J Comput Virol (2007) 3:237–245
DOI 10.1007/s11416-007-0055-z

ORIGINAL PAPER

Detecting masquerades using a combination of Naïve Bayes and weighted RBF approach

Alok Sharma · Kuldip K. Paliwal

Received: 3 January 2007 / Revised: 18 April 2007 / Accepted: 1 May 2007 / Published online: 6 July 2007

Abstract Masquerade detection by automated means is gaining widespread interest due to the serious impact of masquerades on computer system or network. Several techniques have been introduced in an effort to minimize up to some extent the risk associated with masquerade attack. In this respect, we have developed a novel technique which comprises of Naïve Bayes approach and weighted radial basis function similarity approach. The proposed scheme exhibits very promising results in comparison with many earlier techniques while experimenting on SEA dataset in detecting masquerades.

1 Introduction

Masquerade detection is one of the challenging tasks in the computer security area. A masquerader is an illegitimate user who impersonates a legitimate user in a computer network. This can be achieved by obtaining the legitimate user's password and accessing unattended programs or workstations. It is quite difficult to detect the masquerader at the beginning of the attack as the user has valid access and privileges. The masquerade attack can also be performed by malicious programs. The threat posed by masquerades can be very serious. Therefore, masquerade detection is an important problem in the area of computer security.

A. Sharma (✉)
School of Engineering and Physics,
University of the South Pacific, Suva, Fiji
e-mail: sharma_al@usp.ac.jf

A. Sharma · K. K. Paliwal
Signal Processing Lab, Griffith University,
Brisbane, QLD 4111, Australia
e-mail: K.Paliwal@griffith.edu.au

Several techniques have been proposed for the purpose of masquerade detection. The performance of any given technique is measured in terms of false positive rate and hit rate. The false positive rate would normally increase when a normal block of data is recognized as masquerade and hit rate increases when masquerade is detected. A successful technique will have high detection rate (hit rate) and low false positive rate. Schonlau etal [13] examined various approaches in this regard namely Uniqueness [14], Bayes one-step Markov [5], Hybrid multistep Markov [6], Compression [13], Sequence-Match [8] and Incremental Probabilistic Action Model (IPAM) [4]. The Uniqueness approach [4,13] is based on the scheme of command frequency whereby commands not previously seen in the training data and/or used by fewer users may indicate a masquerade attempt. The Bayes one-step Markov [5,13] approach is based on one-step transitions from one command to the next and it verifies whether the observed transition probabilities are consistent with the historical probabilities. The Hybrid multistep Markov [6,13] model utilizes multistep Markov chain and simple independence model. When the observed command in testing data is unseen in the training data, a simple independence is employed instead of Markov model. The Compression [13] approach uses compression ratio to distinguish a legitimate user from a masquerader. The Sequence-Match [8,13] computes a similarity measure between the most recent user commands and the user profile. The IPAM [4,13] approach is based on one-step command transition probabilities, estimated from the training data. None of the above mentioned techniques are able to provide very promising results i.e. low false positive rate at high hit rate. Some techniques provide high hit rate but the false positive rate is also high and some techniques provide low false positive rate but their hit rate is also low. In either of the cases the performance is said to be poor.

Next, Maxion and Townsend [10] proposed Naïve Bayes model for detecting masquerades. The underlying assumption of this model is independence of commands. Though this assumption is unrealistic in the case of masquerades, the Naïve Bayes technique still gives better results than the previously mentioned six techniques [4–6,8,13,14]. Recently, Coull etal [2] proposed a technique based on bioinformatics matching algorithm for a semi-global alignment. Wang [17] used one-class training without using examples from other users and demonstrated that it can achieve similar performance as multi-class training. Szymanski and Zhang [16] proposed a recursive data mining method based on model of cognition for masquerade detection including author identification problems. Kim and Cha [7] proposed an efficient technique based on Support Vector Machine (SVM) architecture. The technique depends on the notion of 'common commands' and "voting engine". The common commands are the set of commands used by more than X number of users at the rate exceeding Y percent. The voting engine is used for classifying a set of commands. A block of 100 commands is considered which is sub-divided into smaller blocks. These sub-blocks are then determined to be normal or not using SVM predictor. The 'voting engine' decides if the block is to be considered as being anomalous. Dash etal [3] proposed an episode based Naïve Bayes technique. In this technique meaningful episodes are extracted from a long sequence of commands. These episodes are identified either as masquerade or normal based on Naïve Bayes algorithm.

Masquerade detection problem is basically similar to the anomaly-based intrusion detection problem. The document analysis techniques with k-nearest neighbour ($k.NN$) classifier have been used successfully in the past for anomaly-based intrusion detection [9,11]. In these techniques, the legitimate user is modeled by a reference vector whose components represent relative frequencies of unique commands used by the user and a block of commands is tested against this model (or, reference vector) to detect intruders. This block is represented by a vector of relative frequencies of unique commands occurring within the block. The $k.NN$ approach categorizes the test block (or vector) as either normal or abnormal depending upon the similarity score. A number of similarity measures [9,11,15] have been proposed in this context to measure similarity between the test vector and the reference vector. These similarity measures have been found to be quite useful in detecting anomalies (intruders). This motivated us to investigate these text-processing techniques for masquerade detection. In this paper we apply these similarity measures from the intrusion detection literature [9,11,15] for masquerade detection and found that they give poor results. However, when we combine these techniques with Naïve Bayes approach, we get promising results. We also propose a radial basis function (RBF) based similarity measure and combine it with the Naïve Bayes approach. This improves the masquerade detection performance further. We also show that the performance of our approach is better than previously used methods such as the SVM based technique [7] and the episode based Naïve Bayes technique [3].

