. The results were unexpectedly inaccurate as compared to other masquerade detection techniques [20].

6. Support Vector Machine Support Vector Machine (SVM) is a well-known machine learning technique that tends to separate data on separate planes. More emphasis is given to efficiency as compared to Naïve Bayes. The results in [31] show that SVM outperforms the Naïve Bayes approach.

7. Other Approaches A combination of two or more of the above methods was used to develop several other hybrid approaches, but the results were unimpressive [1].

We explain two of the most accurate masquerade detection techniques, Hidden Markov Model and One-Class Naïve Bayes in Sections 3 and 4, respectively.

In the foloowing section, we first explain the standard dataset used in masquerade detection called the Schonlau Dataset [19]. We then explain in detail the basis to judge a detection technique. Performance criteria are needed to compare effectiveness, accuracy and correctness of masquerade detection techniques.

2.2.1 Schonlau Dataset

The below Figure 1, is the pictorial view of Schonlau dataset [8].

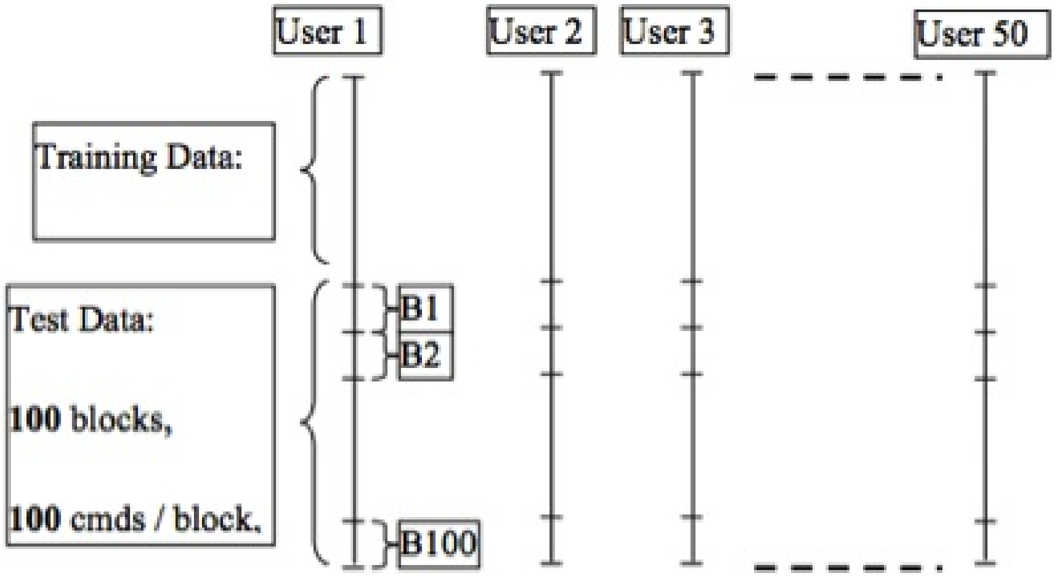


Figure 1: Pictorial view of Schonlau Dataset [8]

Schonlau collected data of 50 users over a period of time. The Schonlau dataset consists of 15000 UNIX commands for each user. Each block consists of 100 commands. First 5000 commands (50 blocks) of every user are assigned for training purposes and the remaining 10000 commands (100 blocks) are assigned for testing. Figure 2 below, is the pictorial view of the summary file.

Schonlau dataset also includes a map file. This map file consists of 0s and 1s assigned for each block for the 50 users. This map file contains the summary of the

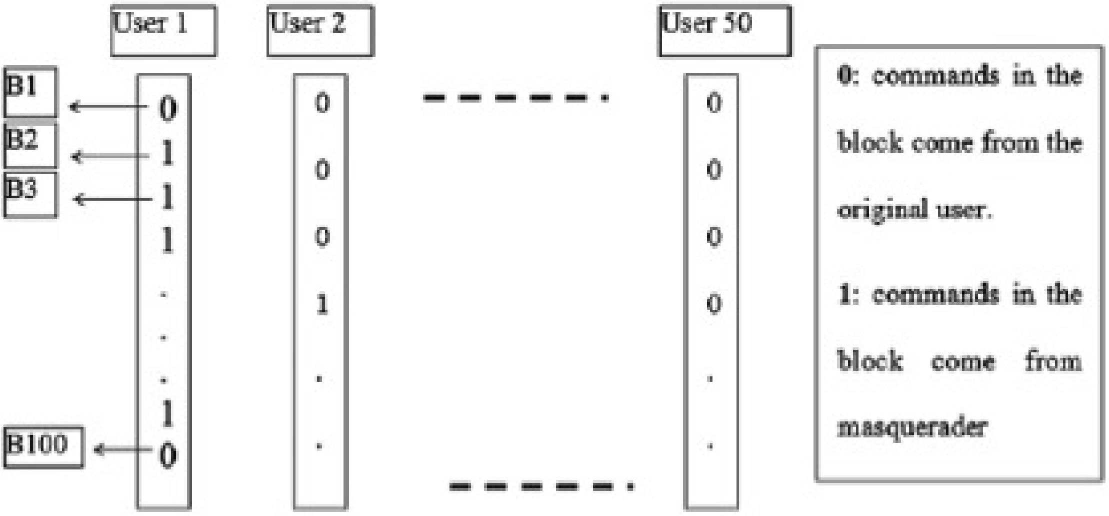

User 1
User 2
User 50
B1
B2
B3
B100
0
1
1
1
1
0
0
0
0
1
0
0
0
0
0: commands in the block come from the original user.
1: commands in the block come from masquerader

Figure 2: Masquerade locations [8]

attack. It consists of a 50 * 100 matrix where, 100 rows represent the 100 testing blocks for each of the 50 users. 0 in the summary file indicates that the data of the block belongs to the same user and it may not be contaminated by the attack commands. A 1 in the summary file indicates the block is contaminated [19].

2.2.2 Performance Criteria

In anomaly-based IDS, first we train a model and set the threshold. Setting this threshold is an essential process, as this threshold is eventually used to determine, if the testing data is normal or anomalous. There are several terms and formula that we will use in this paper, namely detection rate, false positive rate, and Receiver Operating Characteristics (ROC) curve. These terms together comprise performance criteria. We use these terms to compare different masquerade detection techniques. We first define all possible outcomes while performing masquerade detection below in Figure 3.

A false positive (FP) occurs when; an IDS misclassifies non-attack data to be

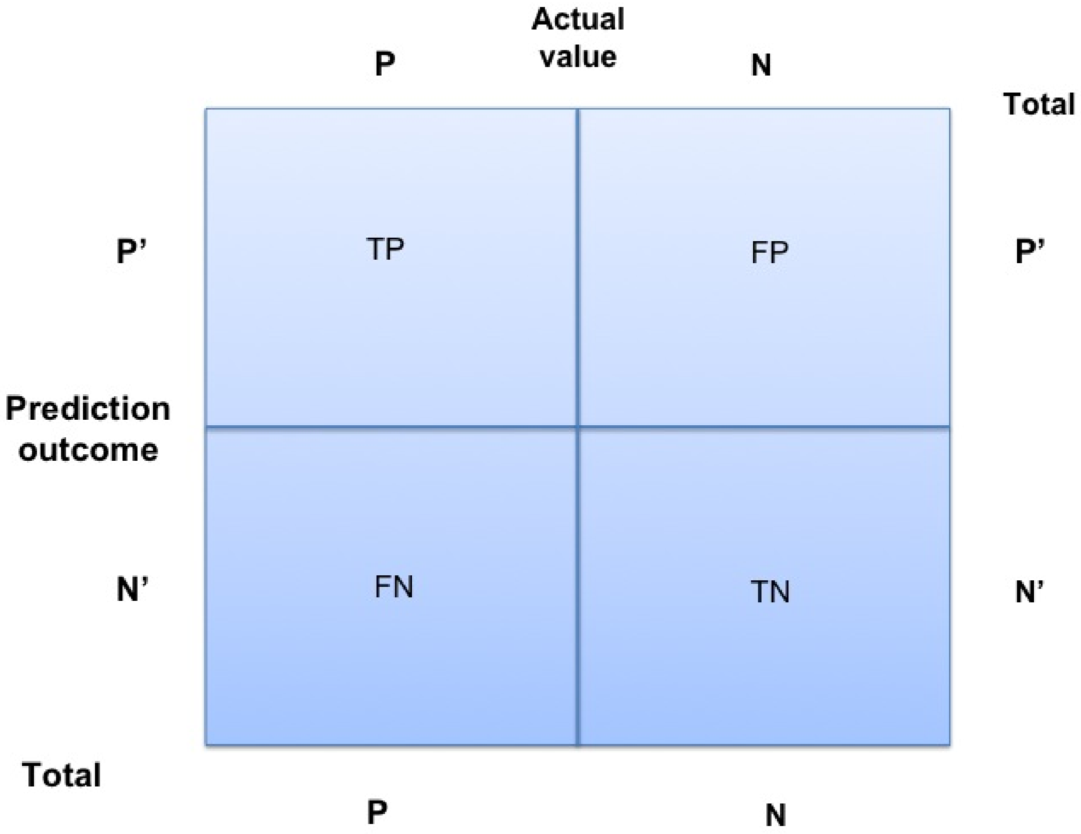


Figure 3: The possible outcome diagram [16]

attack data. A true negative (TN) occurs when the IDS predicts non-attack data to non-attack data. Similarly a true positive (TP) occurs when the IDS correctly classifies an attack data to be attack data. And lastly, a false negative (FN) occurs when the IDS classifies attack data to not attack data. To determine effectiveness of a masquerade detection technique we calculate detection rate and false positive rate for the technique. The ideal masquerade detector should be able to detect all the true positives and should not have any false positives. To compare two masquerade detectors, their effectiveness is measured by comparing their detection rate and false positive rate.

False negative rate (FNR) and false positive rate (FPR) are used to plot the ROC curve used to set the threshold. The formula of the TPR and FPR are as follows,

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + FN} \tag{2}$$

ROC curves are visual comparison between different masquerade detection techniques applied on the same dataset [1]. The co-ordinates to plot ROC curves are obtained by varying the thresholds, obtaining false positive and false negative at each of these threshold values. ROC curves are used to display trade-off between missing alarm rate (FNR) and false alarm rate (FPR). We can then measure Area Under the Curve (AUC). Lesser the AUC better the efficiency of the technique.

If the threshold is set too high, true positive rate increases with increase in false positive rate. On the contrary if the threshold is set too low, the false positive rate decreases, but even the true positive rate decreases. So, we plot a graph between false positive rate (FPR) and false negative rate (FNR) [21]. From the graph we select an acceptable false positive rate. This type of graph is called an ROC curve and hence we say, ROC curves are useful in setting an ideal threshold [8]. In this paper we consider false positive rate of 0.05 as acceptable rate. So, in ROC curve we consider the area before FPR 0.05 as useful. By plotting ROC curves for various masquerade detection techniques applied on same dataset, we can compare them at FPR 0.05 for better efficiency.

To empirically analyze and compare the performances of an intrusion detection technique, their accuracy rates (ACR) are considered. Accuracy rate defines the overall accuracy of the IDS. It can be defined as below,

$$ACR = \frac{TP + TN}{P + N} \tag{3}$$

Detection rate is also known as true positive rate (TPR). Detection rate of an IDS defines its correctness.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{4}$$

Hence we calculate detection rate, accuracy rate and ROC curve for different masquerade detection techniques on the same dataset while comparing and analyzing performances of the techniques considered in this report.

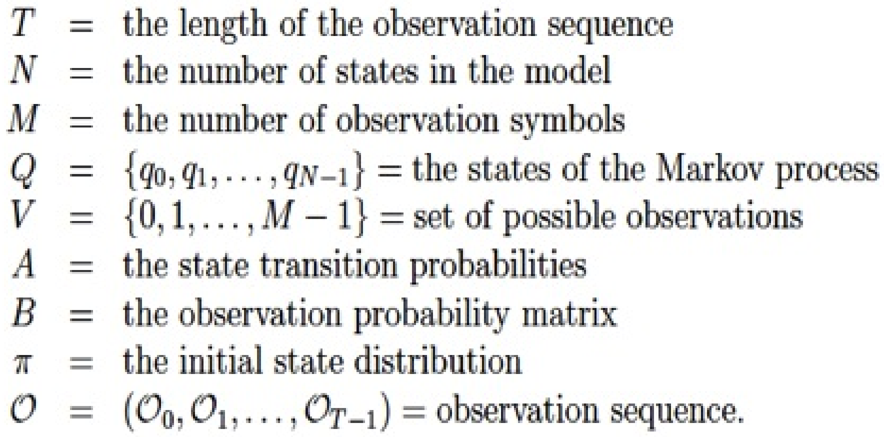
CHAPTER 3

Hidden Markov Model

The Hidden Markov Model (HMM) is a machine-learning technique based on Markov Chains. HMM is widely used in applications like speech recognition, virus detection and malware detection. We use HMM to perform masquerade detection to test its ability to detect the number of attacks successfully blocked. In this report we compare HMM with a detection technique based on Bayes Theorem.

3.1 Introduction to HMM

Markov Chain is a stateless and memory less process. The transition from one state to another state depends only on current state and transition probability matrix [25]. HMM is a finite state model governed by a set of transition probabilities. With each hidden state two probabilities are associated, one provides the probability of transition to another state and second provides probability of being in the same state [5]. Below in Figure 4, are few of the notations we use in HMM.



T = the length of the observation sequence
N = the number of states in the model
M = the number of observation symbols
Q = $\{q_0, q_1, \ldots, q_{N-1}\}$ = the states of the Markov process
V = $\{0, 1, \ldots, M-1\}$ = set of possible observations
A = the state transition probabilities
B = the observation probability matrix
π = the initial state distribution
$\mathcal{O}$ = $(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$ = observation sequence.

Figure 4: Notations for HMM [25]

In masquerade detection, the training data, i.e. the first 5000 commands of the user is used to determine model of normalcy. The possible states would be self-user block and masquerade block. The observation sequence would be the testing block that we want to classify. HMM is defined by A, B and π. HMM model is represented by λ = (A, B, π) as in [25].

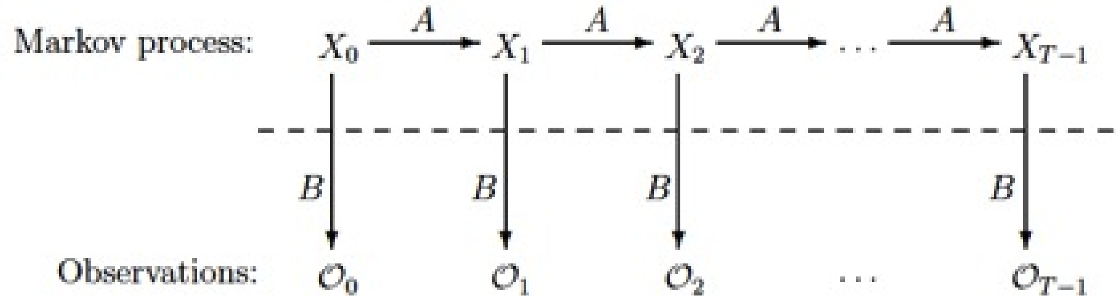


Figure 5: Hidden Markov Process [25]

The Markov Chain is hidden behind the dashed line in the Figure 5 above. The Markov process is determined by the A matrix and the current state. The observations are determined by the Markov process related to the hidden states with the help of matrix B.

