Intrusion Detection: Detecting Masquerade Attacks Using UNIX Command Lines

Dr. Robert F. Erbacher
Utah State University
Logan, Utah 84322-4205
435.797.3291
Robert.Erbacher@usu.edu

Shashi Prakash
Utah State University
Logan, Utah 84322-4205
435.797.2451
prakash@cc.usu.edu

Chet L. Claar
Utah State University
Logan, Utah 84322-3515
435.797.2285
Chet.Claar@usu.edu

Jason Couraud
Utah State University
Logan, Utah 84322-3515
435.797.2285
JasonCouraud@usu.edu

Abstract

Masquerading attacks use legitimate means, such as valid usernames and passwords, to bypass safeguards ensuring confidentiality, integrity, and availability of information resources. Often these attacks originate from inside the organization. Attackers using valid credentials are difficult to detect without first generating a profile for the user credentials and then comparing user activity against established patterns. Significant anomalies in the user's behavior patterns can be detected using a data mining-based classification algorithm. This project uses a naïve Bayes classifier to discover consistent and useful user behavior patterns which can then be used to discern anomalous behavior and known attacks. Experiments are performed by examining the data set comprised of truncated UNIX command sequences to detect masquerades based on the probability of commands.

Keywords: *Anomaly detection, classifiers, fraud detection, masquerade detection, naïve Bayes, user command data*

1. Introduction

Masquerade detection is a difficult proposition. Masqueraders infiltrate a system using legitimate means (valid username & password) to bypass security measures put in place to protect the integrity of the system. Many commercial intrusion software packages are sophisticated enough to detect masquerade attacks. However, detected masquerade attacks often demand further analysis because of the high rate of false positives that are generated during detection (Lee and Stolfo, 1998).

Current approaches to masquerade detection follow one of two techniques. The first technique seeks to determine future behavior based on past actions. The second technique uses a statistical approach to determine if an action constitutes an attack. Yet, both techniques rely on anomaly and misuse detection.

Masquerade detection is a process of gathering information about users and developing a profile for each user. These profiles are constructed of information regarding login time, login location, session duration, CPU time, commands issued, etc. Once the profile has been developed user command logs are compared against the profiles. Behavior that does not match the profile is then designated as an attack.

Several experiments have been performed to decrease the false positive detection rate and to increase detection rate in general. Schonlau et al. (2001) and Maxion and Townsend (2002) used a statistical approach for detecting masquerades. Their work will be discussed in detail later. Mention is made because the data set developed by Schonlau et al. was used by Maxion and Townsend and the experiments reported in this paper. Other methods include decision tree, neural networks, expert systems, compression, and machine learning (Schonlau and Theus, 2000, Greenberg, Heady et al. 1990, Lunt, 1993, Szymanksi and Zhang, 2004, and Peng, 2005).

2. Problem and approach

This paper explores the masquerading problem by classifying blocks of user-command data into one of two categories: self or non-self. Self data is data from a legitimate user, non-self being data from a masquerader. The data is from the data set devised by Schonlau et al. (2000) and is referred to as SEA. The data set is 15000 truncated UNIX commands taken from 70 different users. Using 5000 commands, the algorithm builds the user profiles. The remaining 10000 commands are then used to test the algorithm. The same methodology used in Schonlau (2000) and Maxion (2003) was used in this paper. A more complete description is found in the experimental design section.

The algorithm used in this paper is a modified version of the naïve Bayes using bi-gram probability in the place of normal probability as was done by Maxion (2002). In this study we examine adjacent pairs of commands as a means of boosting hit rate and reducing false positives. A more complete treatment is found in the subsequent sections.

The following section reviews the recent literature related to the masquerading problem. The experimental design section explains the methodology used in this paper. The final two sections will discuss the results of the experiment and directions for future research.

3. Recent literature

Although great progress has been made in intrusion detection since 1980, the number of system vulnerabilities discovered every year is on the rise (Lunt 1993). Though most of these vulnerabilities are fixed within days or weeks of their discovery, effective intrusion detection systems are still a powerful preventative security measure. An intrusion detection system can either detect intruders breaking into a system or monitor legitimate users misusing system resources. There are many commercial intrusion detection systems on the market today. While the majority of present intrusion detection system approaches can handle many types of attacks, there are always certain threats (attacks) that have to be further analyzed because some attacks remain unnoticed due to a high positive or false alarm rate (Lee 1998).

Most of the current approaches for masquerade detection employ techniques and methods for both anomaly and misuse detection. These techniques vary in their approach. Some are based on techniques of predicting future patterns of behavior by utilizing established patterns, while others rely on statistical approaches for determining anomalous behavior.

Currently, the problem of detecting masquerades is a not research issue. Researchers from various fields have implemented different methods to solve the problem of masquerader detection using UNIX command sequence. For example, (Schonlau 2000, 2001), (Maxion, 2002, 2003) (Coull, 2003) (Heady, 1990) all explore the problem of masquerade detection. Some earlier research utilizing a UNIX command sequence is discussed below:

The IDES system was one of the first to work towards identifying the masquerade problems on a single-target host system using a rule-based expert system to detect known malicious activity (Lunt 1988).

Schonlau et al. (2000, 2001) presents a number of techniques for detecting masqueraders in an experiment in which six masquerade-detection schemes are compared on the same user data seeded with “masqueraders.” Some of the techniques used were original, while others were drawn from computer science literature. The investigators targeted a false-alarm rate of 1%. All of the methods had relatively low hit rates (ranging from 39.4% to 69.3%) and high false alarm rates (ranging from 1.4% to 6.7%).(Maxion 2002). Their approach uses statistical methods to detect masquerades based on popular and unpopular commands.