Table 1 Examples of Unix *acct* auditing command

User 2 command name	Command number
man	94
sed	95
awk	96
sh	97
awk	98
awk	99
less	100

The paper is organized as follows: Sect. 2 depicts the setup and database used for the proposed method. Section 3 illustrates the Naïve Bayes classifier for masquerade detection. Section 4 describes RBF similarity measure for the problem. Section 5 presents our overall proposed scheme. Section 6 deals with the experimentation and results of the techniques on truncated command dataset and Sect. 7 presents our concluding remarks.

2 Database and experimental setup

Schonlau etal [13] collected UNIX *acct* auditing command to implement truncated command dataset (so called SEA dataset) for masquerade detection. Examples of some commands are given in Table 1. The dataset consists of 15,000 truncated commands for each user. The 15,000 commands of the dataset are decomposed into 150 blocks of 100 commands each for each user. The first 50 blocks for each user are considered to be normal or genuine data with no malicious commands and are used as training data. For testing purpose the next 100 blocks are used which also belong to each of these 50 users, but data from 20 different users which is considered to be potentially contaminated, is mixed with the data of 50 users. Therefore these 20 users are considered as intruders and the system should identify or classify their entries as masquerade data.

Many methods discussed in this paper have been analyzed for masquerade detection using SEA dataset. The performance of previously discussed methods on SEA dataset is summarized in Table 2.

3 The Naïve Bayes classifier for masquerade detection

In this section we briefly describe the Naïve Bayes classification algorithm for masquerade detection [10]. This technique

Table 2 Hit rate versus false positive rate for previous methods on SEA dataset (where False positive (%) = $\frac{\text{number of normal blocks detected as masquerade}}{\text{Total number of normal blocks}} \times 100$ and Hit (%) = $\frac{\text{number of masquerade blocks detected as masquerade}}{\text{Total number of masquerade blocks}} \times 100$)

Method	Hit rate (%)	False positive rate (%)
Episode based Naïve Bayes [3]	88.3	14.5
SVM [7]	80.1	9.7
Semi-global alignment [2]	75.8	7.7
Recursive data mining [16]	75.0	10.0
Bayes one-step Markov [5,13]	69.3	6.7
Naïve Bayes (no updating)[a] [10]	66.2	4.6
Hybrid Markov [6,13]	49.3	3.2
IPAM [4,13]	41.1	2.7
Uniqueness [13,14]	39.4	1.4
Sequence matching [8,13]	36.8	3.7
Compression [13]	34.2	5.0

[a] The results for Naïve Bayes with updating criteria has not been included in the study because the updating procedure is an additional portion which requires more than usual training data at increased complexity and processing time. Moreover, the proposed scheme comprises of Naïve Bayes algorithm which is experimented on no updating criteria. Therefore it would be sensible to compare methods based on the same test schemes.

is a basic statistical method used widely in the pattern classification problems. Some of the advantages of Naïve Bayes classifier are fast learning rate and robustness to noise. This classifier considers commands to be statistically independent. That is the command sequence is generated by the user in a way that each command has a fixed probability. In other words subsequent command does not depend on the current command. The classifier only takes into account the frequency of commands and not the order of their appearance. See [10] for details.

This property of independence is, however, unrealistic for command-based masquerade detection, but is quite successful in the masquerade detection [10]. To illustrate the method let C be the set of unique commands and $c \in C$ be any arbitrary command. Let the set of commands in the training phase used by user $u_j \in U$ be C_j where U is a set of all users and a_j be frequency vector extracted from C_j where the entries of a_j denote the number of times command c appeared in C_j. Therefore the dimension or length of a_j is same as the length of set C and the sum of entries of a_j is equal to the length of C_j.

The probability of command c for a given user u_j is given by:

$$P_{c,u_j} = \frac{a_{k,j} + \alpha}{|C_j| + \alpha|C|} \tag{1}$$

where α is a pseudocount [10] (0.01 in this study), $|C|$ is the length of C, $|C_j|$ is the length of C_j i.e. the length of training data and $a_{k,j}$ is the k^{th} entry of a_j which corresponds to the command c. The sum of probabilities for all the commands in C_j will be unity. The pseudocount has been added in Eq. 1 to ensure that there are no zero counts. The lower the pseudocount, the more sensitive the detector is to previous unseen commands [10]. We have adopted the value of pseudocount similar to the value of pseudocount used by Maxion and Townsend [10] to properly compare the proposed technique with the Naïve Bayes technique.

In a similar fashion as of Eq. 1 we can define the probability of command c for all U users except the user u_j as [10]

$$P_{c,u_{\neq j}} = \frac{\sum_{i=1,i\neq j}^{|U|} a_{k,i} + \alpha}{\sum_{i=1,i\neq j}^{|U|} |C_j| + \alpha|C|} \tag{2}$$

where $|U|$ is the number of users in the training phase. This $P_{c,u_{\neq j}}$ can simply be viewed as NOT P_{c,u_j}.

