Experiments on command dimension reduction in masquerader detection

Carlos Benitez[1] and Pablo Fierens[2]

[1] CITEFA - Instituto de Investigaciones Científicas y Técnicas para la Defensa
San J. B. de La Salle 4397 (B1603ALO),Villa Martelli, Buenos Aires, Argentina
Tel.: 5411-4709-8289, Fax: 5411-4709-5363
cbenitez@citefa.gov.ar
[2] ITBA - Instituto Tecnológico de Buenos Aires
Av. Eduardo Madero 399 (C1106ACD), Buenos Aires, Argentina
Tel.: (5411) 6393-4822
pfierens@itba.edu.ar

Abstract. We deal with the problem of dimension reduction in masquerader detection through command-line behavior. Although it has been previously suggested that unpopular commands are more relevant for this task, it is shown that there is no conclusive evidence in favor of this hypothesis. Moreover, it is also shown that selection of a fraction of the most popular or frequently used commands leads to a smooth degradation of a masquerader detection algorithm, while the selection of the most unpopular or infrequent commands produces a degradation which is worse than that of simple random selection.
Some evidence is provided that the best performance of a masquerader detection algorithm may not necessarily correspond to accounting for all commands in the training data, but for a smaller, adequately chosen fraction of them. We verify this conclusion using two different datasets and two different masquerader detection algorithms.
Finally, the empirical evidence provided in this paper suggests that, for many masquerader detection techniques, it may be convenient to work with a small fraction of the most popular or frequently used commands.

Keywords. masquerader detection, command-line behavior, dimension reduction, computer security.

1 Introduction

In computer intrusion detection, masqueraders are illegitimate users that try to impersonate legitimate ones stealing passwords and using real user accounts. There are different aspects of masquerader behavior which may be taken into account to detect her, e.g., the list of directories she accessed, the programs she executed, the websites she visited, etc. Among all these aspects, the sequence of commands typed by the masquerader has been one of the most popular choices in the literature (see, e.g., [1][2][3][4][5][6][7][8][9] and [10]). The usual approach

has been to monitor the command-line behavior of legitimate users for a long period in order to build a sufficiently large data corpus, and to apply a technique that looks for anomalies when new commands are typed.

As the "normal"-behavior dataset grows, the detection of masqueraders may become more accurate, but it may also become more complex, i.e., there is more data that needs to be considered before taking a decision. Moreover, intruder detection should be done in real-time, requiring efficient algorithms. In order to deal with both, the data set growth and the efficiency requirement, the dimension reduction of the detection algorithms input might be very useful. This paper deals exclusively with the problem of the dimension reduction in masquerader detection through command-line behavior.

The approach to dimension reduction proposed in this work is very simple: only a subset of all possible commands is considered when applying a masquerader detection technique. Then, the problem of choosing that subset arises. Related work, summarized in Section 2, suggests several criteria which are analyzed in Section 4. In particular, selection criteria are compared on the basis of their behavior when used with the same masquerader detection algorithm. The detection algorithm is based on a variation of a simple statistic put forward by Schonlau et al. [2] which is studied in Section 3.

Finally, in Section 6 the findings are summarized.

2 Related work

It has been argued by Schonlau et al. [1,2] that unpopular commands are more important than popular ones for the detection of masqueraders. Intuitively, there are some commands which are used by a small group of users and which are rarely used by other users: these commands allow to almost uniquely identify users. This idea suggests the possibility of reducing the dimension of the masquerader detection problem by keeping only a fraction of the most unpopular commands. However, Section 3 shows that the quantitative analysis carried out in [2] does not necessarily lead to the conclusion that keeping the most unpopular commands is the best choice.

Wan et al. [5] use high-frequency commands as signatures of normal user behavior and, by means of a particular decision mechanism, find better results than those reported by Schonlau in [1]. The authors of [5] obtain good results using the 40 highest-frequency commands, but they did not showed how the performance changes as the number of commands is varied. One of the goals of this paper is to explicitly evaluate this relation.