- **Uniqueness:** This method is based on the observation that commands not seen at the time of training are unlikely to be seen at the time of testing. If a command is not used by a number of users, then it is all the more unpopular. The authors define a test statistic that supports the bias towards using commands seen in training data in the test data. This implies that the frequency property of the command sequence is an important indicator of intrusion (Schonlau 2000, 2001). Uniqueness was a relatively poor performer in terms of detecting masqueraders, but was the only method able to approach the target false alarm rate of 1%.

- **The Incremental Probabilistic Action Modeling (IPAM):** This method, developed by Davison and Hirsh (1998), uses The Incremental Probabilistic Action Modeling, a single step command transition probability to predict the sequence of user commands. Too many false predictions indicate a possibility of masquerades. IPAM's performance ranks with those in the lowest-performing group.

- **Sequence-match:** Originally proposed by Lane and Brodley (1997), the authors compute a similarity match between the user profiles with the corresponding sequence of commands. A score lower than the threshold value indicates the presence of masquerades. On the Schonlau et al. data it was a poor performer.

- **Bayes 1-Step Markov Model:** Originally proposed by DouMouchel and Schonlau (2001), the authors utilize the information of 1-step command transition probabilities. They build transition matrices for each user's training and testing data. The detector triggers the alarm when there is a considerable difference between the training data transition matrix and the testing data matrix. This technique was the best performer in terms of correct detections, but failed to get close to the desired false alarm rate.

- **Hybrid Multi-Step Markov:** This method is based on Markov chains, and is due to Ju and Vardi (1999). The implementation of this model in Schonlau's (2001) work toggled between a Markov model and a simple independence model, depending on the proportion of commands in the test data that were not observed in the training data. It focuses on a subset of the most used commands and, thus, only needs to deal with a significantly reduced dimensionality. The performance of this method was among the highest of the methods tested.

- **Compression.** The idea behind the compression approach is that new data from a given user compresses at about the same ratio as old data from that same user, and that data from a masquerading user will compress at a different ratio and thereby be distinguished from the legitimate user. This idea is credited to Schonlau and Karr (2001) in Compression was the worst performer of the methods tested.

Recently, Maxion et al (2002, 2003) used the same dataset as Schonlau et al (2001) and demonstrated a statistical approach based on a naïve Bayes classification to detect masquerade attacks. Maxion improved upon the results obtained by Schonlau et al. by applying the naïve Bayes classification algorithm using the "bag of words" features. The

study of masquerade detection using truncated UNIX command lines, with each line consisting of just a single UNIX command, is common to the work of Schonlau (2000, 2001), and Maxion (2002).

Table 1 contains the results for some of the above methods using the Schonlau et al dataset available at http://www.schonlau.net. These results serve as the baseline of comparison for our experiment.

Method	Hits (%)	False Alarms (%)
N. Bayes (Updating)	61.5	1.3
N. Bayes (no Updat.)	66.2	4.6
Uniqueness	39.4	1.4
Hybrid Markov	49.3	3.2
Bayes 1-step Markov	69.3	6.7
IPAM	41.4	2.7
Sequence Matching	36.8	3.7
Compression	34.2	5.0

Table 3.1 Results of Previous Classification Methods

4. Experimental Design

4.1 Method

Our approach to masquerade user detection is based on the simple premise that the legitimate users will establish a profile based on the sequence of commands they use. This profile is termed a "signature". Our primary contribution is the recognition of behavior that deviates from a user's signature and flag it as a potential masquerade. In this method, we detect the masquerade behavior using a modified approach of the simple naïve Bayes classification algorithm using adjacent pairs of commands probability instead of the normal probability function.

We built a separate naïve Bayes classifier for each user. We then divided the entire dataset into two classes for training and testing. The training class consisted of a user's first 5000 commands, and the remaining 10000 commands served as testing data. We inserted the testing data with masquerade command blocks from other users based on some probability. During the training phase, the classifier built a profile for self and non-self using the training class, and the remaining test data were used to identify the masquerade and non-masquerade commands. Our experiment method followed the same methodology of Schonlau (2000) and Maxion (2003) for detecting masquerades. Our experiment focuses on the effects of changing the various parameters of the naïve Bayes algorithm on the detection rates.

4.2 Details

For the masquerade detection problem, the first task is to build a classifier that can detect the commands as either masquerade or non-masquerade. The following are the necessary things that have to be considered while building the naïve Bayes classifier.

4.2.1 Data

For our experiment, we used command stream data collected from the UNIX system's auditing mechanism. This data is the same as the data used in other masquerade detection studies conducted by Schonlau (2000) and Maxion (2003). Table 4.1 shows examples of some of the commands.

Command Name	User
sed	www
more	xxx
cat	yyy
netstat	zzz

Table 4.1. Example of Data Used for Classification.

Due to security issues, the command logs were stripped, and only the command name and the user name were retained. Further, in the command name field, we only consider the command and filter out the arguments for the analysis.

4.2.2 Creating Masquerade Blocks

The dataset of 70 users was collected over a period of time on a stand alone UNIX system. Among these 70 users, 50 users were randomly selected as the targets for intrusion, and the remaining 20 users were selected to serve as intruders. The data used to build profiles of legitimate users consisted of 5000 commands of 50 users that had not been contaminated and represented actual user's normal activity data used to generate a user signature. The remaining 10000 commands were used for testing. These commands were randomly injected with masquerader commands drawn from the remaining 20 users. Figure 4.1 shows the process of creating masquerade command blocks (Kwong 2004).