The probability of a block or sequence of commands B is based on the probabilities obtained during the training phase. Let us define the probability of block B for user u_j as

$$P_{B,u_j} = \prod_{c\in B} P_{c,u_j}. \tag{3}$$

We then define self and non-self probabilities using Equation 3 as

$$P_{self}(B) = P_{B,u_j} \tag{4}$$

$$P_{non-self}(B) = P_{B,\neq u_j} \tag{5}$$

The self probability is the probability of commands in block B entered by user u_j whereas the non-self probability indicates the probability of commands in block B entered by all the users except the user u_j. The ratio of self probability and non-self probability can be given using Eqs. 4 and 5 as [3,10]

$$R = \frac{P_{self}(B)}{P_{non-self}(B)}.$$

The ratio R is the division of self and non-self probabilities. These probabilities are the product of command probabilities and their values become very small when the command probabilities are multiplied among themselves. In some cases it is beyond the precision provided by floating point representation of numbers using normal computers. One way to avoid this precision problem is to represent the ratio in terms of logarithms as

$$\gamma = \log P_{self}(B) - \log P_{non-self}(B) \tag{6}$$

If the ratio γ (Eq. 6) is below the threshold then the block is considered to be masquerade block otherwise it is considered to be normal or legitimate block. The threshold in this paper is progressively varied to obtain the receiver operating characteristics (ROC) curve. The threshold determines the trade off between the hit rate and false positive rate of dataset. It can be evaluated by implementing a criterion on, for example, false positive rate i.e. finding a value of threshold for which the false positive rate is less than some defined value. The threshold can also be found by methods described in [10] and [13].

The following subsection illustrates the Naïve-Bayes classifier for masquerade detection using a toy example.

3.1 An illustration using a toy example

To illustrate the classifier let us assume a unique set of commands given as

$$C = \{\text{ls, chmod, awk, cat}\}.$$

Suppose there are three users u_1, u_2 and u_3. Each user enters 20 commands during the training session. The commands entered by each user are depicted as follows:

$$\begin{aligned} C_1 = \{&\text{chmod, chmod, awk, ls, awk, ls, ls, awk, awk, chmod,} \\ &\text{ls, chmod, ls, awk, ls, ls, awk, ls, chmod, ls}\}, \\ C_2 = \{&\text{ls, awk, cat, ls, awk, cat, cat, ls, awk, ls, ls, ls, awk,} \\ &\text{cat, cat, ls, cat, ls, cat, awk}\}, \\ C_3 = \{&\text{ls, ls, cat, chmod, ls, ls, awk, cat, chmod, ls, ls, cat,} \\ &\text{cat, cat, chmod, awk, awk, ls, cat, cat}\}. \end{aligned}$$

It is clear that the length of C is 4 and the lengths of C_1, C_2 and C_3 are 20. It can also be observed that the elements of C_j (where $j = 1, 2, 3$) is from the unique set C. The set of frequency vectors can now be defined as

$$\begin{aligned} a_1 &= \{9, 5, 6, 0\}, \\ a_2 &= \{8, 0, 5, 7\}, \\ a_3 &= \{7, 3, 3, 7\} \end{aligned}$$

where the frequency vector a_1 corresponds to set C_1, a_2 corresponds to C_2 and a_3 corresponds to C_3. The entries of a_j (where $j = 1, 2, 3$) represent the number of times commands are entered by the users u_j (where $j = 1, 2, 3$). The length of $a_j$is 4 and the sum of entries of a_j is 20. That means the length of a_j is equal to the length of C and the sum of entries of a_j is equal to the length of C_j.

Let us also assume that each of the sets C_1, C_2 and C_3 can be subdivided into 4 blocks each consisting of 5 commands. This means that the size of each block is 5. This subdividing of C_j into blocks, however, will not affect the evaluation of parameters during the training phase but it will indicate that the block during the testing session should be considered as a group of 5 commands.

It is now possible to evaluate the probability of command c for a given user u_j. The probabilities of commands entered by user u_1 (Eq. 1) can be given as

$$\begin{aligned} P_{\text{ls},u_1} &= \frac{9 + 0.01}{20 + 0.01 \times 4} = 0.4496; \\ P_{\text{chmod},u_1} &= \frac{5 + 0.01}{20 + 0.01 \times 4} = 0.2500, \\ P_{\text{awk},u_1} &= \frac{6 + 0.01}{20 + 0.01 \times 4} = 0.2999; \\ P_{\text{cat},u_1} &= \frac{0 + 0.01}{20 + 0.01 \times 4} \approx 0.0005. \end{aligned}$$

In a similar fashion the probabilities of commands for users u_2 and u_3 can be obtained as

$$\begin{aligned} &P_{\text{ls},u_2} = 0.3997,\ P_{\text{chmod},u_2} = 0.0005, \\ &P_{\text{awk},u_2} = 0.2500,\ P_{\text{cat},u_2} = 0.3498, \\ &P_{\text{ls},u_3} = 0.3498,\ P_{\text{chmod},u_3} = 0.1502, \\ &P_{\text{awk},u_3} = 0.1502,\ P_{\text{cat},u_3} = 0.3498. \end{aligned}$$