3.2 Implementation

Below are the three problems, which can HMM solve efficiently.

Problem 1: Given a model λ = (A, B, π), HMM can obtain scores for the observation sequences to determine their similarity to the model by calculating $P(O \mid \lambda)$. This determines how well the observation fit the model [25].

Problem 2: Given a model λ = (A, B, π) and Observation O, HMM can determine the hidden optimal sequence [25].

Problem 3: Given O, N, and M, HMM can train a model to fit the observation by determining the model λ so as to obtain maximum probability of O [25].

In the masquerade detection domain, we are concerned about problems 1 and 3. We first train the model with the training data that is first 5000 commands. We do this using the problem 3 above. After training the model, we then use problem 1 to obtain scores for the testing blocks $P(O \mid \lambda)$, to determine their similarity with the model. A high probability score indicates greater resemblance between the test block and the training block and would be classified as a self-user block. A low probability block would indicate differences between training data and testing block and this block would be classified as an attack block.

CHAPTER 4

One-Class Naïve Bayes

4.1 Introduction to Naïve Bayes

One-Class Naïve Bayes (OCNB) in the domain of masquerade detection is based on Bayes Theorem. Naïve Bayes (NB) classifiers are a supervised learning technique. They are very simple to implement and understand and more often than not give efficient results [7]. Bayes theorem gives a rule for conditional probability. Conditional probability is probability of event A occurring if event B occurs. In masquerade detection we use Naïve Bayes classification to determine the probability of a block of 100 commands belonging to a user. For example, to estimate that an instance of $x = \{x_1, x_2, x_3, \ldots, x_n\}$ belongs to class y as,

$$P(y \mid x) = \frac{P(y)}{P(x)} P(x \mid y) = \frac{P(y)}{P(x)} \prod_{i=1}^{m} P(x_i \mid y) \tag{5}$$

In the next section we discuss in detail how Naïve Bayes Classification helps in masquerade detection in UNIX commands domain.

4.2 Implementation

NB has been used in context of masquerade detection in [26] mainly using Schonlau Dataset. Schonlau Dataset consists of 15,000 commands (150 blocks) for each user out of which first 5000 (50 blocks) commands are used for training and the remaining 10000 (100 blocks) commands are used for testing.

Let $[c_1, c_2, \ldots, c_m]$ where c_i is the $(i)^{th}$ unique command in the block B. This vector consists of names of all the commands appearing in the block B at least once and m is number of unique commands in block B. Every block B to be classified is then represented into a vector of attribute $[n_1(B), n_2(B), \ldots, n_m(B)]$ where $n_i(B)$ is number of times command c_i appears in the block B. Then we can compute $P(y \mid B)$ as follows [26],

$$P(y \mid B) = P(y) \prod_{i=1}^{m} P(c_i \mid y)^{n_i(B)} \tag{6}$$

The probabilities of $P(c_i \mid y)$ for every command can be calculated from the training data (first 5000 commands). To normalize the probability on occurrence of new commands and to obtain non-zero probability we add the smoothing factor while calculating $P(c_i \mid y)$. P(y) and P(x) can be ignored as they are constants and will remain the same for all the commands. The final $P(c_i \mid y)$ [26] can be given by,

$$P(c_i \mid y) = \frac{\sum\limits_{B \in T(y)} n_i(B) + \alpha}{\mid B \mid \cdot \mid T(y) \mid + \alpha \cdot m} \tag{7}$$

Where, α is the smoothing factor and T(y) is the training set for class y. We calculate the score for each block to classify the block as a user block or a non-user block. The score for every block is calculated as follows,

$$score(B) = -\log P(y \mid B) = -\sum_{i=1}^{m} n_i(B) \log P(c_i \mid y) \tag{8}$$

The anomaly of the block is directly proportional to the score [26]. If a block has very less score it has probability to be a users block. We set the threshold to be a score, which gives us the acceptable false positive rate and detection rate.

CHAPTER 5

Evading Masquerade Detection

For any masquerade detection technique, both efficiency and accuracy are its most important features. In this project we try to attack two of the best-known masquerade detection techniques called HMM and OCNB to test their strengths and determine their weaknesses. In this process, we have developed four unique attacks namely,

1. Scattered Attack
2. Scattered Sorted Attack
3. Consecutive Attack
4. Consecutive Random User Attack

These attacks are generated on a block of 100 commands. In Scattered Attack, the attacks commands are scattered randomly throughout the block. Whereas, for Scattered Sorted Attack, the attack blocks are scattered but in a sorted manner. In Consecutive Attack, the attack commands are placed consecutively. But in case of Consecutive Random User Attack, the attack commands are obtained from random users and then placed in a consecutive manner. We explain each of these attacks in this section and then empirically analyze compare their results.

All the four attacks are randomized and non-poisoned. Below are a few notations and basic concepts that we will be using while generating the attacks.

A = attack

P = padding

| | = Denotes length of the sequence

B = block in which attack is being generated

| B | be the block length = 100

So, we have

$$| B |=| A | + | P | \tag{9}$$

In a block of 100 commands, we if we have |A| attack commands, we have 100-|A| as padding commands.

While generating the attack we always maintain |A| ≤ |B|. These three unique attacks differ in positions and attack commands. Below we explain each of these attacks in detail.

5.1 Scattered Attack

5.1.1 Discussion

We generate this attack for |A| from 10, 20, 30,. . .,100 for each of the 50 Users. This attack is called a scattered attack because the attack commands are scattered throughout the block. We select random |A| positions in the block to insert the attack. The attack commands are chosen randomly from the random users training data. We use training data because we do not intend to generate a poisonous attack. Below is the algorithm to generate Scattered Attack.

Algorithm:

For every user:

1. *Obtain random $|A|$ numbers between 1 to 100*

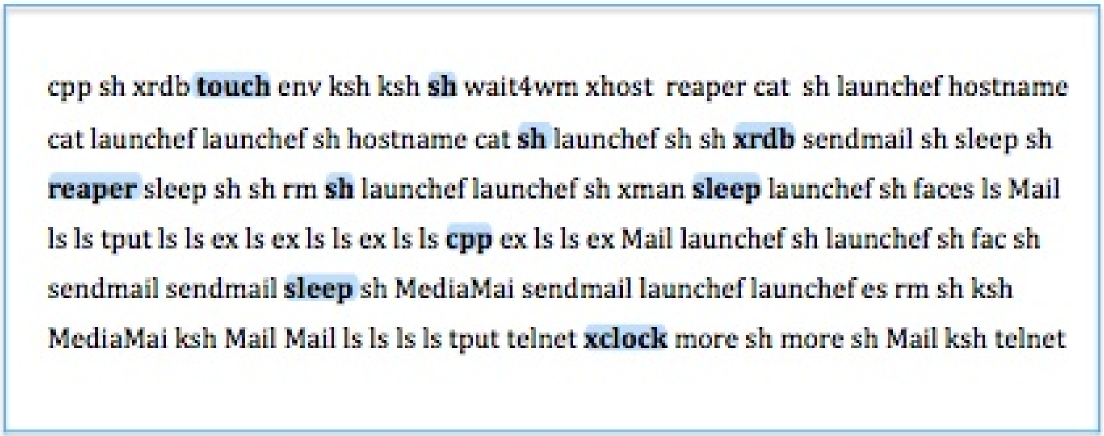
2. *Get a random user except the current user say randUser*

3. *Get |A| commands from a random starting point in randUser'ss training data*

4. *Insert these |A| commands from randUser in the random positions generated in step 1*

5. *Pad the rest of the block with self user commands*

In the above algorithm, |A| is the attack length. Below Figure 6, is the example of Scattered Attack:



cpp sh xrdb **touch** env ksh ksh **sh** wait4wm xhost reaper cat sh launchef hostname
cat launchef launchef sh hostname cat **sh** launchef sh sh **xrdb** sendmail sh sleep sh
reaper sleep sh sh rm **sh** launchef launchef sh xman **sleep** launchef sh faces ls Mail
ls ls tput ls ls ex ls ex ls ls ex ls ls **cpp** ex ls ls ex Mail launchef sh launchef sh fac sh
sendmail sendmail **sleep** sh MediaMai sendmail launchef launchef es rm sh ksh
MediaMai ksh Mail Mail ls ls ls ls tput telnet **xclock** more sh more sh Mail ksh telnet

Figure 6: Example of Scattered Attack of |A| = 10

The above block in Figure 6 is an example of block of a 100 commands. The 10 highlighted commands are the attack commands inserted into the block. These attack commands are obtained from training data of some randomly selected user. We then generate a summary file to keep a track of the blocks in which attack is inserted. We then test the attack with both of the masquerade detection techniques based on HMM and OCNB.

5.1.2 Experimental Results

The attack generated by the above algorithm is tested with HMM and OCNB. The results of these detection techniques are explained below in Figure 7.

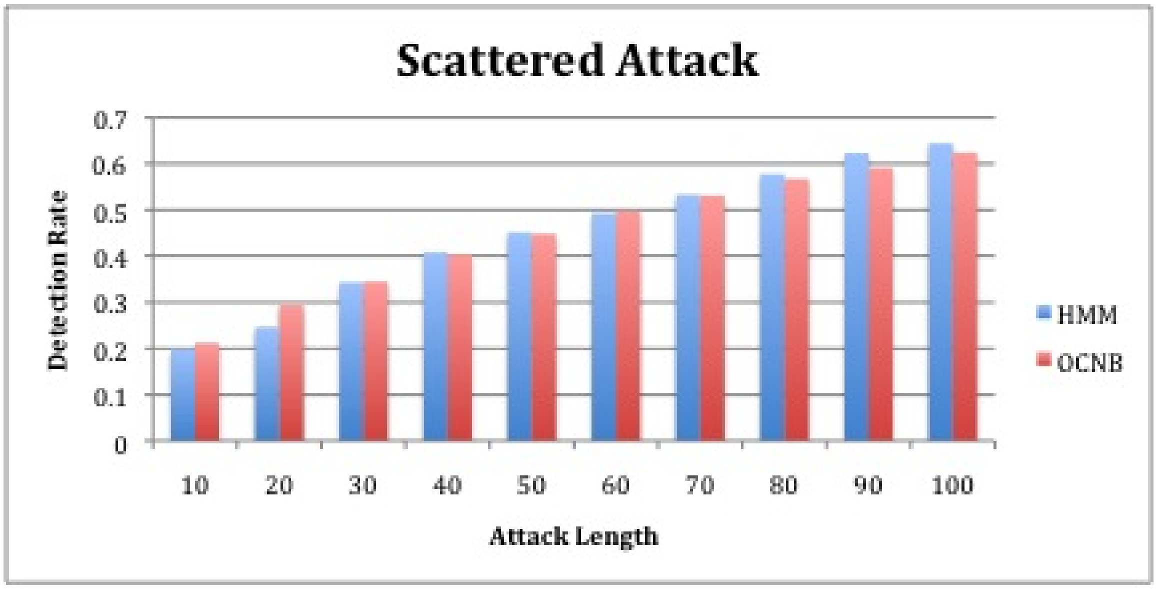


Figure 7: Detection Rate of Scattered Attack for HMM and OCNB

The graphs in Figure 7, is the detection rate obtained for one of the User when tested for Scattered Attack with both HMM and OCNB. In Figure 7 we can clearly see that up to length 70, both HMM and OCNB have almost the same detection rate, but as the attack length increases further, the detection rate of HMM is more than OCNB. So we can say that, HMM can detect Scattered Attack very efficiently beyond attack length of 70.

In the Figures 8, 9, 10 and 11 below, we show examples of Scattered Attack with attack length 10 and 50, tested with HMM and OCNB for one of the User. As we see in these figures, scores of each of the 100 blocks in the testing data are plotted. The self-user blocks are in blue color, whereas the attack blocks are red in color. The threshold differentiates the attack data from self-user data. In case of

HMM, the scores are negative so the attack data area would be below the threshold line. Whereas, in OCNB, the scores are positive, so the attack blocks are above the threshold line. In these examples the false positive rate is tuned to 0.05.

For this example, the detection rate for attack length 10 with HMM is 0.1. We can see in the Figure 8, the attack blocks are camouflaged in the self-user area above the threshold line. In Figure 9, the detection rate for attack length 50 with HMM is 0.7; we can observe that most of the attack blocks are correctly classified. In Figure 10, the scores for all the testing blocks of scattered attack of length 10 tested with OCNB are shown. We can clearly observe that most of attack blocks have similar scores are self-user blocks, thus they lie below the threshold line. In fig 11, we analyze the Scattered Attack of length 50 tested with OCNB. The detection rate of this example was found to be 0.7.