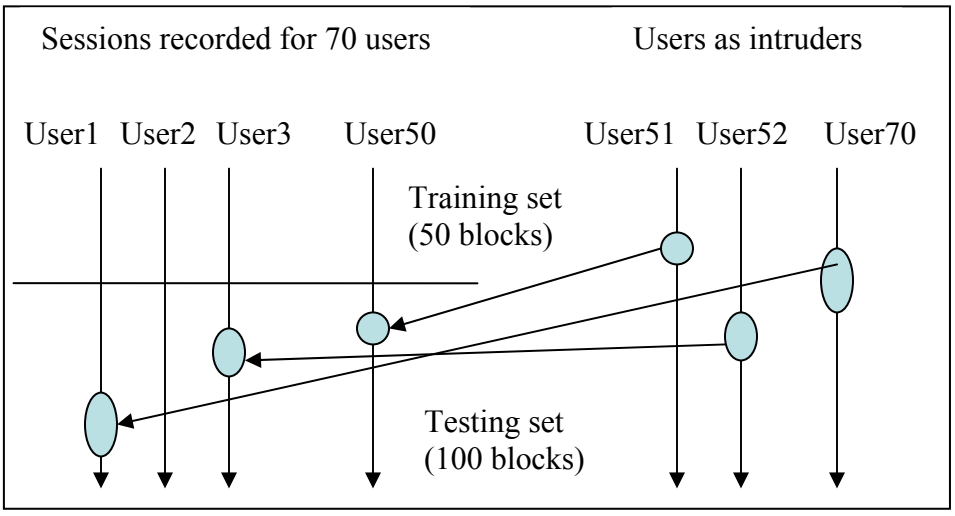

Figure 4.1. Splicing of sessions

At any given block of the test data, a masquerade starts with a probability of 1 percent. That user block is replaced with a masquerader block. If such a replacement was made for a given block, then the next user block would be replaced with a masquerader block from the same masquerader with a probability of 80 percent. Each block of commands was assigned as either a pure block or a masquerade block as the replacements were done probabilistically. This method is referred to as the Schonlau et al. (SEA) configuration.

4.2.3 Classification Procedure

A naïve Bayes classifier was constructed to detect masquerade behavior using the adjacent pairs of command probability. The classification procedure is comprised of the training and testing phase.

During the training phase the goal is to establish profiles of self and non-self, and to determine a decision threshold for discriminating between them. During this phase, the classifier built a profile of self and non-self based on command frequencies using the first 5000 commands of 50 users' data. It is always assumed that the training set is free from any masquerade data.

During the testing phase the accuracy and performance of the detection system is measured as compare the user profile generated during the training phase against the test data. The detector was presented with 100 blocks (10000 commands) of mixed, unlabelled, self and non-self data. Each test block was classified by the detector as self or non-self, according to whether the ratio of the log probabilities assigned to the block under the self and non-self models was greater than a threshold (non-self) or less than or equal to that threshold (self).

The main tasks of the training and testing phase are to match characteristic groups of commands in a test block with a user signature to detect the masquerade user commands and to measure the percentage of false positives generated.

4.2.4 Prediction and Analysis Phase

The naïve Bayes classifier was applied as a detection mechanism for masquerade blocks. The classifier was trained on the features of training data and predicted on the features of testing data. After the prediction, comparison of the true value from the SEA dataset and the value obtained from the prediction phase was used to determine the hit rate and false alarm rate.

4.2 Experiment Metrics and Parameters

Four parameters were identified (equation 3.1) that influence the naïve Bayes' detection score. The number of commands in the training data was also determined to be an influencing factor on the detection score.

$$P(u|d) = \frac{\text{Training Count } (u|d) + p}{\text{Training Data Length} + A * p}$$

a = alphabet size
p = naïve Bayes pseudocount
b = block size
t = threshold for detection

Equation 4.1

The alphabet size (a) was held constant to 122. The naïve pseudocount (p) was set to 0.01 and was used for smoothing, which controlled the sensitivity to previously unseen commands. The size of the block (b) was set to 100. The size of the training data was set to 5000. The above values were chosen for the parameters to maintain consistency with previous work done in masquerade detection. The threshold (t) was set between 10 and 30 percent to update the classifier during the experiment based on the sensitivity of the unique commands using 5-fold cross validation. The threshold value was set to 10 percent or 0.1 for the experiment.

By applying naïve Bayes to the dataset, the parameters that influence the detection rate were examined. We obtained the parameters based on the experiment settings of the naïve Bayes classifier algorithm that were used for determining the detection results. To facilitate comparison with other masquerade detection algorithms, we used false positive, false negative, false alarm, and hit rate metrics to determine how well our algorithm performed. Equations 4.2 and 4.3 (Coull 2003) summarize our algorithm's metric calculations.

To calculate the mismatch score, we used the ratio of the number of times a particular command in the test data set appears in the training data. The score is set either as 0 or 1.

$s^* = (c + a)/ (5000 + 0.01 * c_u)$;

s = score
c = # of occurrences of command in the user signature
a = Naïve pseudocount
c_u = # of unique commands

Equation 4.2 Score calculation.

$$\text{false negative}_{overall} = ([\Sigma^{i}_{users}(fn_i/n_i)]/c)*100$$

$$\text{hit rate}_{overall} = 100 - \text{false negative}_{overall}$$

f = number of false positives
n = number of non-intrusion command sequence blocks
u = number of users (50 in our case)

$$\text{false positive}_{overall} = ([\Sigma^{i}_{users}(f_i/n_i)]/u)*100$$

fn = number of false negatives
n = number of intrusion command sequence blocks
c= number of users who have at least one intrusion block

Equation 4.3 Metric calculations.

5. Results

Our results show that the performance of the naïve Bayes using the dual command probability is very competitive with the performance of experiments conducted by Schonlau(2000, 2001) and Maxion (2002). Our evaluation was conducted in the training phase. We evaluated the naïve Bayes classifier based on both command frequency and threshold for determining the classification performance.