The NOT probabilities for user u_1 can be obtained using Eq. 2 as follows:

$$\begin{aligned} P_{\text{ls},u_{\neq 1}} &= \frac{(8 + 7) + 0.01}{(20 + 20) + 0.01 \times 4} = 0.3749; \\ P_{\text{chmod},u_{\neq 1}} &= \frac{(0 + 3) + 0.01}{(20 + 20) + 0.01 \times 4} = 0.0752, \\ P_{\text{a}wk,u_{\neq 1}} &= \frac{(5 + 3) + 0.01}{(20 + 20) + 0.01 \times 4} = 0.2000; \\ P_{\text{cat},u_{\neq 1}} &= \frac{(7 + 7) + 0.01}{(20 + 20) + 0.01 \times 4} = 0.3499. \end{aligned}$$

The NOT probabilities for users u_2 and u_3 can be evaluated in a similar fashion as above and the results are shown as below:

$$\begin{aligned} &P_{\text{ls},u_{\neq 2}} = 0.3999,\ P_{\text{chmod},u_{\neq 2}} = 0.2000, \\ &P_{\text{awk},u_{\neq 2}} = 0.2250,\ P_{\text{cat},u_{\neq 2}} = 0.1751, \\ &P_{\text{ls},u_{\neq 3}} = 0.4248,\ P_{\text{chmod},u_{\neq 3}} = 0.1251, \\ &P_{\text{awk},u_{\neq 3}} = 0.2750,\ P_{\text{cat},u_{\neq 3}} = 0.1751. \end{aligned}$$

In the training phase probabilities are computed. Now in the testing phase let us consider a block of 5 commands given as

$$B = \{\text{ls, ls, awk, cat, cat}\}.$$

If this block is tested against user $u_1$then the self and non-self probabilities can be evaluated using Eqs. 3, 4 and 5 as follows:

$$\begin{aligned} P_{\text{self}}(B) = P_{B,u_1} &= 0.4496 \times 0.4496 \times 0.2999 \\ &\quad \times 0.0005 \times 0.0005, \end{aligned}$$

$$P_{\text{non-self}}(B) = P_{B,\neq u_1} = 0.3749 \times 0.3749 \times 0.2000 \times 0.3499 \times 0.3499.$$

The self and non-self probabilities will give the ratio γ (Eq. 6) as follows:

$$\gamma = -18.0049 - (-5.6718) = -12.3331.$$

If the block B is tested against users u_2 and u_3 then corresponding values of the ratio γ will be 1.4884 and 0.3907 respectively. If we set the threshold to 0 then the block will be labeled as masquerade block for user u_1 (since $-12.3331 < 0$). It can be also inferred from set C_1 that some commands (e.g. "*cat*") that are repeatedly used in block B were not even used by user u_1 during the training session. This has contributed small γ ratio for user u_1 in comparison with other users. Clearly, such block could cause suspicion, in particular for user u_1 since the trend of commands entered by user u_1 were not familiar with the history of commands of the same user. The obtained ratio γ is highest for user u_2 which suggests that the tested block is most likely a legitimate block for user u_2. It can also be deduced from the illustration that there is high likeliness that small values of ratio γ tends to be masquerade block.

4 Weighted RBF similarity measure

The proposed weighted RBF similarity measure falls under the similarity measure techniques in the area of anomaly-based detection [15]. The weighted RBF similarity can be considered as the combination of binary similarity measure [11] and RBF similarity measure. To illustrate the similarity measure let v be any frequency vector of dimension d extracted from a test block or a sequence of commands. This vector v can be easily transformed to its binary representation vb of dimension d. The entries of vb is vb_i where $vb_i = 1$, if the i^{th} command is present in v, otherwise $vb_i = 0$. Similarly we can define binary representation ab_j of a_j (frequency vector of training data by user u_j). Then the binary similarity measure $\mu(vb, ab_j)$ between vb and ab_j can be defined as follows:

$$\mu(vb, ab_j) = \frac{\text{match}}{\text{match} + \text{non-match}}$$

where *match* and *non-match* are the sum of matching and non-matching entries between vb and ab_j and can be obtained by the following "and" and "xor" (exclusive-or) binary operations:

$$\text{match} = \sum_d \text{and}(vb, ab_j) \tag{7}$$

and

$$\text{non-match} = \sum_d \text{xor}(vb, ab_j) \tag{8}$$

The sum of *match* (Eq. 7) and *non-match* (Eq. 8) can simply be reduced to the sum of binary "or" operation as

$$\text{match+non-match} = \sum_d \text{or}(vb, ab_j).$$

On the other hand, RBF functions [1] provide a similarity score between the input frequency vector v of dimension d and frequency vector a_j from the training dataset. The RBF similarity measure can be considered as an implicit measure of similarity between v and a_j in some dot product space F by mapping the frequency vectors using a mapping function ϕ in space F. This dot product which is implicitly embedded with RBF is an appealing mathematical characteristic for similarity measurement. One popular choice of RBF function is Gaussian radial basis function and is defined as $k(v, a_j) = \exp(-\frac{1}{2}||v - a_j||^2/||a_j||^2)$. It can be noticed that the function $k$is an unsymmetrical relation. This unsymmetrical similarity measurement, however, assists in normalizing the distance between the two vectors with the training vector or reference vector. This makes the measurement less susceptible to the input frequency vectors since the input frequency vectors are extracted from a small block of commands whereas the training vectors are extracted from a large set of commands in the masquerade detection. The symmetrical similarity measurement is also considered in the experimentation and as was expected it is showing inferior results as compared to the unsymmetrical similarity measurement. The symmetrical similarity measure is defined as

$$k(v, a_j) = \exp\left(-\frac{1}{2}\left[\frac{||v-a_j||^2}{||a_j||^2} + \frac{||v-a_j||^2}{||v||^2}\right]\right).$$