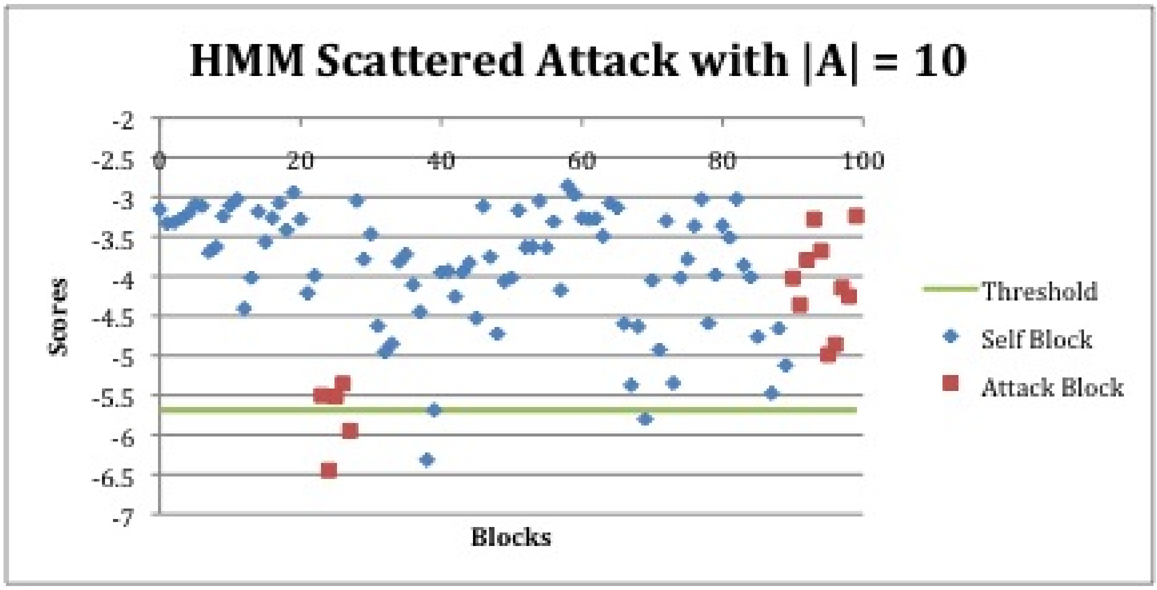


Figure 8: Scores by HMM for Scattered Attack of length 10

We then plot ROC curves by generating false positives and false negative for each of the masquerade detection technique. ROC curve is a visual comparison between

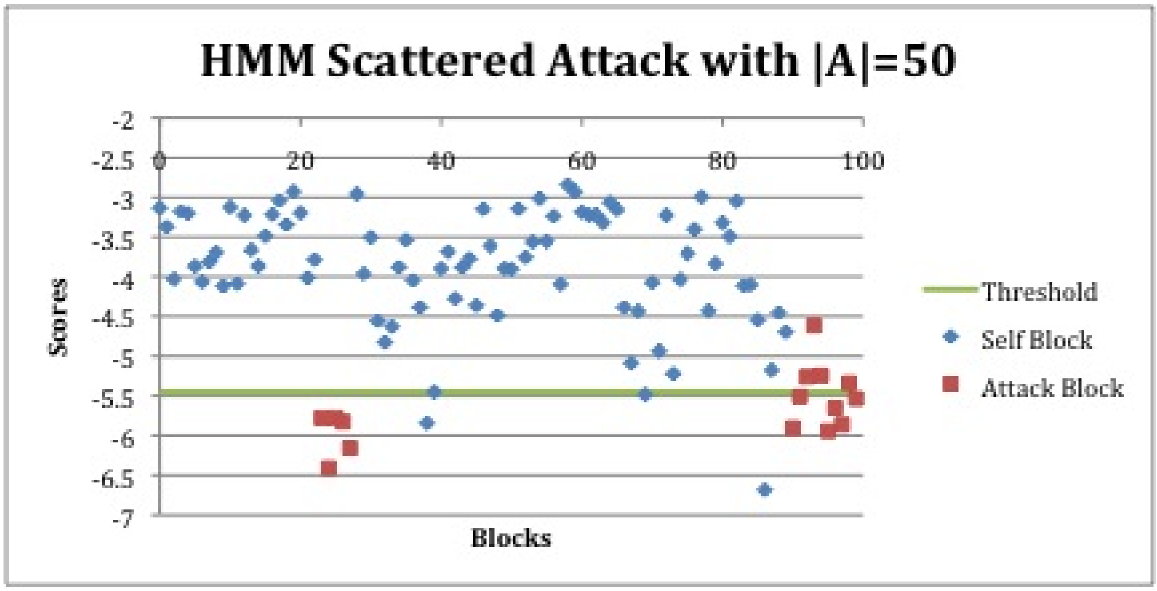


Figure 9: Scores by HMM for Scattered Attack of length 50

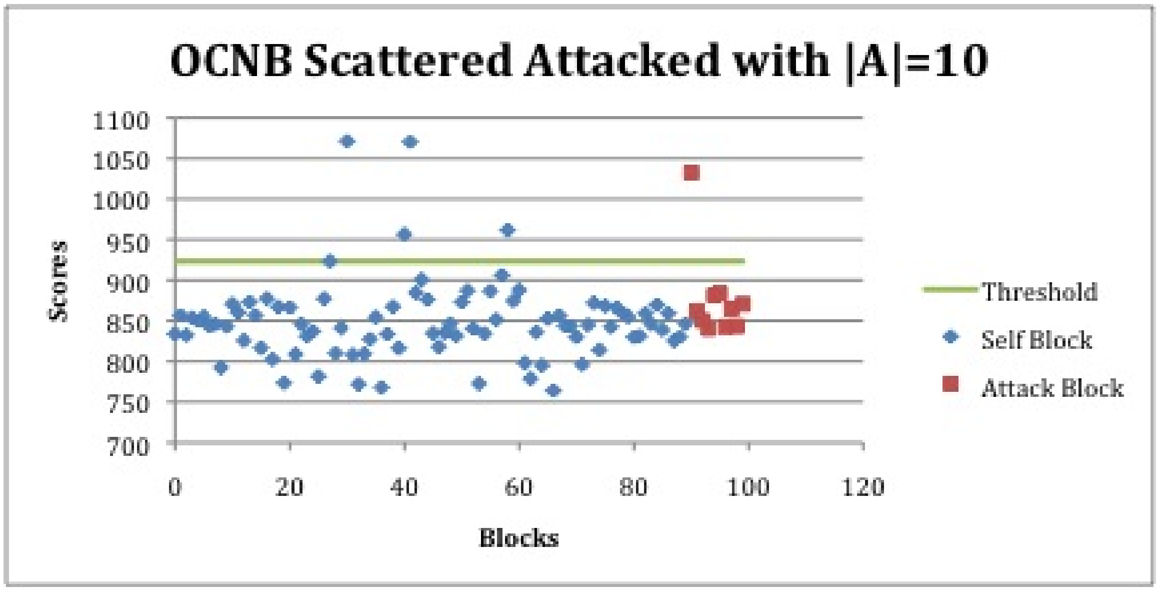


Figure 10: Scores by OCNB for Scattered Attack of length 10

HMM and OCNB. We will consider a technique to be better than other, if the threshold cuts the curve at lower rate of false negative. In Figure 12 we can clearly see the for attack length 10, HMMs performance is better than OCNB. Whereas in Figure 13, we compare an example with attack length 50, we can see that at the threshold

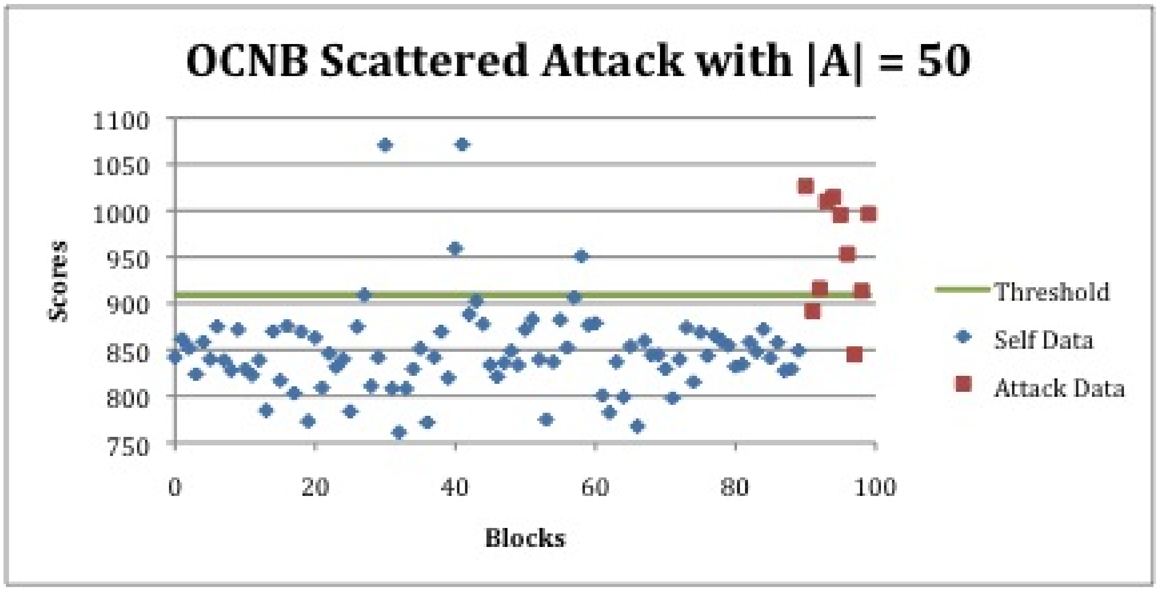


Figure 11: Scores by OCNB for Scattered Attack of length 50

mark, both HMM and OCNB intersect with threshold at same point thus we can say both perform equally for SA, |A| = 50.

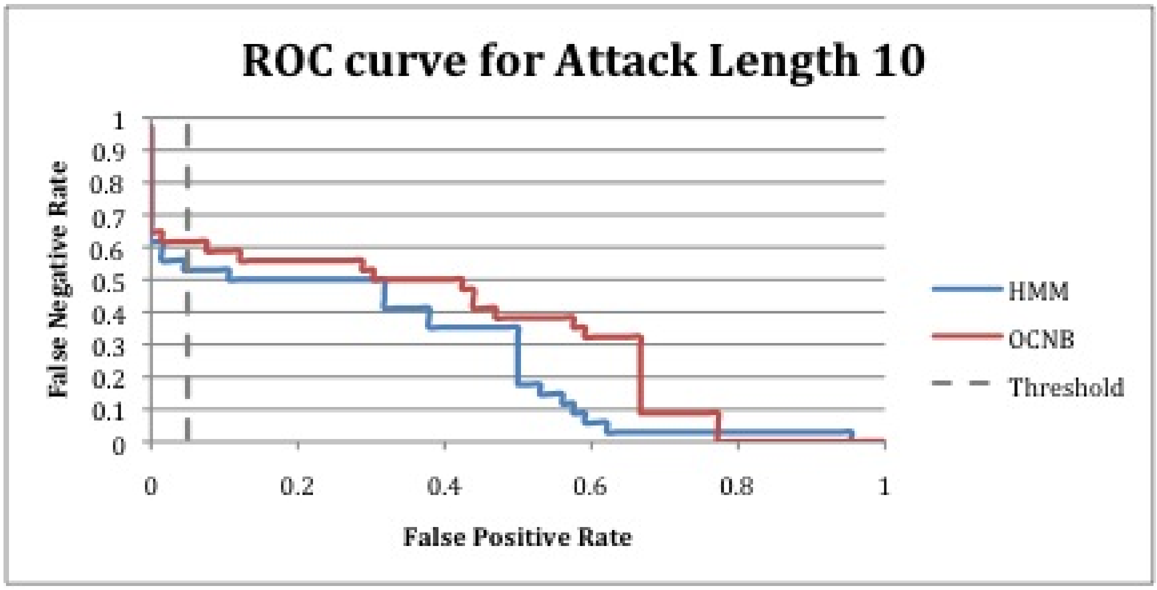


Figure 12: ROC curve for SA, |A| = 10

In Figure 14 we can observe the ROC curves plotted, we detect that, OCNB is more efficient than HMM when |A| = 100. In Figure 15 we compare ROC curves for

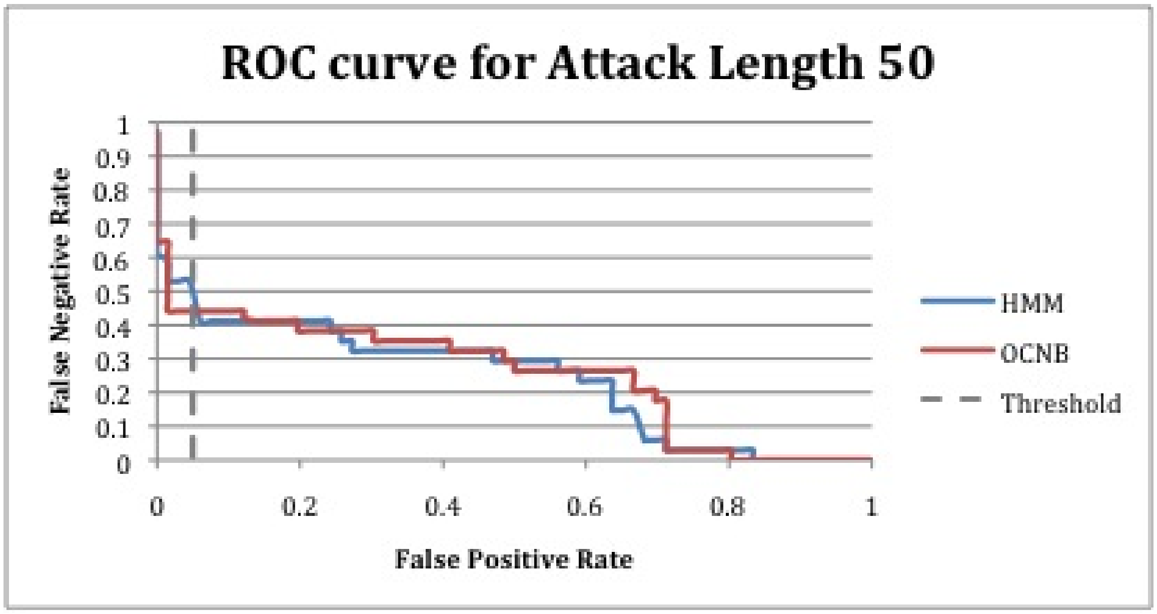


Figure 13: ROC curve for SA, |A| = 50

SA of |A| = 10, 50 and 100 with both HMM and OCNB. We conclude that efficiency of both the masquerade detection technique increases with increase in |A|, as the area under the curve gradually decreases.