5.1 Overall Results

The performance of masquerade detection was measured by hits, misses, and false alarm rates. The experiment obtained a hit rate of 60.17 percent, a miss rate of 39.82 percent, and a false alarm rate of 3.86 percent.. These results were obtained by comparing the location of masquerades in the command blocks. Table 5.1 summarizes the overall performance of the naïve Bayes classifier on a sequence of 100 commands.

Data	Schonlau Dataset [SEA]
Type of Data	Truncated
Training data	5000 commands
Testing Data	10000 commands
Block size	100
Hits%	60.17
Misses%	39.82
False Alarms%	3.86

Table 5.1 Naïve Bayes Classifier - Sequence Length = 100

These results are the trade-off between the correct detections and false detections. A Receiver Operation Characteristic curve (ROC curve) is often used to represent the results. The percentage of hits and false alarm rate are shown over the x and y axis respectively. Figure 5.1 shows in fine detail the ROC curve for the naïve Bayes classifier on sequences of 100 commands of 50 users, for a false alarm rate of 0 to 50 percent.

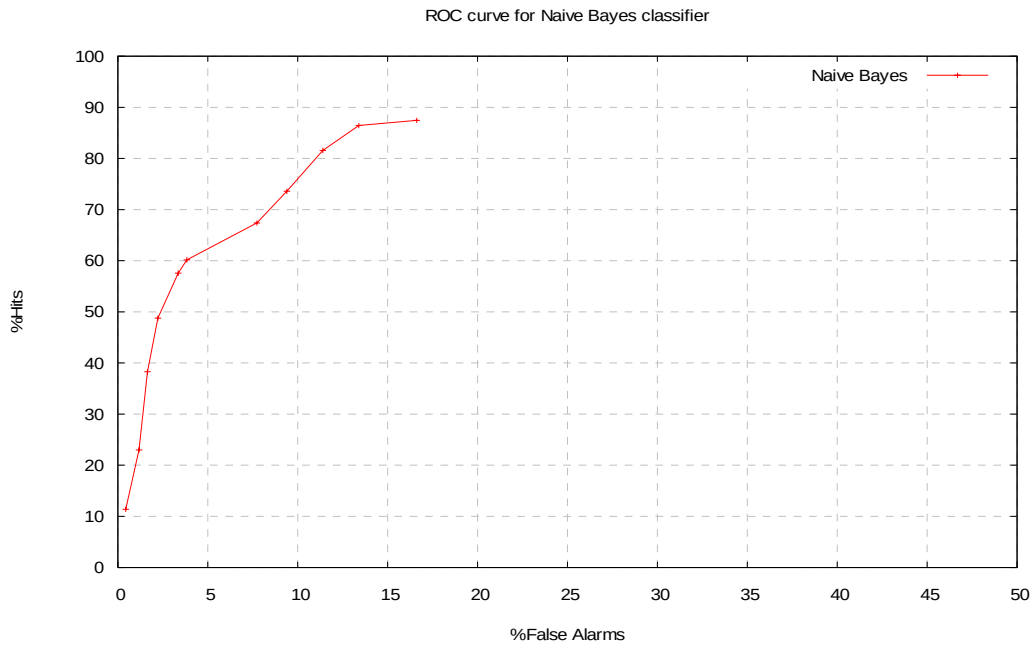

Figure 5.1. Relative operating characteristic (ROC) curve for the naïve Bayes classifier

Each point on the curve indicates the relation between the hit rate and false alarm rate. Points nearer to the upper left corner of the graph are the most desirable; as they indicate high hit rates and correspondingly low false alarm rates. From the Figure, we see the accuracy rate achieved is around 87 percent when considering hit rate alone for the data. However, when considering the false-alarm rate, which is quite important in a real-world situation, we were able to achieve results very competitive to those obtained by Maxion (2003) using the same dataset.

Consequently, the main goal in our approach of masquerade detection was to reduce the false alarm rate as far as possible.

5.2 Comparison with Prior Results

Compared to the previous approach, we were able to achieve a 16 percent reduction in the false alarm rate from 4.6 to 3.86 percent, while the detection (hit) rate also decreased by 8 percent from 66.2 to 60.17 percent making the experiment result roughly equivalent, in terms of classification accuracy, to that of the SEA and Maxion experiments. Table 5.2 shows the results obtained by Schonlau (2000, 2001)., Maxion (2003). and other researchers with a variety of detectors for a sequence of 100 commands.

Method	**Hits (%)**	**Missed Alarms (%)**	**False Alarms (%)**
N.Bayes (no updating)	66.2	33.8	4.6
N.Bayes (1v49 by Maxion)	62.8	37.2	4.63
N.Bayes (our method)	**60.17**	**39.82**	**3.86**
Uniqueness	39.4	60.6	1.4
Hybrid Markov	49.3	50.7	3.2
1-step Markov	69.3	30.7	6.7
IPAM	41.4	58.9	2.7
Sequence matching	36.8	63.2	3,7
Compression	34.2	65.8	5.0

Table 5.2. Methods for Masquerade Detection.

We studied the results from two different perspectives; accuracy measurement for different threshold values for command frequency and accuracy measurement for different parameters of the naïve Bayes classifier. We found that there was a considerable difference in hit and false alarm rates for different values of threshold and experimental parameters, clearly showing that command frequency and threshold rates played an important role in classification.

5.3 Threshold Determination

A threshold mechanism is required need to determine the point at which intrusion detection begins. We can set this threshold value by comparing a user signature to itself by choosing random blocks of 100 commands and aligning each of them to a 1000 command user signature sequence. This gives an initial score that is equal to the score that the test data should produce. We keep updating this score by averaging the current testing block's score with all the previous blocks' scores. We can consider percentages of this threshold as a cut off value, allowing us to customize the threshold for each user. Thus, if a particular user did not have consistently high scoring alignments with their user signature, this user's testing blocks will not be unduly flagged as intrusions.