We can now define our weighted RBF similarity measure as follows:

$$\lambda(v, a_j) = \mu(vb, ab_j)\, k(v, a_j) \tag{9}$$

The weighted RBF similarity takes into account the similarity measure based on the frequency of commands (due to $k(v, a_j)$) and the weight associated with the frequency vectors (due to$\mu(vb, ab_j)$). Therefore the proposed $\lambda(v, a_j)$ measure embeds more information than individual measures which is a more appropriate way of seeking some score or similarity between any two blocks or set of commands.

The other similarity measures like cosine metric [9] and binary-weighted cosine metric [11] from the intrusion detection area are also experimented on masquerade detection but not very promising results are obtained. We therefore excluded the discussions for these measures in this paper, however, the interested readers may wish to check the references [9,11] and the references given therein.

5 Combined weighted RBF-Naive Bayes classifier for masquerade detection

In this section we combined the Naïve Bayes algorithm and weighted RBF similarity measure. In order to combine the scores from both the techniques first we look at the ratio γ from Naïve Bayes algorithm. It can be observed from Eq. 6 that when the ratio γ is low there is a high likeliness for a test block to be masquerade. It can also be observed that the values of γ can be either positive or negative in the vicinity of real numbers. This range of γ creates a problem while tying it with the weighted RBF similarity measure since $\lambda(v, a_j)$ is a similarity measure with positive values only. For instance, if a high negative valued γ (possible masquerade block) is tied with high valued $\lambda(v, a_j)$ (possible legitimate block) in the form of multiplication then the resulting product would be highly negative (possible masquerade). On the other hand if a high positive valued γ (possible legitimate block) is multiplied with high valued $\lambda(v, a_j)$ (possible legitimate block) then the resulting block could be legitimate. It is obvious that negative valued γ tied with $\lambda(v, a_j)$ would lead to an erroneous result. To avoid this problem we propose the γ ratio as

$$\gamma = \frac{\log P_{self}(B)}{\log P_{non-self}(B)} = \frac{\sum_{c \in B} \log P_{c,u_j}}{\sum_{c \in B} \log P_{c,\neq u_j}}, \tag{10}$$

which will be always positive. Also note, now the block will be considered a masquerade block if the ratio γ is above the threshold and it will be considered a legitimate block when the ratio is below the threshold. Therefore the ratio γ can be viewed as a dissimilarity measure.

Conversely, for weighted RBF similarity measure the high value of $\lambda(v, a_j)$ represents more similarity between blocks which would lead to label the test block as normal. It is also obvious from Eq. 9 that the value of $\lambda(v, a_j)$ lies between 0 and 1. This means a similarity measure $\lambda(v, a_j)$ can be made a dissimilarity measure by subtracting it with unity. This helps in combining $\lambda(v, a_j)$ and γ as

$$r = \gamma(1 - \lambda(v, a_j)) \tag{11}$$

For representation simplicity we exclude syntaxes (v and a_j) from the ratio r. The high value of r would indicate dissimilarity between the undertaken blocks and low value would signify similarity. If r is greater than threshold then the block will be considered as masquerade otherwise as normal.

The motive behind combining the approaches is to possibly include more information about commands than is possible with individual techniques. The weighted RBF approach encompasses information based on frequency of commands and weights associated with frequency vectors. On the other hand, the Naïve Bayes algorithm includes information related to the probabilities of commands entered by one user over the other users. Furthermore, the individual algorithms may have their own local regions where they perform the best. It is therefore sensible to combine them.

The training and testing phases can be illustrated as follows:

- **Training phase**
  - Compute frequency of commands and its binary representation (ab_j) of each user u_j.
  - Compute the probabilities P_{c,u_j} (Eq. 1) and $P_{c,u_{\neq j}}$ (Eq. 2).
- **Testing phase**
  Compute the ratio of self and non-self probabilities (γ from the Eq. 10) and RBF similarity measure ($\lambda(v, a_j)$) for each block B of testing dataset of user u_j. The values of γ and $\lambda(v, a_j)$ yields combined ratio r (Eq. 11). The block B will be determined as legitimate or masquerade depending upon the following condition:

$$block(B) = \begin{cases} \text{masquerade}, & \text{if } r \geq \theta \\ \text{normal}, & \text{otherwise} \end{cases}$$

  where, θ is a threshold. For brevity we call this proposed approach weighted RBF-Naïve Bayes (WRBF-NB) algorithm.