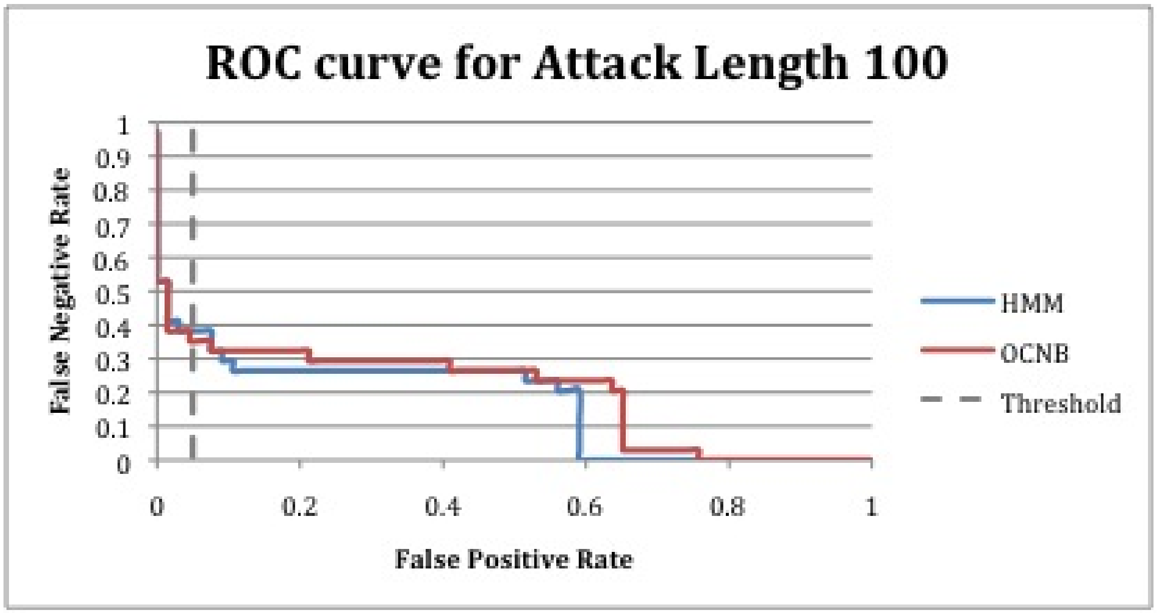


Figure 14: ROC curve for SA, |A| = 100

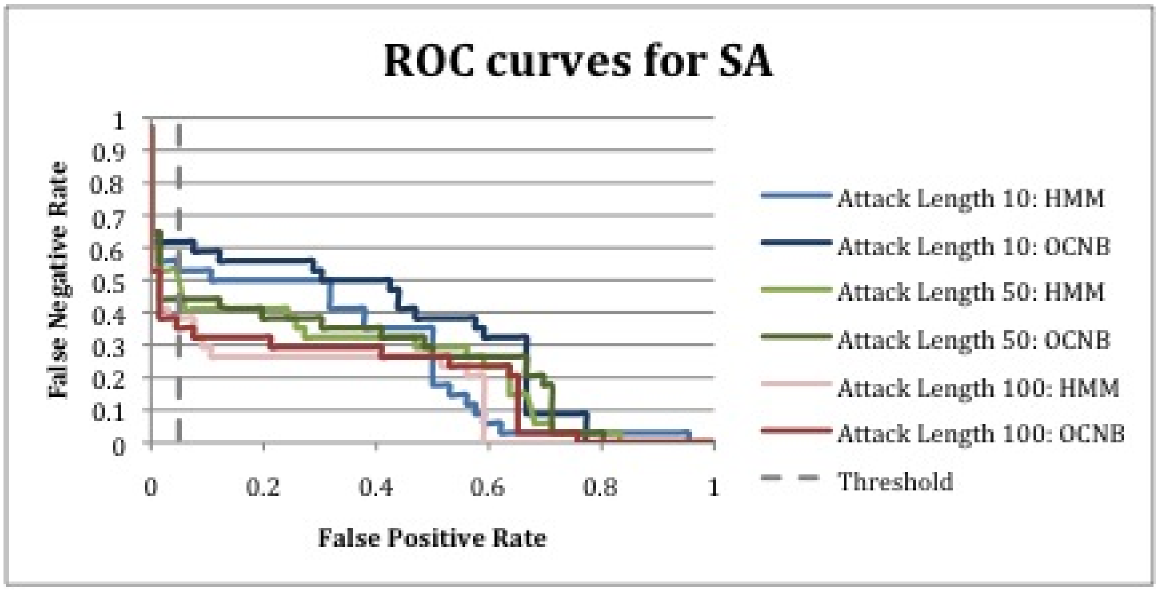


Figure 15: ROC curve for SA

We observe, for the considered example of SA, the detection rate of OCNB is better for attack lengths less than 20, but beyond attack length 20 HMM performs better than OCNB.

5.2 Scattered Sorted Attack

5.2.1 Discussion

Scattered Sorted Attack (SSA) is a similar attack to Scattered Attack but with a modification. In SSA, the attack commands are inserted in a random and scattered manner, but in ascending order of position. That is, we first randomly select the positions and sort them, and then insert the attack commands. We perform this attack for 50 Users and attack length varying from 10, 20, 30,. . .,100.

In SSA, we first randomly select |A| positions then we then sort these positions. After sorting the positions we insert attack commands in these positions.

Algorithm:

For every user:

1. *Obtain random $|A|$ numbers between 1 to 100*
2. *Sort these $|A|$ positions in ascending order*
3. *Get a random user except the current user say randUser*
4. *Get $|A|$ commands from a random starting point in randUser'ss training data*
5. *Insert these $|A|$ commands from randUser in the random positions generated in step 1*
6. *Pad the rest of the block with self user commands*

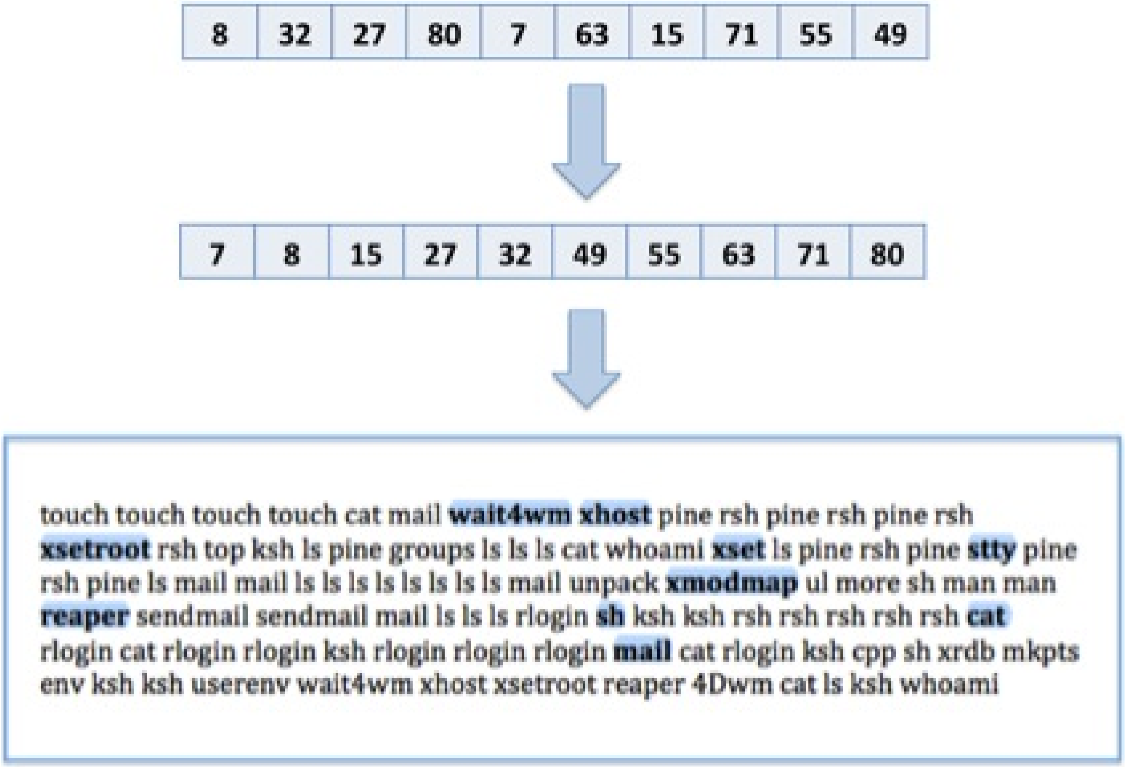
Figure 16 below is an example of Scattered Sorted Attack.

In the Figure 16, we can observe, random positions are selected in a block of 100 commands, these positions are in random order. The positions are then sorted; the attack commands are then inserted at these positions. The intuition behind this attack is that when the attack length is 100, the attack will be in consecutive manner.

5.2.2 Experimental Results

After generating the attack, we test it with both HMM and OCNB. In Figure 17, the detection rate for both HMM and OCNB is plotted for a user. We can clearly observe OCNB performs better when attack is less, but as attack length increases the detection rate for SSA is more in case of OCNB. HMM performs better for attack lengths 30, 40, 50 and 70 for this example.

We then plot ROC curves to determine better masquerade detection technique. The ROC curve in Figure 18, is for attack length 10, we can say that OCNB performs



8 32 27 80 7 63 15 71 55 49

7 8 15 27 32 49 55 63 71 80

touch touch touch touch cat mail **wait4wm** **xhost** pine rsh pine rsh pine rsh **xsetroot** rsh top ksh ls pine groups ls ls ls cat whoami **xset** ls pine rsh pine **stty** pine rsh pine ls mail mail ls ls ls ls ls ls ls mail unpack **xmodmap** ul more sh man man **reaper** sendmail sendmail mail ls ls ls rlogin **sh** ksh ksh rsh rsh rsh rsh rsh **cat** rlogin cat rlogin rlogin ksh rlogin rlogin rlogin **mail** cat rlogin ksh cpp sh xrdb mkpts env ksh ksh userenv wait4wm xhost xsetroot reaper 4Dwm cat ls ksh whoami

Figure 16: Example of Scattered Sorted Attack of |A| = 10

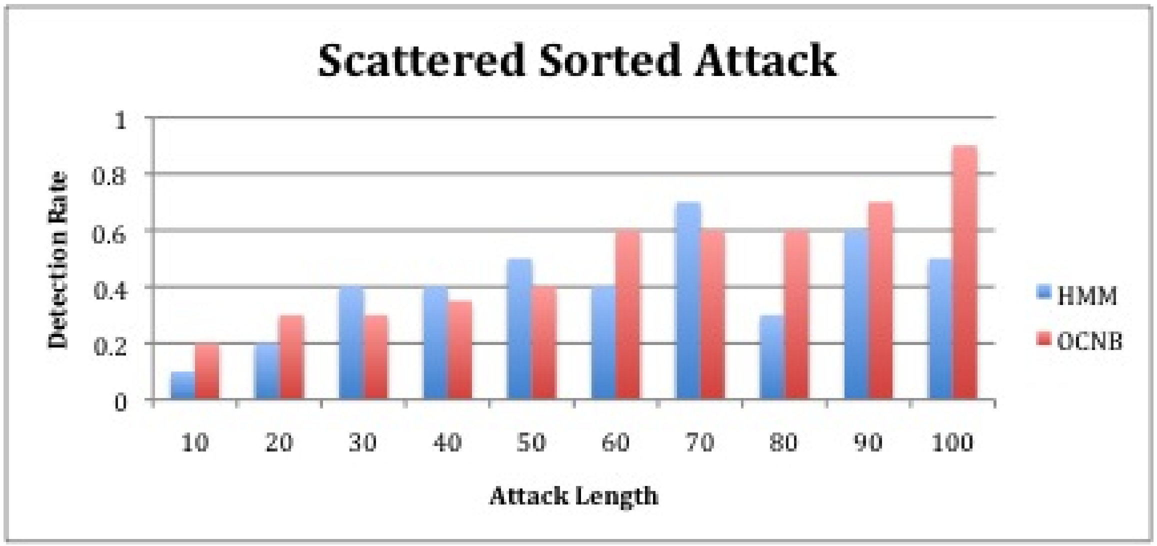


Figure 17: Detection Rate of SSA with HMM and OCNB

better. We also observe ROC curves for attack length 50 and 100 respectively in Figures 19 and 20. By examining these curves, we can conclude that OCNB performs better than HMM for this example of SSA.