This choice of threshold gives us the choice of sensitivities. This threshold can determine the number of false positives and negatives along with correct matches. We illustrated this by using a test file containing 100 rows and 50 columns. Each column corresponds to one of the 50 users. Each row corresponds to a set of 100 commands, starting with command 5001 and ending with command 15000. The entries in the files are 0 or 1. Zero (0) means that the corresponding 100 commands are not contaminated by a masquerader. One (1) means they are contaminated.

The graph in Figure 5.2 is for 50 users with a threshold varying from 10 to 100 percent. The false positives and false negatives increase and decrease respectively when the threshold values change. This graph shows that by choosing a value for the threshold, one can satisfy the sensitivity requirements of IDS.

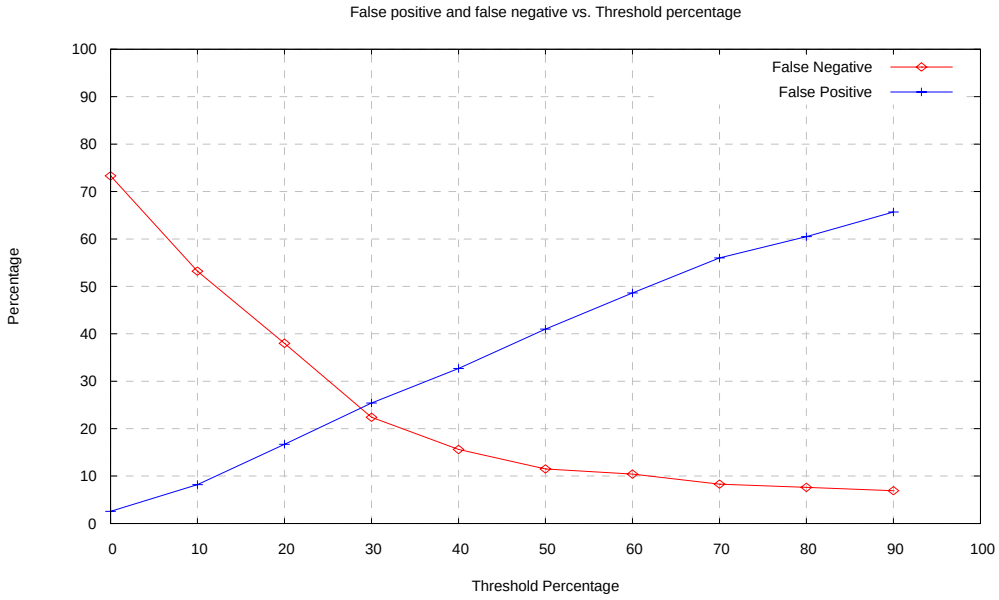

Figure 5.2 False positive and false negative vs. threshold percentage.

5.4 Detection of Masquerades in Command Blocks Based on the Threshold

In our experiments, we found that threshold value makes a slight difference in masquerade detection. To find the effect of the threshold, we used various threshold values for detecting masquerades in the command blocks. Though there was not much deviation in the results of various threshold values, we still found dependence of classification on threshold values. Figures 5.3 through 5.5 show the masquerade detection rates for our approach using threshold values varying from 10 to 30.

We chose the threshold value that was appropriate with the amount of sensitivity for command sequences with which we could train the classifier. If the concern is security, a higher threshold percentage could chosen a for a higher detection rate with a resulting higher false alarm rate. Conversely, if the concern is lowering the false alarm rate, a

lower threshold percentage would be chosen, with a resulting lower secure environment because of the lower detection rate.

Figure 5.3 shows the detection of masquerade blocks for threshold value of 10, 20 and 30 percent. From the graph, we can clearly see the detection of masquerade in the command blocks for each user is quite comparable to the detection rates obtained by Schonlau (2000). While there is not much difference in the masquerade block detection rate for different values, the false alarm is considerably less for a threshold value of 10 percent.

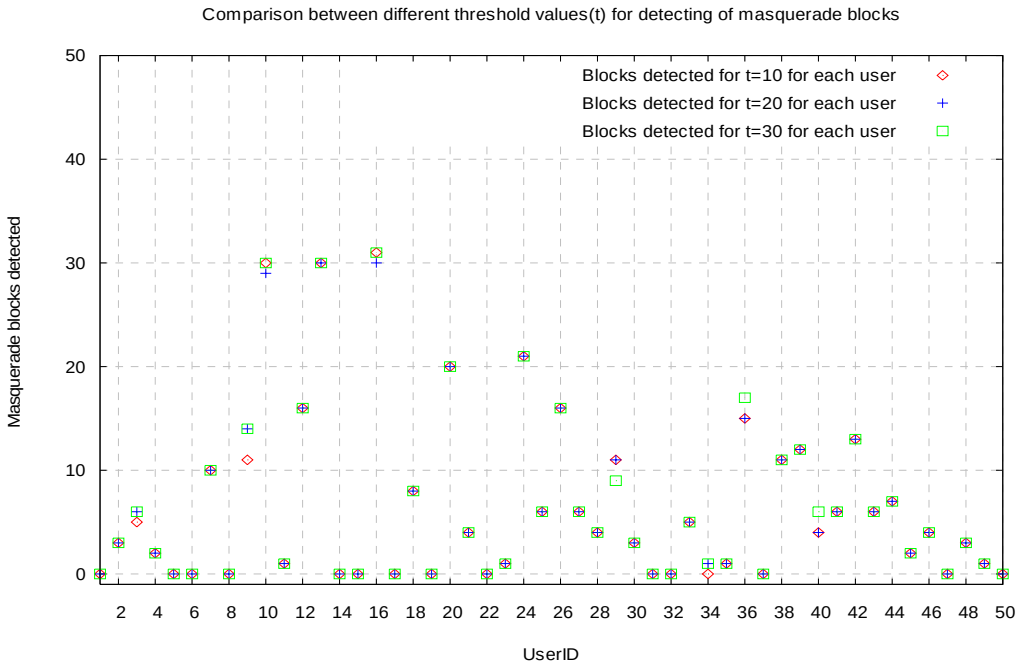

Figure 5.3 Comparison of detection of masquerades in each block for different (t)