5.1 An illustration using weighted RBF-Naïve Bayes classifier

This section illustrates weighted RBF-Naïve Bayes approach using the example provided in Sect. 3.1. The ratio γ obtained for the three users u_1, u_2 and u_3 against block B in Sect. 3.1 were −12.3331, 1.4884 and 0.3907 respectively. It is clear that the range of γ is the entire real axis (i.e. γ can obtain negative and positive values). To restrict the values of γ only in positive real axis equation 10 has been used. The values of γ using Eq. 10 for users u_1, u_2 and u_3 against block B are 3.1744, 0.7814 and 0.9398. Here the high values of γ signifies a possible masquerade block.

The frequency vector of block B can be computed by finding the number of times commands appeared in the block. Thus the frequency vector will be $v = \{2, 0, 1, 2\}$. The binary representation of v and a_j can be given as

$vb = \{1, 0, 1, 1\}$; $ab_1 = \{1, 1, 10\}$;
$ab_2 = \{1, 0, 1, 1\}$; $ab_3 = \{1, 1, 1, 1\}$.

The binary similarity measure for the tested block against the commands of user u_1 during the training phase can be evaluated as follows:

$$\mu(vb, ab_1) = \frac{2}{2+2} = 0.5.$$

The weighted RBF similarity measure can also be computed for user u_1 using Eq. 9 as follows:

$$\lambda(v, a_1) = \mu(vb, ab_1)k(v, a_1) = 0.3479.$$

It is now possible to find dissimilarity measure r using Eq. 11 as follows:

$$r = \gamma(1 - \lambda(v, a_1)) = 3.1752(1 - 0.3479) = 2.0705.$$

Similarly, the dissimilarity measure r for users u_2 and u_3 can be found in the following respective manner:

$$\begin{aligned} r &= \gamma(1 - \lambda(v, a_2)) = 0.7814(1 - 0.7566) \\ &= 0.1902(\text{ for user } u_2), \\ r &= \gamma(1 - \lambda(v, a_3)) = 0.9397(1 - 0.5716) \\ &= 0.4025(\text{ for user } u_3). \end{aligned}$$

It is clear that $r \geq 0$. It is also evident that the higher the value of r the greater the dissimilarity between the train and test commands. The difference between the values of r for user u_2 and user u_3 against block B is not very significant, this implies that the considered block is not behaving in an abnormal way for these two users. If a threshold is fixed somewhere around 0.5 then block B will be labeled as masquerade for user u_1. The threshold value will determine the false positive rate and its corresponding hit rate values. The threshold value can be fixed such that the system will give false positive rate less than or equal to some defined value. In this way the obtained hit rate has the corresponding false positive rate in the predefined vicinity or range. It can also be noted that the difference in γ ratio of user u_2 and user u_3 in Sect. 3.1 is significant as compared with the difference in ratio r of the same two users. This signifies that the trend of commands in block B for user u_2 and user u_3 is not very distinguishable. This is also the case when careful observation of commands of block B and commands of user u_2 and user u_3 have been made. In other words, the proposed technique is providing better insight about the commands. This fact will be apparent when the techniques will be experimented on SEA dataset in the following section.

6 Experimentation on truncated command dataset

The experimentation part is subdivided into four phases. In the first phase we show the ROC curve for the unsymmetrical weighted RBF measure for masquerade detection. Then we show a comparison of ROC curves between symmetrical WRBF-NB and Naïve Bayes algorithms. Then experiment for the proposed algorithm is conducted and compared with all the techniques discussed in this paper. Next we compare the ROC curve for the proposed unsymmetrical WRBF-NB or simply WRBF-NB technique and the Naïve Bayes technique. All the experiments were conducted using SEA dataset [13] where a set of 100 commands is determined either to be masquerade or normal.

Figure 1 illustrates the ROC curve for the unsymmetrical weighted RBF measure for masquerade detection. It can be observed from the figure that the hit rate increases as the false rate is increased. Although the results obtained is not very challenging when compared with other masquerade detection techniques, it gives an insight that the measure embeds some information which could be useful for masquerade detection. A comparison between symmetrical WRBF-NB and Naïve Bayes algorithms is depicted in Fig. 2. It can be observed from the figure that symmetrical WRBF-NB is performing better than Naïve Bayes algorithm. Although the improvement here is not very significant but it indicates that there is some useful information in symmetrical weighted RBF technique for masquerade detection when combined with the Naïve Bayes algorithm. Next, the experiment for the proposed WRBF-NB algorithm is conducted. Different values of $\theta$are used to obtain the ROC curve for the algorithm as shown in Fig. 3. It can be observed from the figure that the proposed combination of Naïve Bayes and weighted RBF approach produces very promising results. To illustrate a few, the episode based Naïve Bayes approach produces hit rate of 88.3% at false positive rate of 14.5%. The WRBF-NB technique at the same false positive rate produces 92.2% hit rate which is an improvement of 3.9% over the episode based technique. The SVM technique produces 80.1% hit rate at 9.7% false positive rate. Here the WRBF-NB technique produces 87.9% hit rate at the same 9.7% false positive rate which is an improvement of 7.8% over the SVM technique. Similarly Semi-Global Alignment and Recursive Data Mining produce hit rates 75.8% and 75.0% at false positive rates 7.7% and 10.0% respectively whereas the WRBF-NB technique produces 83.1% and 88.8% at respective false positive rates which are 7.3% and 13.8% improvements over Semi-Global Alignment and Recursive Data Mining techniques. It can also be observed from the ROC curve that the algorithm achieves 84.0% hit rate at 7.8% false positive rate which seems to be one of the best results. The results obtained by the proposed WRBF-NB technique are clearly significant than those of earlier techniques. The results for some previous methods in comparison with WRBF-NB method are summarized in Table 3. Column 1 in the table lists previous methods, column 2 denotes false positive rate, column 3 denotes the corresponding Hit rate of previous methods, column 4 highlights the corresponding hit rates of WRBF-NB method and column 5 denotes the corresponding false positive