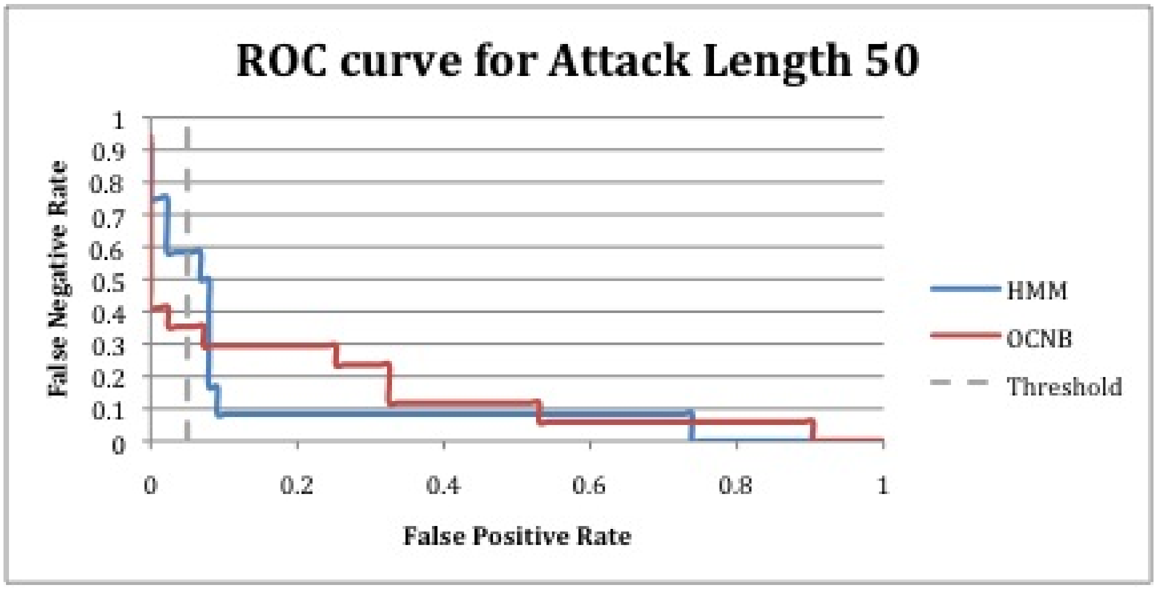


Figure 18: ROC curve for SSA, |A| = 10

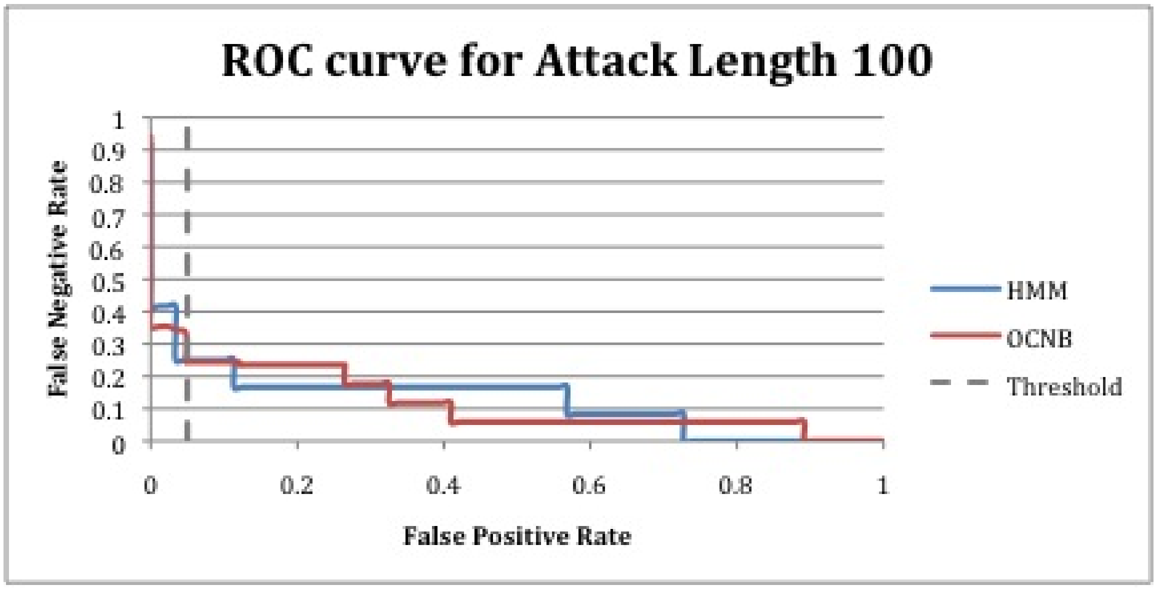


Figure 19: ROC curve for SSA, |A| = 50

It can be clearly observed that the threshold intersects the ROC curve of OCNB

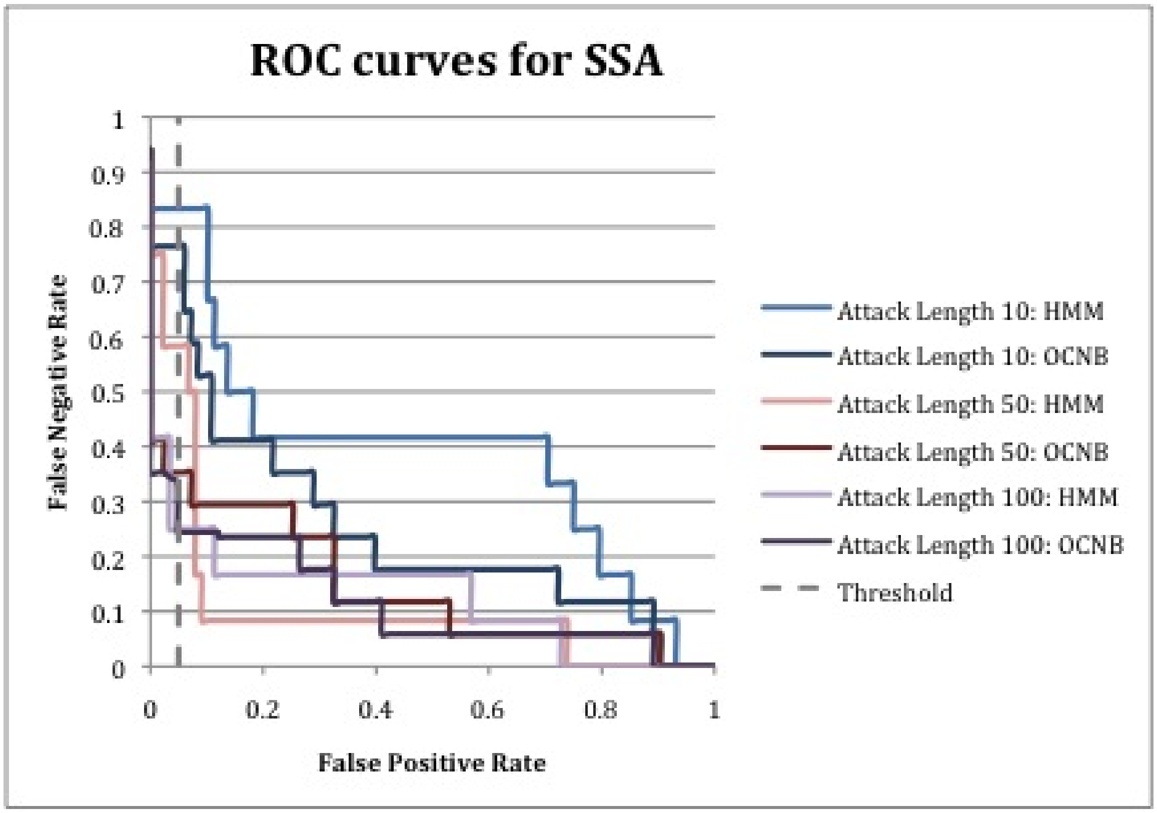


Figure 20: ROC curve for SSA, |A| = 100

at lower false negative rate in all the above cases and also the area covered by OCNBs ROC curve is less than that of HMM. Thus we can say for this example OCNB performs better than HMM.

5.3 Consecutive Attack

5.3.1 Discussion

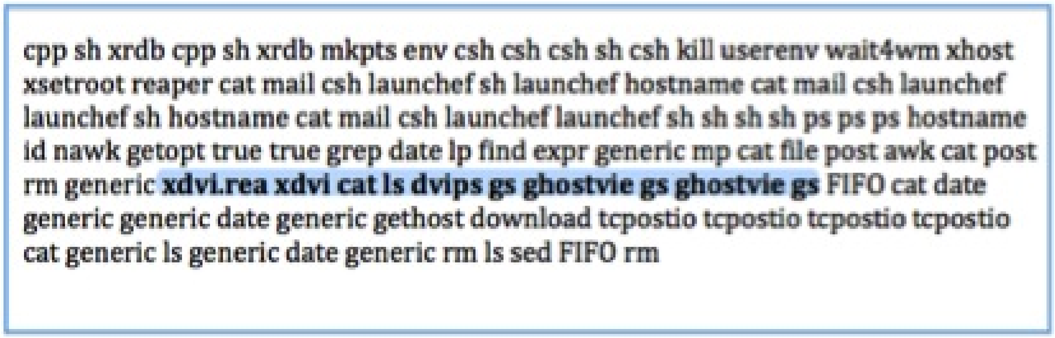
Consecutive Attack is also a random attack. In Consecutive Attack, the attack commands are inserted in a consecutive order in the attack block. The attack commands are obtained from a randomly selected users training data. After inserting the attack commands, the block is padded with self-user commands. We generate this attack for all the 50 users with attack lengths 10, 20, 30,. . .,100. Below is the algorithm for Consecutive Attack.

Algorithm:

For every user:

1. *Get a random starting point in the block say strpt such that (strpt +* $|A|$*)* $\leq$ *100*
2. *Obtain a random User say randUser from 1 to 50 except current user and a random point between 1 to (5000-*$|A|$*) say rdpt*
3. *Get consecutive* $|A|$ *commands from randUser starting from rdpt*
4. *Insert these* $|A|$ *commands from strpt in consecutive manner*
5. *Pad the rest of the block with self-commands.*

Figure 22 below, is an example of Consecutive Attack of attack length (|A|) is 10. The highlighted commands are the attack commands and the non-highlighted ones are the padding commands.



```
cpp sh xrdb cpp sh xrdb mkpts env csh csh csh sh csh kill userenv wait4wm xhost
xsetroot reaper cat mail csh launchef sh launchef hostname cat mail csh launchef
launchef sh hostname cat mail csh launchef launchef sh sh sh sh ps ps ps hostname
id nawk getopt true true grep date lp find expr generic mp cat file post awk cat post
rm generic xdvi.rea xdvi cat ls dvips gs ghostvie gs ghostvie gs FIFO cat date
generic generic date generic gethost download tcpostio tcpostio tcpostio tcpostio
cat generic ls generic date generic rm ls sed FIFO rm
```

Figure 21: Example of Consecutive Attack of |A| = 10

Consecutive Attack is then tested with both HMM and OCNB. The results are given in the next section.

5.3.2 Experimental Results

Below is an example of CA, where it has been tested with HMM and OCNB. In Figure 23, the bar graphs represent detection rate. For attack length 10 to 60 we can clearly observe that OCNB, has a better detection rate than HMM. But, for this example we can see that the detection rate of attack length 70 and 80 are slightly better for HMM. Again for attack length 90 and 100, the detection rate for OCNB is better. More examples of scores of each block of Consecutive Attacks with both HMM and OCNB are placed in the Appendix.

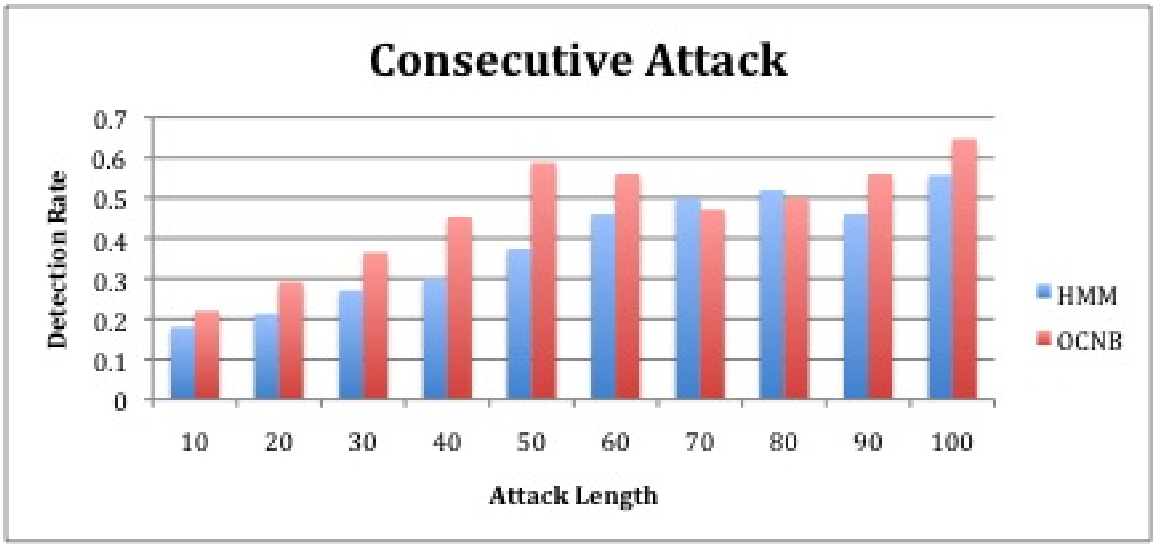


Figure 22: Detection Rate of Consecutive Attack for HMM and OCNB

Below in Figures 24, 25 and 26 we display ROC curve for examples of CA tested with HMM and OCNB for attack lengths 10, 50 and 100 respectively. We can observe from these ROC curves that the efficiency for HMM and OCNB is almost the same for this examples.

We can conclude that detection rate for OCNB-based masquerade detection technique is better than HMM-based masquerade detection technique for the above example in case of Consecutive Attack from Figure 23.

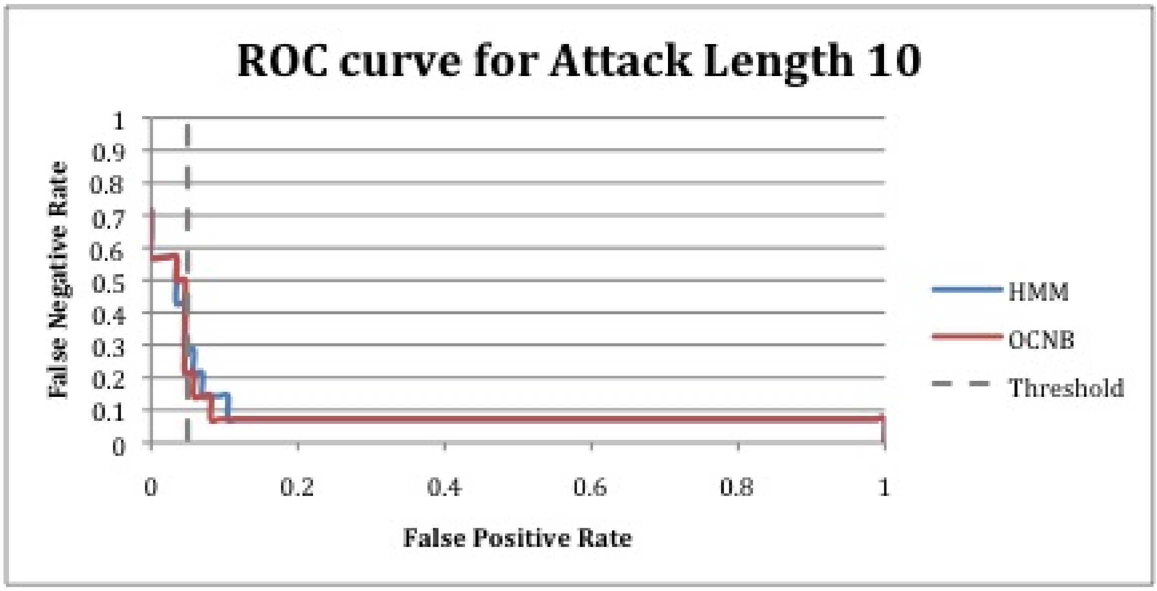


Figure 23: ROC curve for CA, |A| = 10

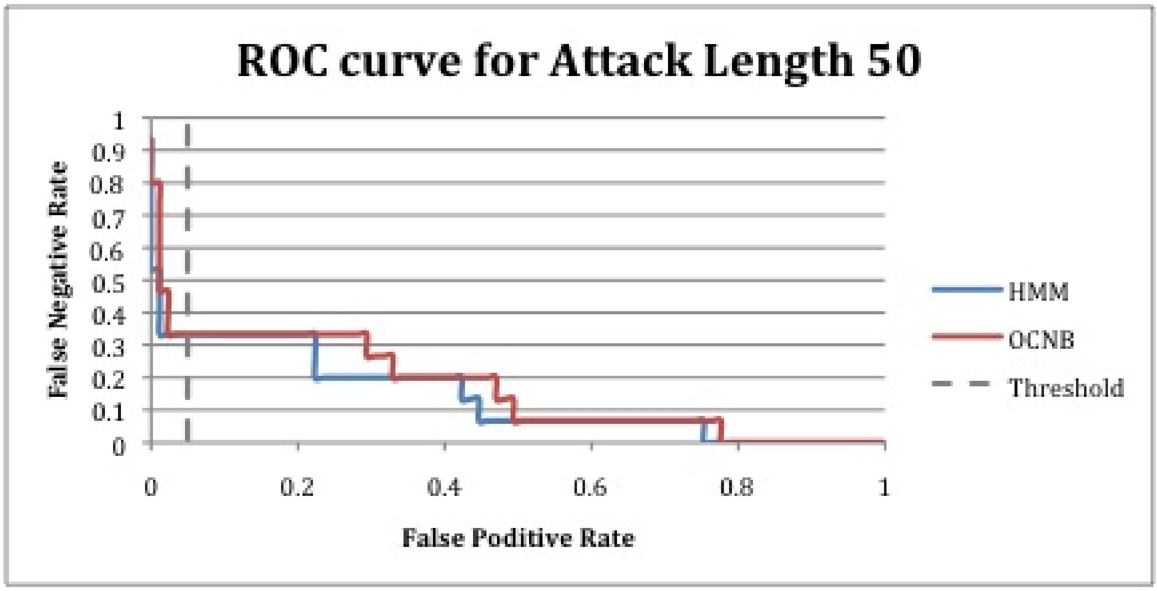


Figure 24: ROC curve for CA, |A| = 50

5.4 Consecutive Random User Attack

5.4.1 Discussion

Consecutive Random User Attack (CRUA) is similar to the Consecutive Attack (CA). In this attack, the attack commands are placed in a consecutive manner as in