5.5 Masquerade User Detection Depending on the Threshold for Frequency of Commands

We measured the similarity between the commands using the unique frequency of commands in the blocks. We defined the threshold for the frequency to limit the number of hits and reduce the false alarms in our experiment. Even a little variation in threshold value gave us substantial variations in detection rates. When a user has more than 10 blocks in his data, we predict the user to be a masquerader.

Based on the threshold value and the frequency of the commands (most common commands occurring), we can clearly distinguish which users are masqueraders. It was determined that the threshold value of 10 produced the most accurate detection rate. The results support the conclusion that the threshold value plays a major role in determining masquerade users, as the frequency of the commands are sensitive to the threshold value. This would lead to the conclusion that the detection rate decreases with increase in the threshold value.

5.7 Error Analysis

The results of the experiment are 60.17 percent hits, 39.82 percent misses with a false alarm rate of 3.86 percent, making the experiment result roughly equivalent, in terms of classification accuracy, to that of the SEA and Maxion experiments. The detailed breakdown of the results of our experiment, on a user-by-user and block-by-block basis are shown in Figures 5.5 through 5.7.

Figure 5.5 illustrates the user-by-user comparison of our approach with the SEA approach for masquerade detection. From the graph, we can clearly see that our approach is competitive in detecting masquerades. Error analysis provides us an idea of whether we are able to detect the masquerades more accurately. From the graphs, we can see that detection by our approach is similar to the SEA method, but not better.

Figures 5.6 and 5.7 illustrates the user-by-user and block-by-block comparisons of our approach with the SEA approach for masquerade detection, respectively.

From the these graphs, we can see that an error analysis provides us with a better picture of the detection process. It clearly explains that though for certain users we have achieved the same results as that of SEA, we need further work in order to better the performance of the classifier.