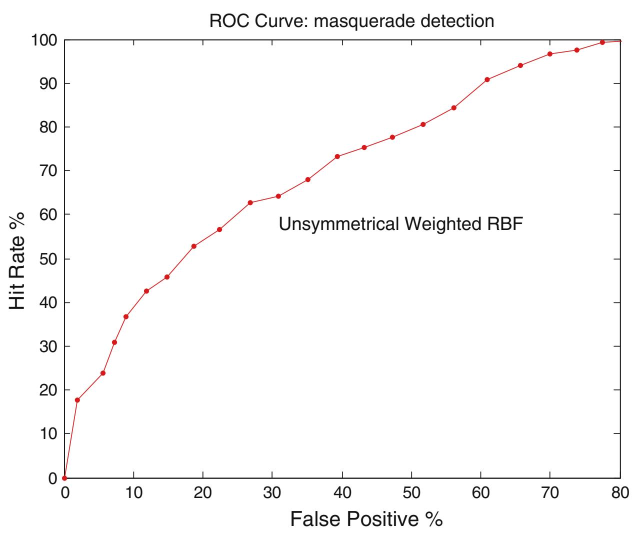

Fig. 1 ROC curve for unsymmetrical weighted RBF measure on masquerade detection

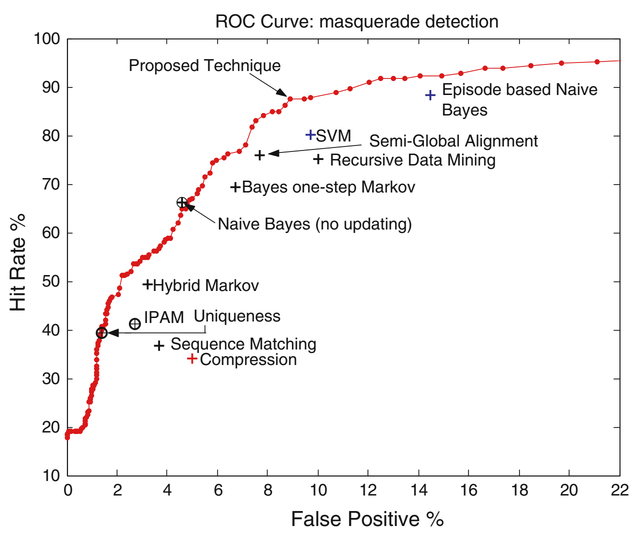

Fig. 3 ROC curve of the weighted RBF – Naïve Bayes algorithm with other presented techniques