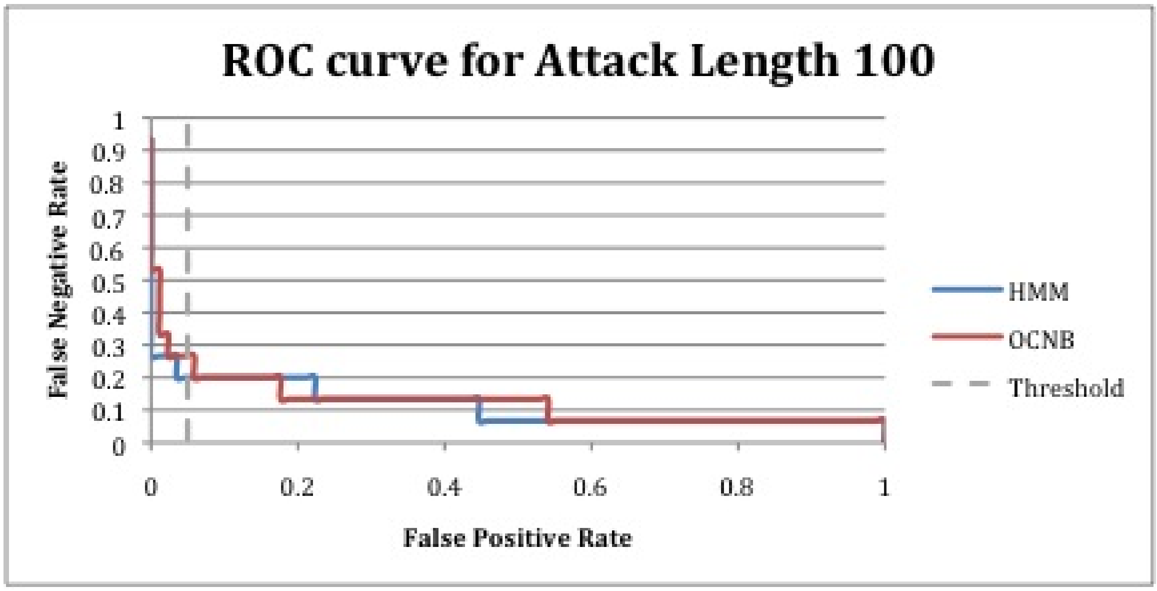


Figure 25: ROC curve for CA, |A| = 100

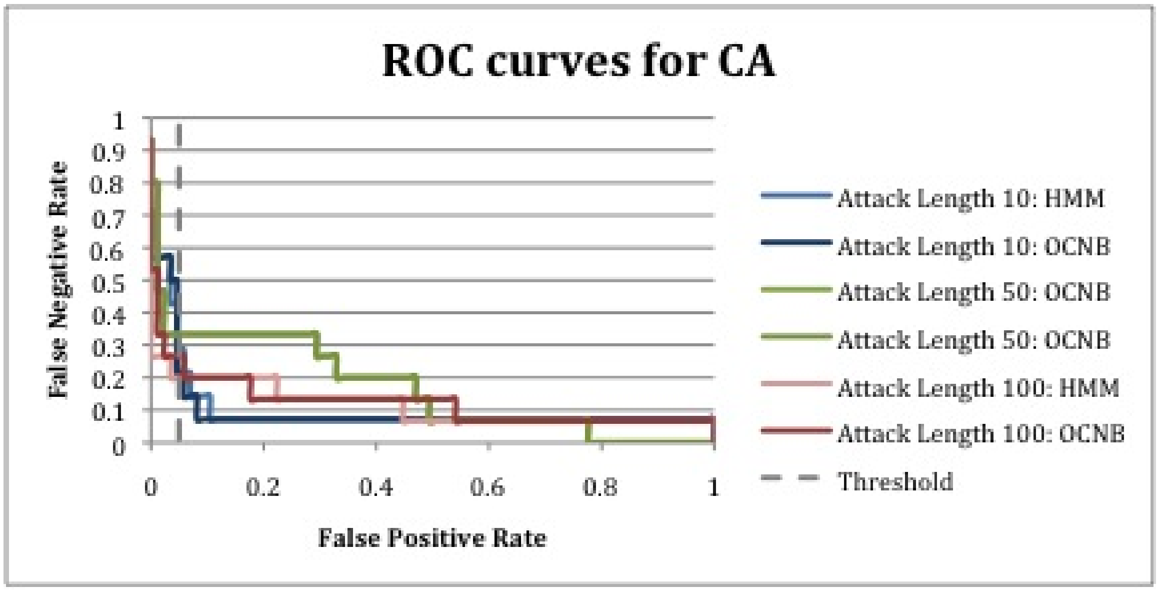


Figure 26: ROC curve for CA

Consecutive Attack. But the procedure to obtain the attack commands is different. In Consecutive Random User Attack, all the attack commands are obtained from all distinct users. In this attack, we want all the attack commands from different users. We first obtain attack commands from |A| different users and insert them consecutively in a block. After inserting the attack commands we pad the rest of the block with self-user commands. Below is the algorithm for Consecutive Random User Attack.

Algorithm:

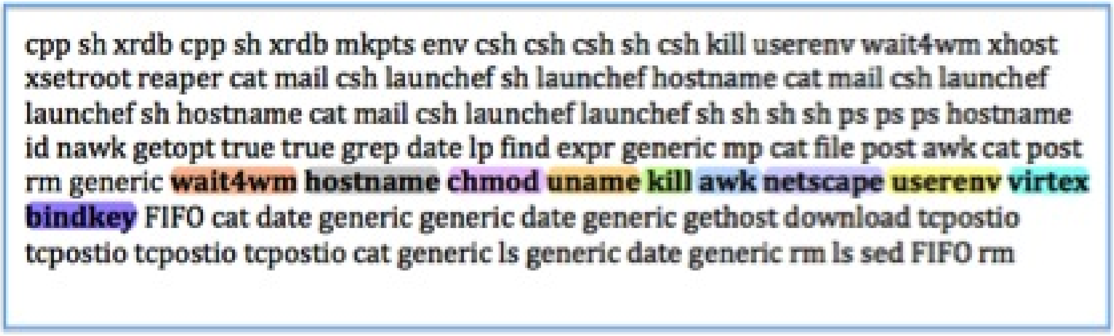
For every user:

1. *Select a random position say rdpt in the attack block, such that rdpt +* $|A| \leq$*100*
2. *Obtain* $|A|$ *random commands one from each of* $|A|$ *random users*
3. *Insert these commands in the attack block starting at rdpt in a consecutive manner*
4. *Pad the rest of the block of self commands*

The Figure 28 below is an example of Consecutive Random User Attack with attack length 10, where the highlighted commands are the attack commands. These 10 attack commands are obtained from 10 different users selected randomly. The attack commands are then inserted in the block in a consecutive manner. The non-highlighted commands are the padding commands.

5.4.2 Experimental Results

We then test the Consecutive Random User Attack with both HMM and OCNB. The graph in Figure 29 shows the detection rate of attack blocks for a user. We can see the detection rate increases with increase in attack length in both the masquerade detection techniques namely, HMM and OCNB. But HMM performs better than OCNB in this example. At attack length 10, the detection rate for HMM and OCNB is about 20% but for attack length 50, detection rate for HMM is about 50% and the



cpp sh xrdb cpp sh xrdb mkpts env csh csh csh sh csh kill userenv wait4wm xhost xsetroot reaper cat mail csh launchef sh launchef hostname cat mail csh launchef launchef sh hostname cat mail csh launchef launchef sh sh sh sh ps ps ps hostname id nawk getopt true true grep date lp find expr generic mp cat file post awk cat post rm generic wait4wm hostname chmod uname kill awk netscape userenv virtex bindkey FIFO cat date generic generic date generic gethost download tcpostio tcpostio tcpostio tcpostio cat generic ls generic date generic rm ls sed FIFO rm

Figure 27: Example of Consecutive Random User Attack of |A| = 10

OCNB detection rate is about 40%. HMM could detect more than 50% of the attack blocks beyond attack length 50 but in case of OCNB, more than 50% of the attack blocks are detected for attack length more than 60. The examples of Consecutive Random User Attack can be found in the Appendix.

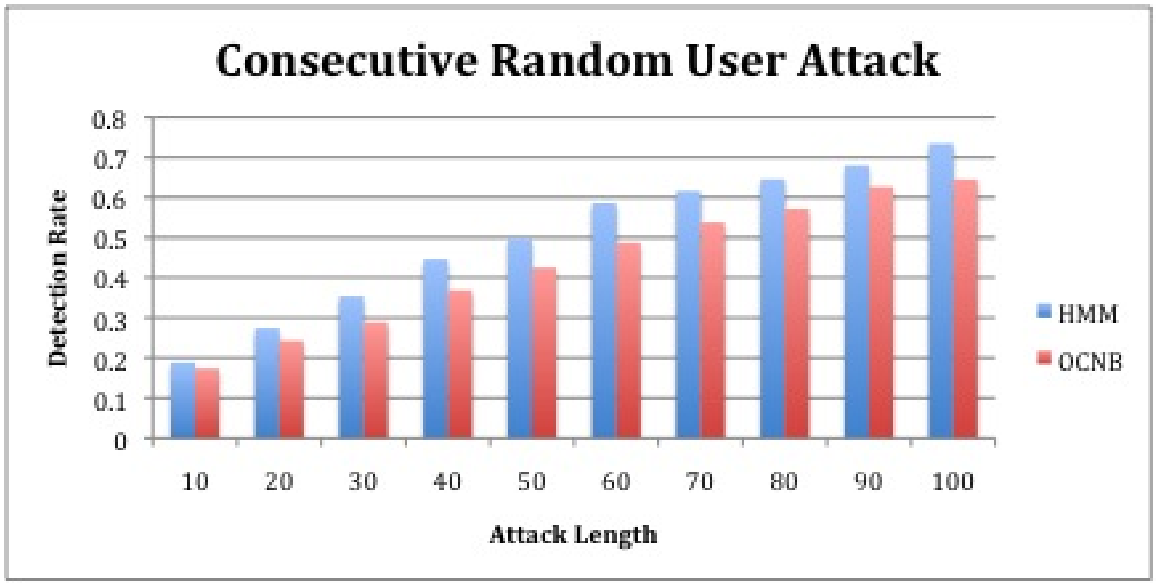


Figure 28: Detection Rate of CRUA with HMM and OCNB

The Figures below 30, 31, 32 and 33 are the ROC curves plotted by HMM and OCNB when tested with CRUA.

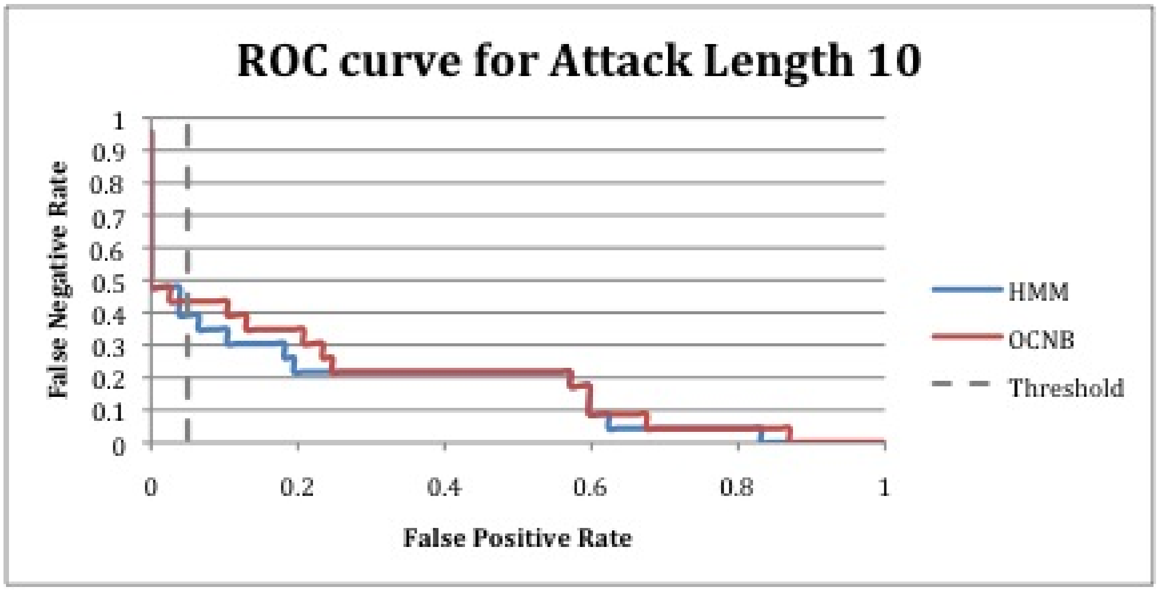


Figure 29: ROC curve for CRUA, |A| = 10

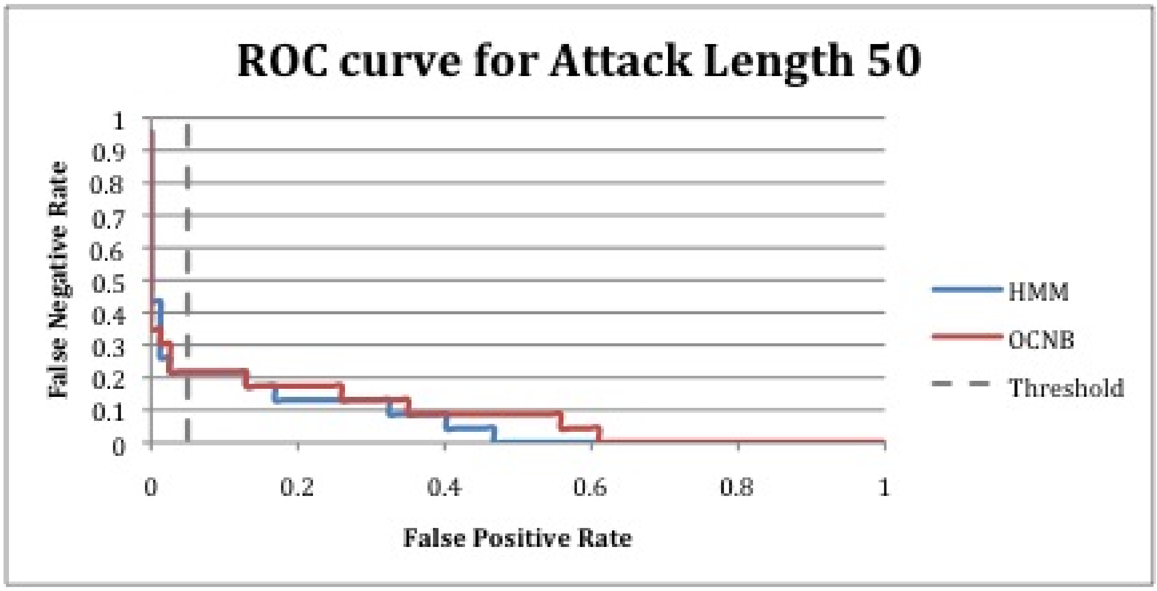


Figure 30: ROC curve for CRUA, |A| = 50

In Figure 30, 31 and 32 we plot the ROC curve for a user. We observe that HMM performs better in CRUA. Thus we conclude for this example, HMM is more efficient than OCNB for Consecutive Random User Attack.