Kim and Cha [7] introduce the concept of "common commands", that is, commands used by more than a given number of users. Moreover, they show that the use of common commands improves the performance of their masquerader detection algorithm. This observation suggests that it may be convenient to retain only the most popular commands. This hypothesis is tested in this paper.

2.1 Schonlau's dataset

In this paper, the same data as Schonlau's et al. [1,2] is used. Command data can be downloaded from Matthias Schonlau's Internet homepage [11]. This dataset consists of commands collected from UNIX `acct` audit data. From all fields of audit data provided by `acct`, only the username and the command were taken. The command history of 70 users was used. Out of these 70 users, 50 were chosen randomly to represent the user base for testing, and the remaining 20 were used as masqueraders. Data was split into 50 files corresponding to 50 different users. Each file contains 15000 commands, one per line. The first 5000 commands of each file are considered genuine. The remaining 10000 commands of each user are divided into 100 blocks of 100 commands each. These blocks are seeded with masquerading users, i.e. with data from another user not among the 50 users. A block is a masquerader with a probability of 1%. If a block is a masquerader, then the following one is a masquerader too with a probability of 80%. As a result, approximately 5% of the test data contain masquerades. Not all users test data have masquerades interspersed.

It should be noted that Schonlau collected truncated commands, i.e., commands with no arguments nor any parameters, as this additional information was not provided by `acct`. It should also be mentioned that due to the way `acct` collects audit data from the system, it is impossible to tell commands typed by human users on the command line prompt from those run automatically from shell scripts. This can lead to some users having a very regular pattern of few commands, due to the repetitive automated execution of shell scripts, such as `.profile` and `make` files.

3 A masquerader detection algorithm

Schonlau et al. [1,2] argued that unpopular commands are more important than popular ones for masquerader detection. Their argument is based on a quantitative analysis of the results obtained by a particular masquerader detection algorithm. This algorithm has two steps:

- Whenever a new session of commands is analyzed, Schonlau et al. evaluate a simple statistic based on that session data and information from previous sessions (i.e., the "normal"-behavior dataset).
- This statistic is compared to a threshold in order to decide whether the commands were typed by an intruder or a legitimate user.

Schonlau and colleagues [2] show that good results are obtained when the particular statistic used takes into account the popularity of the commands, assigning a heavier weight to the unpopular ones. However, they did not compare those results with similar ones obtained using different weighting functions. In this section, such a comparison is made and it is found that the weighting function proposed by Schonlau et al. may not be the optimal and, hence, it may not be completely true that unpopular commands are more relevant to the detection task than popular ones.

3.1 Schonlau's statistic

In order to analyze Schonlau's statistic, some notation is needed:

- N_u (n_u) is the length of user u training (test) data;
- N_{uk} (n_{uk}) is the number of times user u used command k in training (test) data;
- K is the total number of distinct commands;
- U is the total number of users;
- U_k is the number of users that, in the training dataset, use command k.

Schonlau et al. [2] proposed a masquerader detection algorithm that they called *Uniqueness*. This approach is based on the following simple statistic:

$$x_u = \frac{1}{n_u} \sum_{k=1}^{K} F_{uk} \left(1 - \frac{U_k}{U}\right) n_{uk} \tag{1}$$

where F_{uk} is essentially the frequency of use of command k by user u in the "normal"-behavior dataset. More precisely, F_{uk} is defined as

$$F_{uk} = \begin{cases} \frac{v_{uk}}{v_k} & \text{if user } u \text{ training data contains command } k, \\ -1 & \text{otherwise,} \end{cases} \tag{2}$$

where

$$v_{uk} = \frac{N_{uk}}{N_u},\ v_k = \sum_u v_{uk}. \tag{3}$$

Whenever a new command session is available, this statistic is computed and compared to a threshold value: if the statistic is lower than the threshold, then it is decided that the session must correspond to a masquerader.