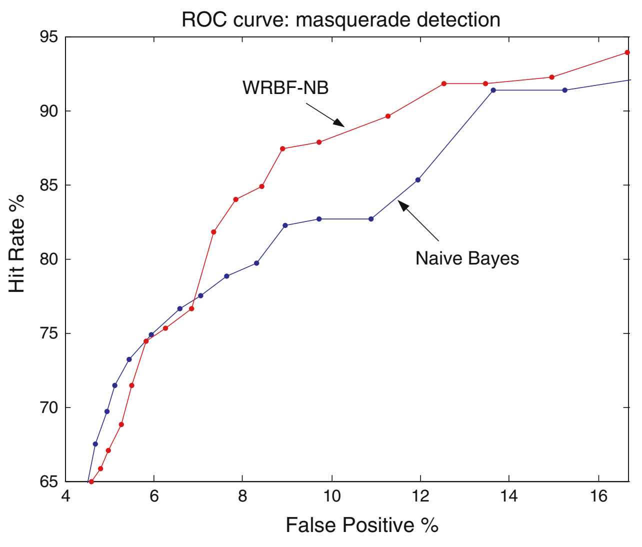

Fig. 4 A comparison of ROC curves for the WRBF-NB and Naïve Bayes algorithms

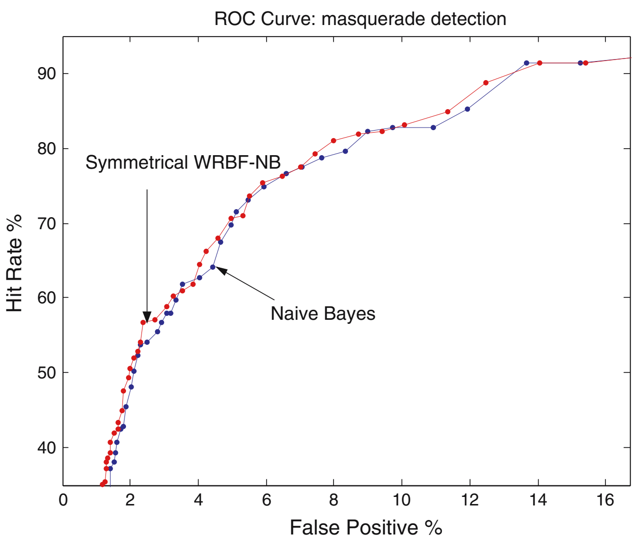

Fig. 2 ROC curves for symmetrical WRBF-NB and Naïve Bayes algorithms

rate of WRBF-NB method. We have also illustrated ROC curves for WRBF-NB and Naïve Bayes techniques for comparison purpose in Fig. 4. It can be observed from the figure that at high hit rates the Naïve Bayes is performing poorly. We are not very interested in low hit rate region as the low hit rate would simply mean to allow more masquerades to enter the system unnoticed. The ROC area of interest is the high hit rate region where the WRBF-NB technique is exhibiting significant improvement. This means that the risk of attack by masquerades using WRBF-NB technique is considerably less than Naïve Bayes technique.

7 Conclusion

In this paper, we presented a technique for masquerade detection based on Naïve Bayes approach and weighted RBF similarity approach. The proposed approach was experimented on SEA dataset and its performance was compared with other masquerade detection techniques. It was observed that the proposed approach performed significantly better than many earlier techniques.

During the review process of this paper we came to know about the paper written by Rieck and Laskov [12] where they proposed an interesting idea for intrusion detection. This idea could be used in masquerade detection. We will investigate

Table 3 Comparison of WRBF-NB with previous methods

Method	False positive rate (%)	Hit rate (%)	Hit rate of WRBF-NB (%)	False positive rate of WRBF-NB (%)
Episode based Naïve Bayes [3]	14.5	88.3	92.2	14.5
SVM [7]	9.7	80.1	87.9	9.7
Semi-global alignment [2]	7.7	75.8	83.1	7.7
Recursive data mining [16]	10.0	75.0	88.8	10.0

this approach for masquerade detection and compare their performance with our approach.

References

1. Aizerman, M., Braverman, E., Rozonoer, L.: Theoretical foundations of the potential function method in pattern recognition learning. Autom. Remote Control **25**, 821–837 (1964)
2. Coull, S., Branch, J., Szymanski, B., Breimer, E.: Intrusion detection: a bioinformatics approach. In: 19th Annual Computer Security Applications Conference, pp. 8–12. Las Vegas, Nevada, (2003)
3. Dash, S.K., Reddy, K.S., Pujari, A.K.: Episode based masquerade detection. Lecture Notes in Computer Science, vol. 3803. Springer, Berlin, pp. 251–262 (2005)
4. Davision, B.D., Hirsh, H.: Predicting sequences of user actions. Predicting the future: AI approaches to time series problems. In: Technical Report WS-98-07, pp. 5–12. AAAI Press (1998)
5. DuMouchel, W.: Computer intrusion detection based on Bayes Factors for comparing command transition probabilities. In: Technical Report 91, National Institute of Statistical Sciences (1999)
6. Ju, W., Vardi, Y.: A hybrid high-order Markov chain model for computer intrusion detection. In: Technical Report 92, National Institute of Statistical Sciences (1999)
7. Kim, H.-S., Cha, S.-D.: Empirical evaluation of SVM-based masquerade detection using UNIX commands. Comput. Secur. **24**, 160–168 (2005)
8. Lane, T., Brodley, C.E.: Approaches to online learning and concept drift for user identification in computer security. In: Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining, pp. 259–263. AAAI Press (1998)
9. Liao, Y., Vemuri, V.R.: Use of K-nearest neighbor classifier for intrusion detection. Comput. Secur. **21**(5), 439–448 (2002)
10. Maxion, R.A., Townsend, T.N.: Masquerade detection using truncated command lines. In: Proceedings of the International Conference on Dependable Systems and Networks, pp. 23–26 (2002)
11. Rawat, S., Gulati, V.P., Pujari, A.K., Vemuri, V.R.: Intrusion Detection Using Text Processing Techniques with a Binary-Weighted Cosine Metric. J. Inf. Assur. Secur. (1), pp. 43–50 (2006)
12. Rieck, K., Laskov, P.: Language models for detection of unknown attacks in network traffic. J. Comput. Virol. **2**(4), 243–256 (2007)
13. Schonlau, M., DuMouchel, W., Ju, W., Karr, A.F., Theus, M., Vardi, Y.: Computer intrusion: detecting masquerades. Stat. Sci. **16**(1), 58–74 (2001)
14. Schonlau, M., Theus, M.: Detecting masquerades in intrusion detection based on unpopular commands. Inf. Process. Lett. **76**, 33–38 (2000)
15. Sharma, A., Pujari, A.K., Paliwal, K.: Kernel Based Metrics for Intrusion Detection Using Text Processing Techniques (under review) (2006)
16. Szymanski, B.K., Zhang, Y.: Recursive Data Mining for Masquerade Detection and Author Identification. In: Proceedings of 5th IEEE System, Man and Cybernetics Information Assurance Workshop, West Point, pp. 424–431. IEEE CS Press, Los Alamitos (2004)
17. Wang, K., Stolfo, S.J.: One-class training for masquerade detection. In: 3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security, Florida, 19 November 2003