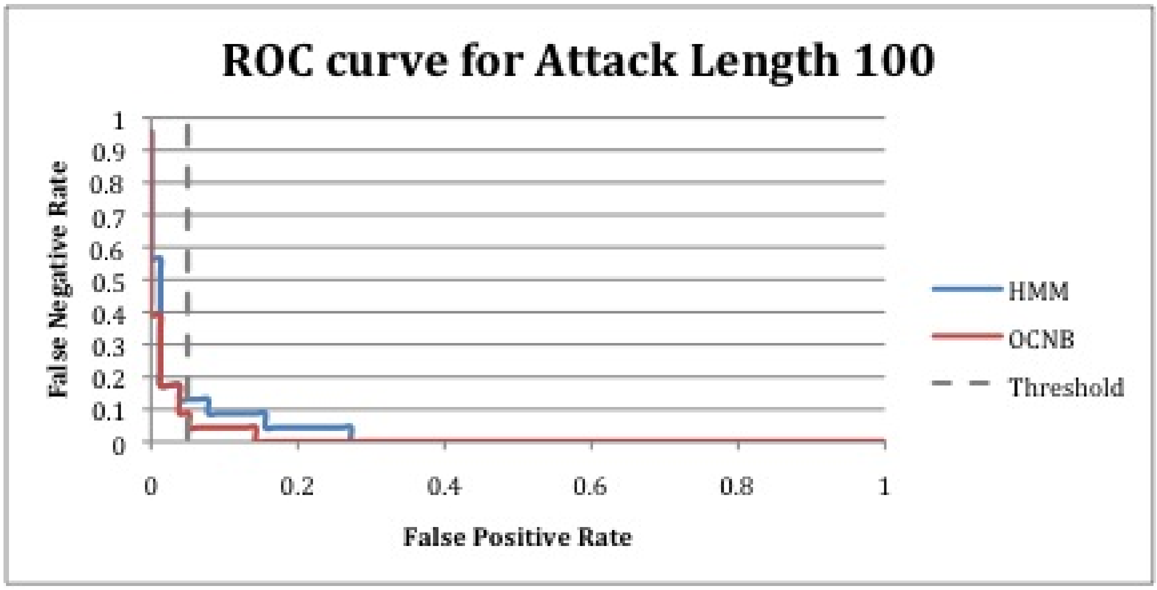


Figure 31: ROC curve for CRUA, |A| = 100

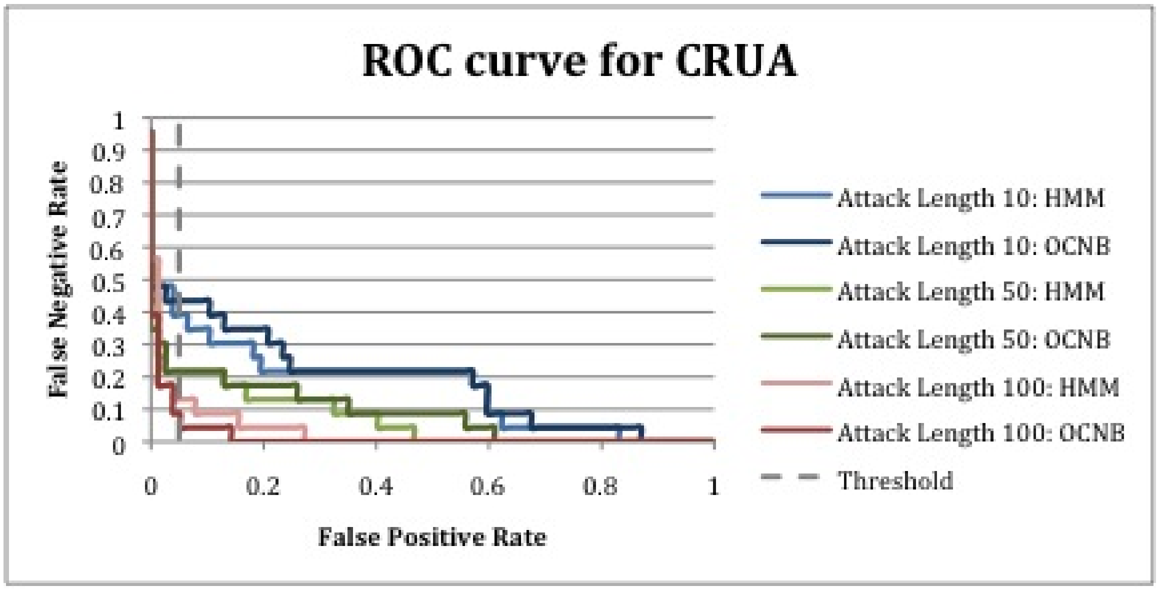


Figure 32: ROC curve for CRUA

CHAPTER 6

Analysis of Attacks

In this section we compare and analyze the results of the four attacks with both the masquerade detections based on HMM and OCNB. We do this by calculating their detection rates and accuracy rates.

For comparing the above generated four attacks, we perform a few experiments. We generate ten attacks of each of the four types. For each attack, we generate datasets of attack lengths varying from 10, 20, 30,. . .,100. We do this for all the 50 Users. Then each of these attacks are tested with both the masquerade detection techniques namely HMM and OCNB. We then calculate detection rate and accuracy rate for each of these attacks. Detection rate illustrates how well the detection technique identifies the attack data and the accuracy signifies how well the detection technique identifies both attack blocks as well as self-user blocks. While obtaining accuracy rate and detection rate we keep the false positive rate to be 0.05 that is 5%.

In the Table 1 below, we can generate the averaged detection of all the four attacks named Scattered Attack (SA), Scattered Sorted Attack (SSA), Consecutive Attack (CA) and Consecutive Random User Attack (CRUA) tested with Hidden Markov Model (HMM) and One-Class Naïve Bayes (OCNB). Detection rates for attack lengths 10, 20, 30,. . .,100 are shown below.

Below Table 2, displays the average accuracy rate of all the experiments mentioned above.

In Figures 34, 35, 36 and 37 we plot curves of detection rate and accuracy rates from above generated results to compare HMM and OCNB for each of the attacks.

Table 1: Average of Detection Rate for the four attacks with HMM and OCNB

	SA		SSA		CA		CRUA	
| A |	HMM	OCNB	HMM	OCNB	HMM	OCNB	HMM	OCNB
10	0.20735	0.21645	0.18798	0.21309	0.18153	0.22012	0.10937	0.18321
20	0.25862	0.29319	0.25278	0.28726	0.22666	0.29436	0.12185	0.24999
30	0.33835	0.34528	0.3314	0.35196	0.27824	0.36401	0.1315	0.30495
40	0.42428	0.40199	0.37697	0.40170	0.32671	0.4054	0.15428	0.35917
50	0.48489	0.45805	0.42813	0.43830	0.37496	0.44230	0.17514	0.40572
60	0.52894	0.49541	0.47876	0.49336	0.43679	0.49745	0.19291	0.44007
70	0.55584	0.53503	0.52267	0.53070	0.46761	0.53924	0.21771	0.47777
80	0.58277	0.56846	0.53311	0.56765	0.49919	0.55767	0.24024	0.50899
90	0.60756	0.59376	0.55255	0.58969	0.54613	0.6140	0.24834	0.53682
100	0.62855	0.6220	0.56324	0.62814	0.58426	0.62808	0.26154	0.56136

Table 2: Average of Accuracy Rate for the four attacks with HMM and OCNB

	SA		SSA		CA		CRUA	
| A |	HMM	OCNB	HMM	OCNB	HMM	OCNB	HMM	OCNB
10	0.8549	0.85002	0.85356	0.85696	0.85272	0.85032	0.83564	0.8723
20	0.86118	0.85882	0.86124	0.86612	0.85812	0.85912	0.84102	0.8844
30	0.8725	0.86506	0.87128	0.87592	0.86504	0.86752	0.84474	0.896
40	0.88156	0.87226	0.8774	0.8822	0.87028	0.87244	0.84802	0.90264
50	0.88788	0.87926	0.88468	0.88744	0.8772	0.87776	0.8505	0.90874
60	0.8926	0.884	0.88976	0.89248	0.88456	0.88428	0.85398	0.91222
70	0.89658	0.889	0.8952	0.89792	0.88812	0.88916	0.8592	0.91632
80	0.90092	0.89298	0.89692	0.90116	0.89216	0.89132	0.86144	0.91908
90	0.90396	0.89642	0.89856	0.90792	0.89788	0.89916	0.86112	0.92178
100	0.90638	0.89984	0.89932	0.91208	0.90248	0.90088	0.86146	0.9227

In Figure 34 we can see that the detection rate for Scattered Attack is initially higher for OCNB but with increase in attack length the detection rate is greater for HMM. Whereas the accuracy rate for HMM and OCNB is almost same. So we can say that HMM outperforms OCNB in SA.

In Figure 35, we plot detection rate and accuracy rate curves for SSA tested with

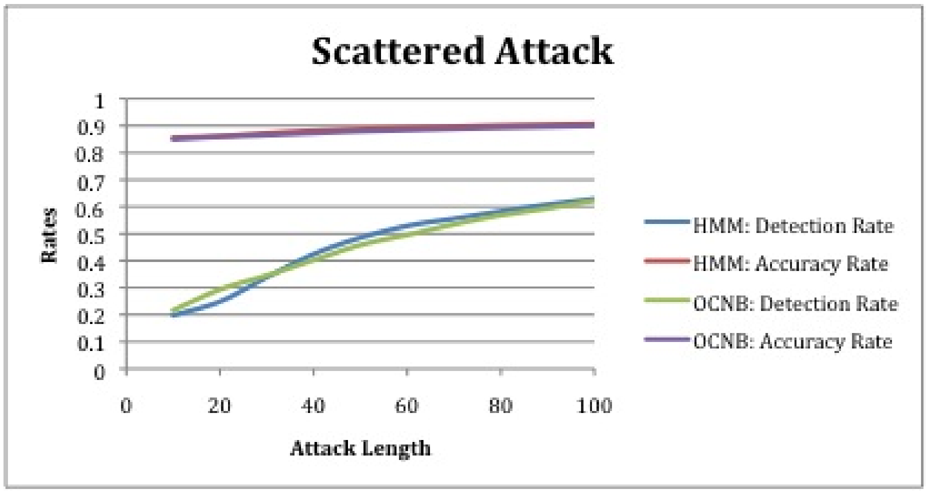


Figure 33: Comparison between HMM and OCNB for Scattered Attack

HMM and OCNB. We view that detection rates for OCNB are higher than HMM for all the attack lengths. The accuracy is more or less same for both HMM and OCNB. We can thus, conclude OCNB outperforms HMM in case of SSA.

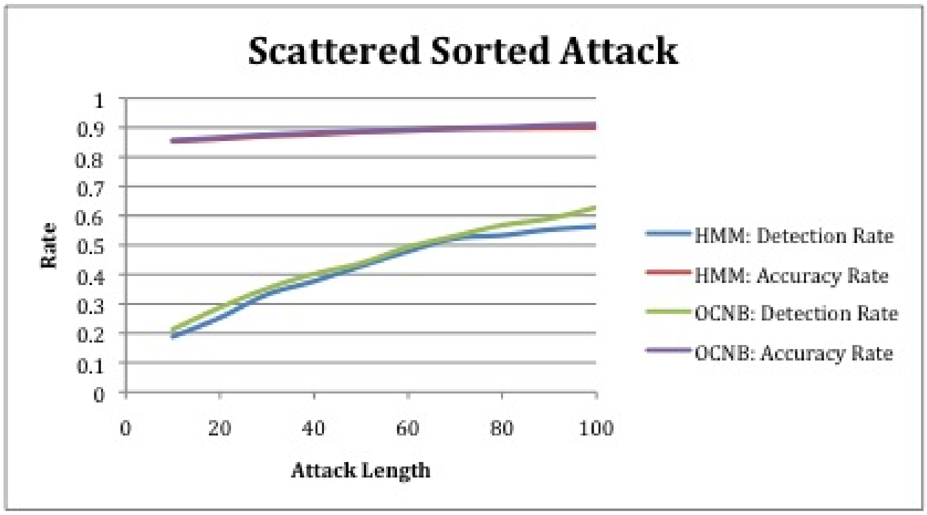


Figure 34: Comparison between HMM and OCNB for Scattered Sorted Attack

In Figure 36, below we can see results of Consecutive Attack tested with both HMM and OCNB. The detection rate and accuracy rate for OCNB is higher for all attacks lengths as compared to HMM. OCNB performs better than HMM in case of

Consecutive Attack.

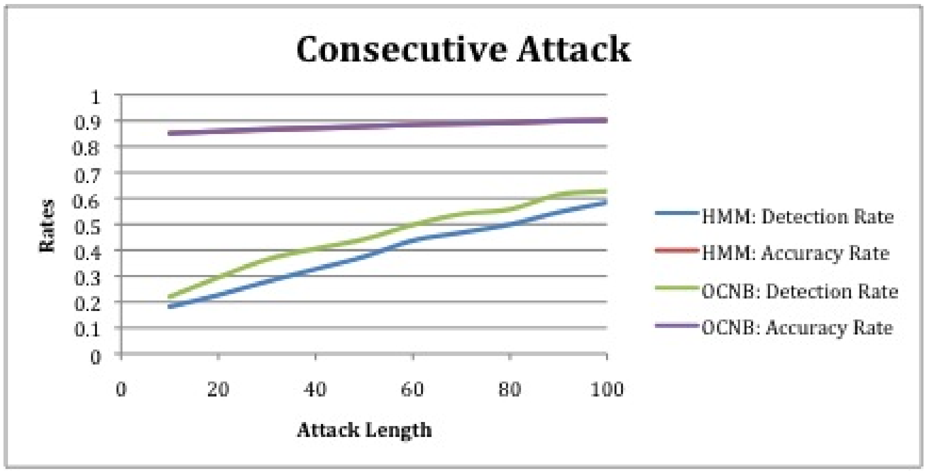


Figure 35: Comparison between HMM and OCNB for Consecutive Attack

In Figure 37 below, the results of Consecutive Random User Attack are shown graphically. The detection rate for HMM is better than OCNB for all the attack lengths. But the accuracy rate remains more or less the same for both HMM and OCNB.

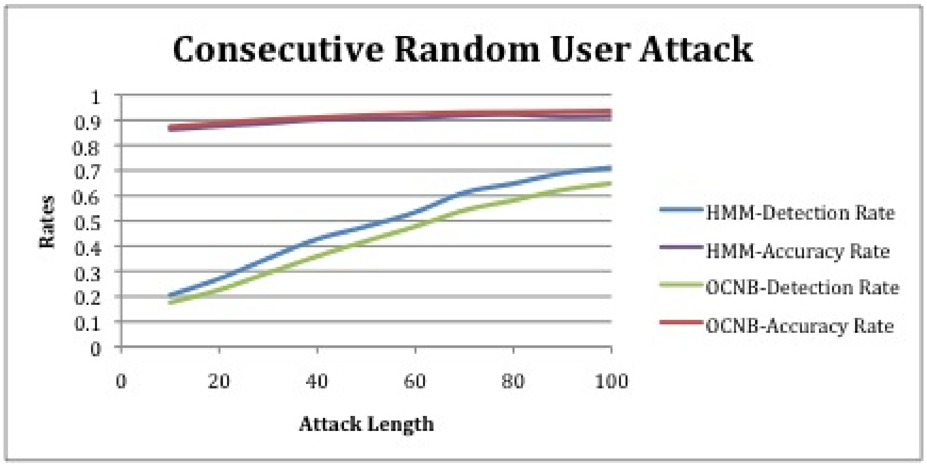


Figure 36: Comparison between HMM and OCNB for Consecutive Random User Attack

Thus from the Figures 34, 35, 36 and 37 we conclude, HMM performs better in case of Scattered Attack (SA) and Consecutive Random User Attack (CRUA) but

OCNB performs better in Scattered Sorted Attack (SSA) and Consecutive Attack (CA).