According to Schonlau et al. [2], U_k denotes command k popularity, so the term $(1 - U_k/U)$ acts as a *uniqueness index*, i.e., the index is 0 if all users have used this command before and is 1 if none of the users has used it before.

3.2 The importance of being unpopular

Note that the uniqueness index works as a weighting function for F_{uk} in Eqn. 1. As such, it has nothing special, except for the intuition offered by Schonlau and colleagues:

> *The uniqueness approach is based on the idea that commands not previously seen in the training data may indicate an attempted masquerade. Moreover, the fewer users that are known to use that command, the more indicative that command is of a masquerade.* (from Schonlau et al. [1])

In order to support this intuitive lemma, the authors of [2] provided two quantitative elements:

1. They computed the values of the statistic using test data from user i and training data for user j, for all pair (i, j) of users. The results were displayed in an $U \times U$ figure in which darker squares corresponded to higher values of the statistics ("scores") and lighter squares to lower scores. Ideally diagonal scores should be perfectly discriminated from non-diagonal ones and the only exceptions should be the masqueraders. The results are reproduced in Fig. 1(a), where it is clear that a good discrimination was achieved.
2. Schonlau et al. [1] computed the False Alarm Rate (FAR) and Missing Alarm Rate (MAR) as a function of the decision threshold. Moreover, they plotted FAR vs. MAR in a graph which is known as a ROC (Receiver Operating Characteristic) curve. The authors of [1] compared the resulting ROC curve to those obtained by other algorithms and they showed that the detection of masqueraders through the proposed statistics behaved reasonably well.

Although Fig. 1(a) shows good results when using the statistic proposed by Schonlau and colleagues, they do not provide enough information to support the claim that unpopular commands are more relevant to masquerader detection than popular ones. In order to effectively measure whether unpopularity is an important feature, the statistic in Eqn. 1 has been modified, replacing the uniqueness index $(1 - U_k/U)$ by a constant, i.e., the same weight is assigned to all commands. Fig. 1(b) and 2 show the results. As can be seen, there are no substantial differences neither between Fig. 1(a) and Fig. 1(b) nor between the corresponding ROC curves shown in Fig. 2. It can be concluded that Fig. 2 shows that there are almost no differences between ROC curves using a constant weight from others applying a higher weight to unpopular commands. Therefore, *no conclusive evidence is found in favor of the use of unpopular commands for masquerader detection.*

3.3 Generalization of the statistic

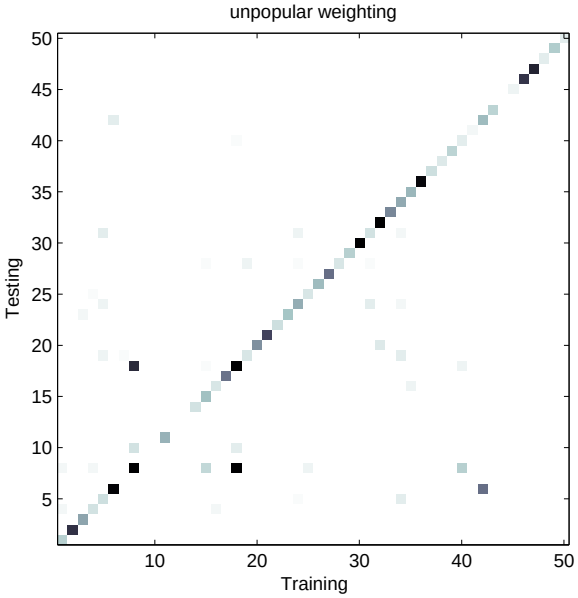
The uniqueness index and the constant weight are only two possible choices for the weighting function of a statistic like in Eqn. 1. In general, it can be said