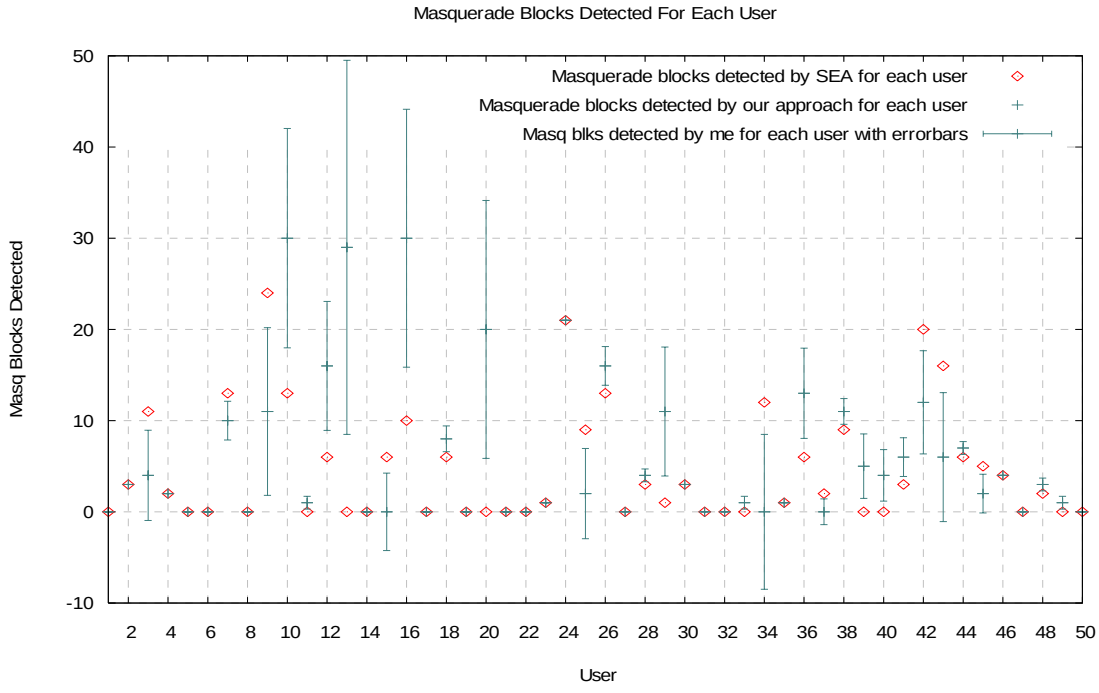

Figure 5.4. Detection of masquerade blocks with error analysis

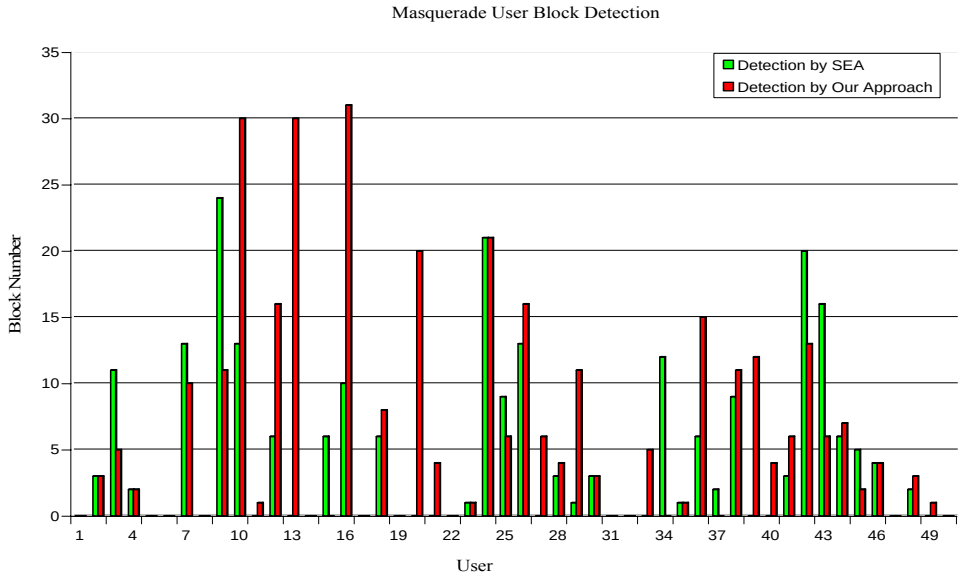

Figure 5.5. Comparisons of the masquerade block and user detection by our approach and the SEA approach.

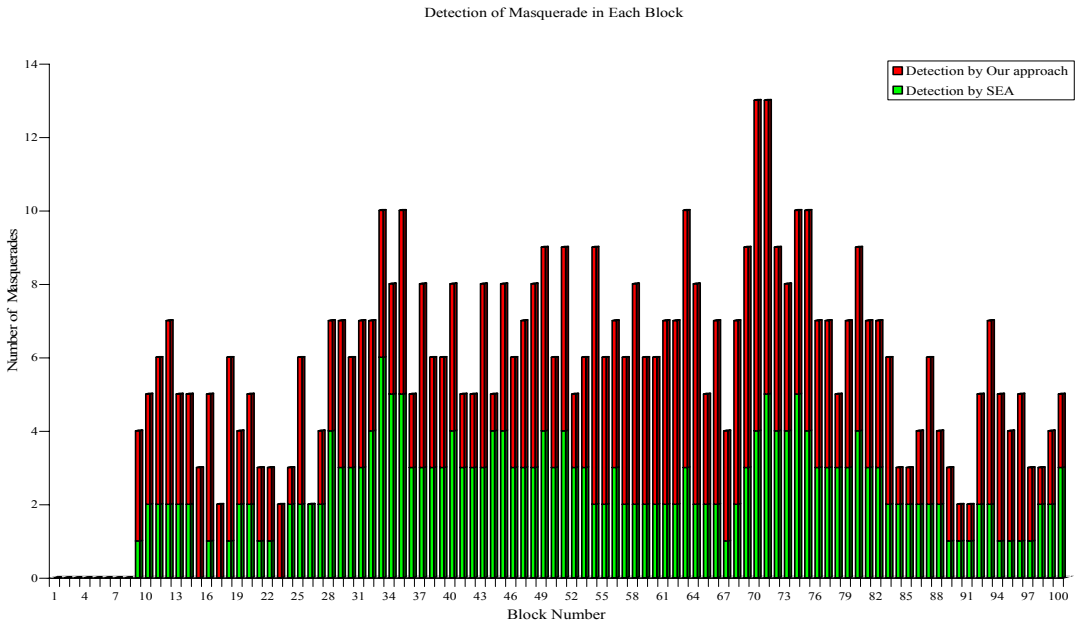

Figure 5.6. Comparison of the number of masquerades detected in each block by our approach and the SEA approach

6. Conclusion

This project used a simple naïve Bayes implementation to detect masquerade attacks. The performance of the algorithm was on par with the performance of top masquerade detection algorithms. In addition, the influence of command frequency and threshold rate on classification was studied to explain how some patterns help in detecting potential masquerades.

Masquerade detection is a very difficult problem. Unfortunately, our approach was not entirely successful. The two reasons for this are the nature of the available data and the problem itself. Data used in the experiment is from the Schonlau dataset (2000), and is only good as a training example. Real world data on masquerade attacks is not usually available for analysis. So, no one can construct an accurate dataset. We can model data that accurately reflects self and non-self, but not what constitutes a masquerader. This absence of data forces researchers to simulate masquerade attacks by injecting non-self data which reflects the everyday actions of a normal user and is not malicious. Additionally, this dataset does not consider command arguments in classification, making detection much harder and inaccurate.

We tested the naïve Bayes algorithm based on command frequency and used a ROC curve to describe the relationship between false positives (normal activity flagged as masquerades) and false negatives (masquerades not detected) at multiple threshold levels. The evaluations suggest that our approach is feasible in detecting masquerades. We gained a 16 percent decrease in false positive rate from 4.6 to 3.86 percent, our detection (hit) rate, however, decreased by 8 percent from 66.2 to 60.17 percent.

We also put forward that sequences of 2 commands can be a viable alternative to using individual commands for classification. On examining the relationship between command sequences (command pairs) and successful masquerade attack, we find that the more common sequences of commands (such as greater than 2 command sequences) in the classification pattern the greater the chance of a successful masquerade attack (Maxion, 2003, Greenberg, Head et al. 1990, and Chen et al. 2004). Advantages to using command sequences include the ability to produce better patterns to recognize user profiles and improved accuracy. We believe that using command sequences greater than two may further improve user profile patterns and accuracy of detection.

7. Future Research

Work done on this project has suggested several avenues of future research into command line-based feature selection for masquerade attack detection. We would like to investigate why some command features performed much better than others. In addition, we noticed that some masquerade activities are more likely to avoid detection while some normal activities are more likely to be mislabeled as an attack. Understanding why this is would advance our research significantly.