CHAPTER 7

Conclusion

A masquerade detection technique identifies the masquerader. In a UNIX command-based domain, a masquerade detection technique identifies attack blocks and differentiates them with the self-blocks. In this report we generated four distinct attacks and compared their results with two very well known masquerade detection techniques. The four attacks generated namely Scattered Attack, Scattered Sorted Attack, Consecutive Attack and Consecutive Random User Attack. We tested these four attacks with two well-known masquerade detection techniques, Hidden Markov Model and One-Class Naïve Bayes. The detection rates and accuracy rates were generated for each attack by setting the false positive to 0.05.

Scattered Attack can successfully evade Hidden Markov Model-based masquerade detection technique for attack length 10 to 50 attack commands, whereas Scattered Attack can evade One-Class Naïve Bayes up to 60 attack commands. As attack length increases the detection rate and accuracy rate for both masquerade detection techniques based on Hidden Markov Models and One-Class Naïve Bayes gradually increases. But, Hidden Markov Model-based masquerade detection technique outperforms One-Class Naïve Bayes. In case of Scattered Sorted Attack, it can evade both HMM-based masquerade detection and One-Class Naïve Bayes till attack length of 60 commands are inserted in the block, beyond attack length of 60 commands both these techniques were able to detect more than 50 percentage of the attack blocks, but One-Class Naïve Bayes performs better. Whereas, Consecutive Attack can successfully evade HMM-based masquerade detection even when we insert 80 percentage of the block with attack commands and One-Class Naïve Bayes with 60 percentage of attack commands. One-Class Naïve Bayes performs better in detecting Consecutive Attack. On the contrary in case of Consecutive Random User Attack, HMM detects better One-Class Naïve Bayes and Consecutive Random User Attack can evade up to attack length of 60 attack commands in Hidden Markov Model-based masquerade detection technique and 70 attack commands in case of One-Class Naïve Bayes.

CHAPTER 8

Future Work

We would like to analyze and gain knowledge about additional machine learning techniques to test with these attacks. In the future, we would like to develop a new and efficient masquerade detection algorithm by using salient features of all the pre-existing techniques. We would also like to research more in the field of masquerade detection and develop a technique that could not only detect masquerade attack, but also efficiently separate attack commands and padding commands.

LIST OF REFERENCES

[1] Bertacchini, M. & Fierens, PI. (2009). A Survey on Masquerader Detection Approaches. *CIBSI*, Retrieved August 25, 2011, from: `http://www.criptored.upm.es/cibsi/cibsi2009/docs/Papers/ CIBSI-Dia2-Sesion5(2).pdf`

[2] Chandola, V., Banerjee, A. and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 1-58

[3] Chen, L. and Dong, G. (2006). Masquerader detection using OCLEP: One-class classification using length statistics of emerging patterns.*Proceedings of Web-Age Information Management Workshops (WAIM 06).*

[4] Dash, S. K., Reddy, K. S. and Pujari, A. K. (2011), Adaptive Naïve Bayes method for masquerade detection.*Security Comm. Networks*, 4: 410417. doi: 10.1002/sec.168

[5] Devarakonda, N.R., Pamidi, S. and Kumari, V.V. (Dec. 2011). ABIDS system using Hidden Markov Model. *Information and Communication Technologies (WICT)*, 11(14), 319-324.

[6] El-Ghali, M. and Masri, W. (2009). Intrusion detection using signatures extracted from execution profiles. *Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems*,17-24.

[7] Hastie, T., Tibshirani, R., Friedman, et al. (2009). The elements of statistical learning: Data mining, inference, and prediction. *Springer-Verlag Publishers.*

[8] Huang, L. and Stamp, M. (2011). Masquerade detection using profile hidden markov models. *Computers and Security*, 30(8), 732.

[9] Killourhy, K. and Maxion, A. (2007). Learning from a Flaw in a Naïve-Bayes Masquerade Detector. Workshop on Machine Learning in Adversarial Environments, 20, 1-2. Retrieved from: `http://www.cs.cmu.edu/~ksk/KillourhyMaxion2007a.pdf`

[10] Kim, H. , and Cha, S. (2005). Empirical evaluation of svm-based masquerade detection using unix commands. *Computers and Security*, 24(2), 160.

[11] Lin, D. and Stamp, M. (2011, August 3). Hunting for undetectable metamorphic viruses. *Journal in Computer Virology*, 7(3), 201- 214

[12] Maxion, R. and Townsend, T. (2002). Masquerade detection using truncated command lines.*DSN*, pp. 219-228.

[13] Murali, A. and Rao, M. (August 2005). A Survey on Intrusion Detection Approaches. *Information and Communication Technologies*, 2005. ICICT 2005. First International Conference, 233- 240 doi: 10.1109/ICICT.2005.1598592.

[14] Patel, H. (2009). Intrusion Alerts Analysis Using Attack Graphs and Clustering. *Master's Projects. Paper 46* , Retrieved from: `http: //scholarworks.sjsu.edu/etd_projects/46`

[15] Rabiner, L.R. (Feb 1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol. 77, no.2, pp.257-286, doi: 10.1109/5.18626. Retrieved from: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=18626\&isnumber=698`

[16] Receiver operating characteristic. In Receiver operating characteristic-Wikipedia, the free encyclopedia. Retrieved October 19, 2011, from `http://en.wikipedia.org/wiki/Receiver_operating_characteristic.`

[17] Runwal, N., Low, R. and Stamp, M. (2012). Opcode graph similarity and metamorphic detection. *Journal in Computer Virology.* 1-16.

[18] Schonlau, M., DuMouchel, W., Ju W-H., Karr, AF., Theus, M., Vardi, Y. (Feb 2001). Computer Intrusion: Detecting Masquerades. *Stat Sci.* 16(1): 58-74.

[19] Schonlau, M. Masquerading User Data. In Matthias Schonlau's (Matt Schonlau) Homepage. Retrieved August 30, 2011, from `http://schonlau.net/.`

[20] Schonlau, M., and Theus, M. (2000). Detecting masquerades in intrusion detection based on unpopular commands. *Information Processing Letters*, 76(1/2), 33.

[21] Sensitivity and specificity. In Sensitivity and specificity From Wikipedia, the free encyclopedia. Retrieved April 23, 2012, from `http://en.wikipedia.org/wiki/Sensitivity_and_specificity.`+

[22] Sommer, R. & Paxson, V. (2003). Enhancing byte-level network intrusion detection signatures with context.*Proceedings of the 10th ACM conference on Computer and communications security*, 262-271.

[23] Sharma, A. & Paliwal, K. (August 20, 2007). Detecting masquerades using a combination of Naïve Bayes and weighted RBF approach. *Journal in Computer Virology*, 3, 237-245. doi: 10.1007/s11416-007-0055-z

[24] Stamp, M. (2011). Information security: principles and practice (2nd ed.): *Wiley*

[25] Stamp, M. (February 22, 2012). A Revealing Introduction to Hidden Markov Model. Retrieved March 1, 2012, from `http://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf.`+

[26] Tapiador, J., and Clark, J. (2011, May 10). Masquerade mimicry attack detection: A randomised approach. *Journal in Computer Virology*, 30(5), 297-310.

[27] Tavallaee, M. , Stakhanova, N. , and Ghorbani, A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man and Cybernetics: Part C - Applications & Reviews*, 40(5), 516.

[28] Tejinder, A. (2009). Intrusion Detection And Prevention System: CGI Attacks. *Master's Projects.* Paper 41. Retrieved from: *http://scholarworks.sjsu.edu/etd_projects/41*

[29] Varian, L. (2010). Intrusion Detection And Prevention System: SQL-Injection Attacks. *Master's Projects. Paper 16.* Retrieved from: `http://scholarworks.sjsu.edu/etd_projects/16`+

[30] Venkatachalam, S. (2010). Detecting undetectable computer viruses. Master's Projects. Paper 156. Retrieved from: `http://scholarworks.sjsu.edu/etd_` projects/156+

[31] Wang, K. and Stolfo, S. (2003). One class training for masquerade detection. ICDM Workshop on Data Mining for Computer Security.+

[32] Wing, W. and Stamp, M. (August 11, 2006). Hunting for Metamorphic Engines. *Journal in Computer Virology*, (2), 211-229.

APPENDIX

Appendix

A.1 Additional Results

The Figures 38, 39, 40 and 41 shows scores of HMM and OCNB when applied to CA with |A| of 10 and 50 respectively.

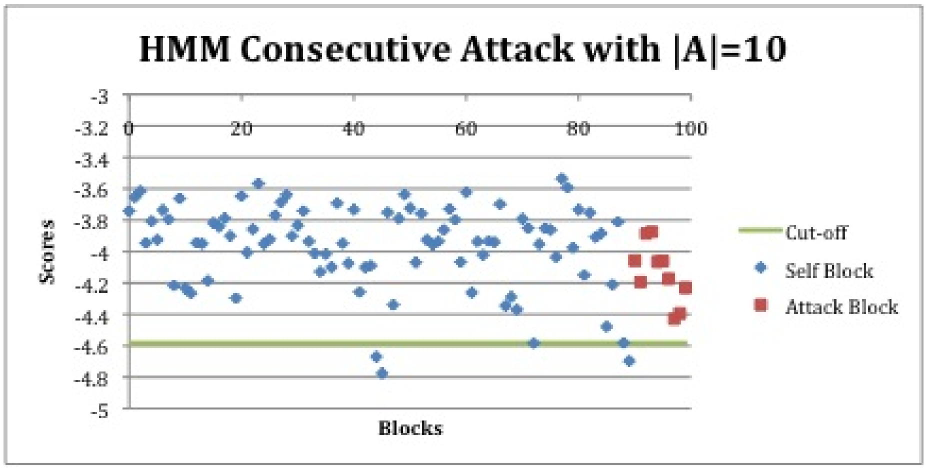


Figure A.37: Scores by HMM for Consecutive Attack of length 10 for User 45

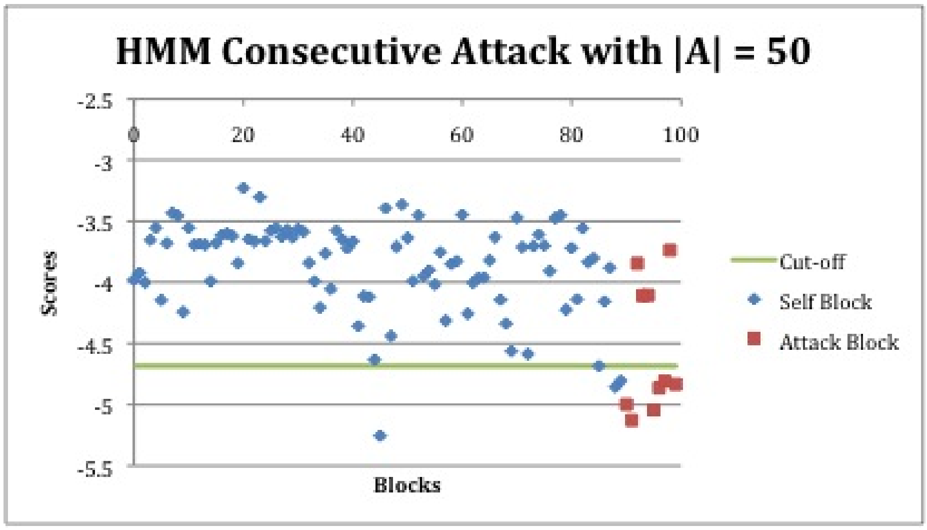


Figure A.38: Scores by HMM for Consecutive Attack of length 50 for User 45

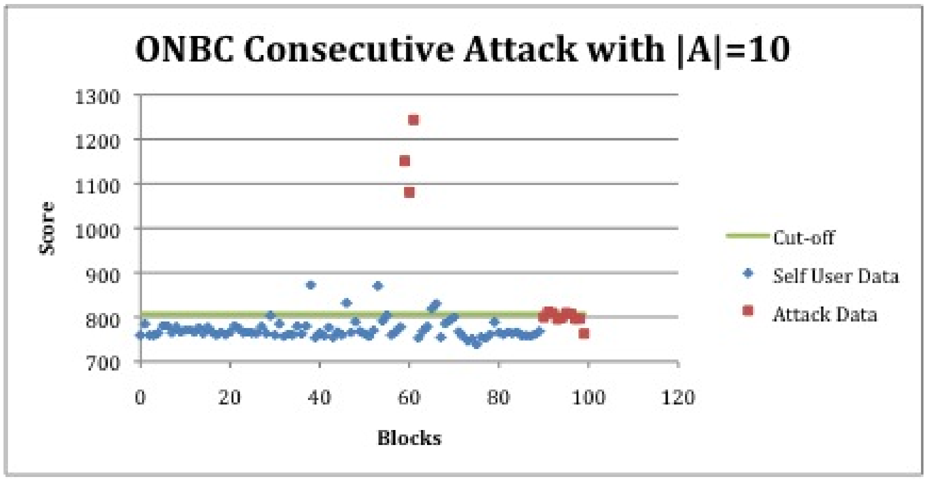


Figure A.39: Scores by OCNB for Consecutive Attack of length 10 for User 45

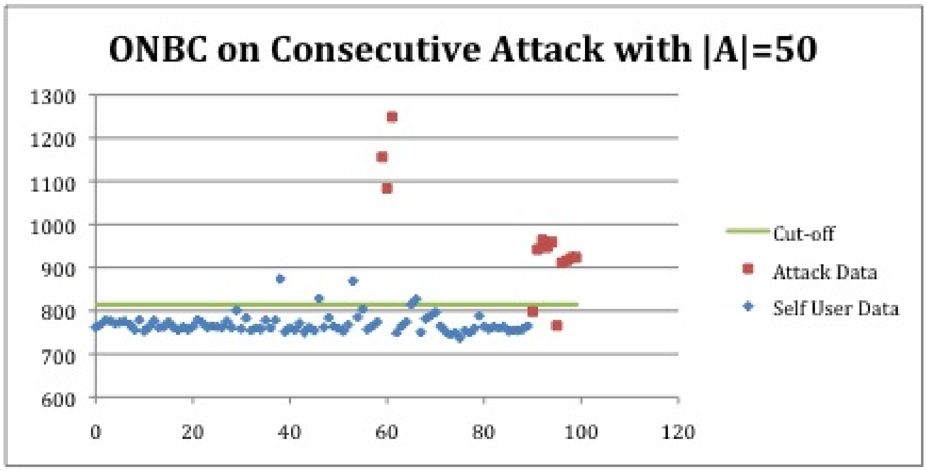


Figure A.40: Scores by OCNB for Consecutive Attack of length 50 for User 45