Another avenue for future research is to use commands with arguments to improve the accuracy of detection (Wang and Stolfo, 2003). Further, by varying the parameters and metrics described in the experimental design section, we hope to find the threshold for optimal detection using an enriched dataset (Wang and Stolfo, 2003). We also would like to evaluate feature selection to find the most useful feature while eliminating features that have no bearing on the detection of masquerades.

Error evaluation and concept drift (change of user behavior over time) are major considerations that were not explored in this experiment. Error analysis can provide a basis for future improvement (Maxion, 2003). While concept drift renders the naïve Bayes user profiles outdated, decreasing detection rate and increasing false positive hits. These are two issues we would like to consider while building a detection system.

The main focus of this experiment has been anomaly detection. In future, we would like to include other approaches such as bioinformatics, data mining, and other machine learning methods in parallel to increase accuracy and decrease false alarm rates.

References

Caragea, D. Learning Classifiers from Distributed, Semantically Heterogeneous, Autonomous Data. Doctoral Dissertation, Iowa State University, 2004, http://www.cs.iastate.edu/~dcaragea/thesis.pdf

Chen, Y.R., Åström, M., Wang, L.H. Session Comparison Measurement and Learning in Masquerading Detection, Technical report, Lulea Tekniska University, 2004. http://epubl.luth.se/1402-1536/2004/14/LTU-TR-0414-SE.pdf

Coull, S., Branch, J., Szymanski, B., and Breimer, E. Intrusion detection: A bioinformatics approach, In *19th Annual Computer Security Applications Conference*, 2003, 24-33.

Davison, B.D. and Hirsh, H. Predicting sequences of user actions. Presented at the *AAAI-98/ICML'98 Workshop on Predicting the Future: AI Approaches to Time Series Analysis*, 1998. Published in Predicting the Future: AI Approaches to Time Series Problems, Technical Report WS-98-07, pp. 5-12, *AAAI Press*.

DuMouchel, W. and Schonlau, M. A fast computer intrusion detection algorithm based on hypothesis testing of command transition probabilities, Statistic. Sci. 16, no. 1 (2001), 58–74

Greenberg Dataset, http://pages.cpsc.ucalgary.ca/~saul/

Heady, R., Luger, G., Maccabe, A., and Servilla, M. The Architecture of a Network Level Intrusion Detection System. Technical Report CS90-20, University of New Mexico, Department of Computer Science, August 1990.

'ISO/IEC DTR 15947, information technology---Security techniques—IT intrusion detection framework', ISO/IEC JTCI /SC 27 N2691.

Ju, W., Vardi, Y.(1999), A Hybrid High-order Markov Chain Model for Computer Intrusion Detection. National Institute of Statistical Sciences, Report No. 92.

Kwong H. Y. (2004) Using Self-Consistent Naïve -Bayes to Detect Masquerades. *PAKDD,* 329-340.

Lane, T. and Brodley, C.E.. Sequence matching and learning in anomaly detection for computer security. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, 1997, 43—49

Lee, W. and Stolfo, S. Data mining approaches for intrusion detection. In *Proc. 7th USENIX Security Symposium*, 1998.

Lunt, T. F. A survey of intrusion detection techniques. *Computers & Security*, 12 (1993) 405-418.

Lunt, T. F. and Jagannathan, R. A prototype real-time intrusion-detection expert system. In *IEEE Symposium on Security and Privacy*, 1988,.

Marin J., Ragsdale D., Surdu J. A hybrid approach to the profile creation and intrusion detection. In *Proceedings of the DARPA Information Survivability Conference and Exposition – DISCEX 2001*, June 2001.Lunt, T.F. Detecting intruders in computer systems. In *Conference on Auditing and Computer Technology*, 1993.

Maxion, R.A. "Masquerade detection using enriched command lines. In *International Conference on Dependable Systems & Networks (DSN-03)*, 2003, 5-14.

Maxion, R.A. and Townsend, T. N, Masquerade detection using truncated Command lines. In *International Conference on Dependable Systems and Networks (DSN-02)*, 2002, 219-228.

McCallum, A. and Nigam, K. A comparison of event models for naïve bayes text classification. In *Learning for Text Categorization*, papers from the 1998 AAAI Workshop, 27 July 1998, Madison, Wisconsin, pages 41–48. Published as AAAI Technical Report WS-98-05, AAAI Press, Menlo Park, California, 1998.

Mitchell, T. M.. *Machine Learning*. McGraw-Hill, Boston, Massachusetts, 1997.

Peng, J. Detect Masqueraders Using UNIX Command Sequences, Lab Report, Center for Automated Learning and Discovery, 2005.

Schonlau Dataset, http://www.schonlau.net

Schonlau, M. and Theus, M. (2000), Detecting masquerades in intrusion detection based on unpopular commands, *Information Processing Letters*, 76, 33-38.

Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y. (2001), Computer intrusion: Detecting masquerades. *Statistical Science*, 2001; 58-74.

Smaha, S.E. Haystack: An intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference*, 1988, -44,.

Sundaram, A. *An Introduction to Intrusion Detection* [online]. URL: http://www.acm.org/crossroads/xrds2-4/xrds2-4.html (1996).

Szymanski, B. and YZhang, Y.Q. Recursive data mining for masquerade detection and author identification, In *Proc. 5th IEEE System, Man and Cybernetics Information Assurance Workshop*, 2004, 424-431.

Wang, K. and Stolfo, S.J. "One-class training for masquerade detection. In *3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security*, 2003.

Ye, N., Li, X.Y., Chen, Q., Emran, S.M., and Xu, M.M. Probabilistic techniques for intrusion detection based on computer audit data. *In press, IEEE Transactions on System, Man and Cybernetics.*