$$x_u = \frac{1}{n_u} \sum_{k=1}^{K} F_{uk} W_{uk} n_{uk}, \tag{4}$$

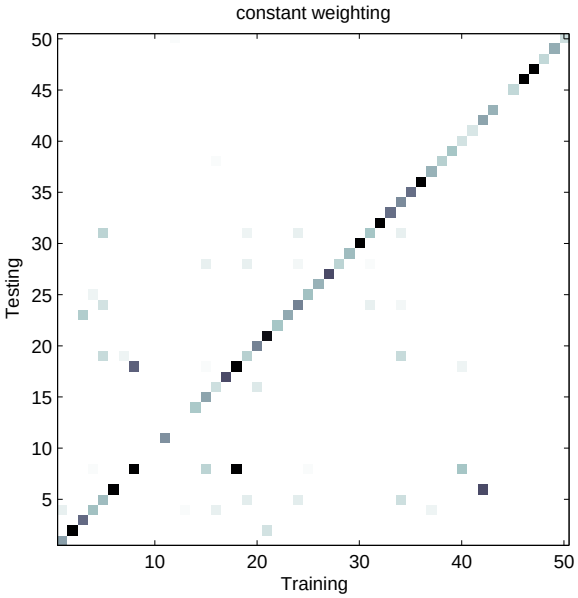
where W_{uk} is a weighting function which depends on the command and the user. Table 1 shows five possible different choices for the weighting function. Among those weighting functions, the command use index and the related inverse command use index require some comment.

The command use index is defined as v_k/V, where $V = max\,(v_k)$. This index is defined using the same criteria and format than for popular commands. The variables v_{uk} and v_k are defined in Eqn. 3 so, the command use index is defined as:

$$\frac{v_k}{V} = \frac{1}{V} \times \left[\frac{1}{N_u} \sum_{u} N_{uk} \right], \tag{5}$$



(a)



(b)

Fig. 1. Visual display of Schonlau et al.'s statistic.

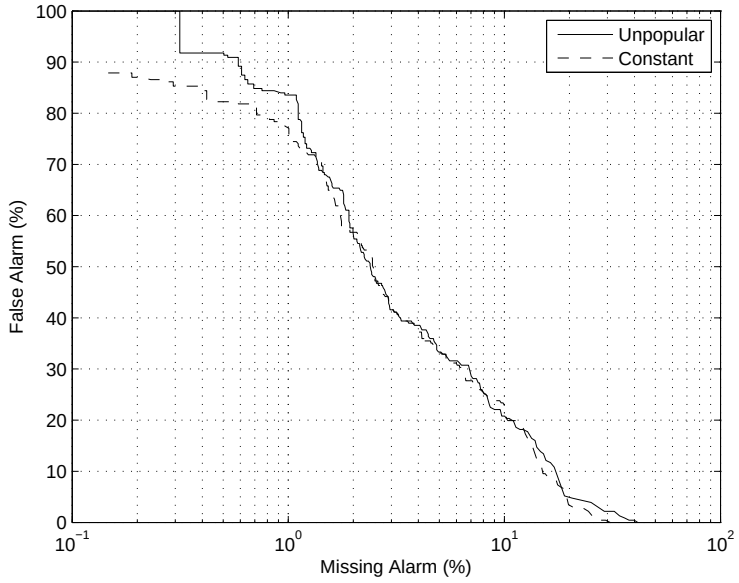


Fig. 2. ROC curves using Schonlau et al.'s statistic

then, it is clear that the command use index is the average frequency of use of command k, where the proportionality constant $\frac{1}{VN_u}$ acts as a normalization factor.

Criterion	Comments	Weighting vector
uniqueness index	$\uparrow$ unpopularity $\Rightarrow \uparrow$ weight	$1 - U_k/U$
inverse uniqueness index	$\uparrow$ unpopularity $\Rightarrow \downarrow$ weight	U_k/U
command use index	$\uparrow$ use frequency $\Rightarrow \uparrow$ weight	$1 - v_k/V$
inverse command use index	$\uparrow$ use frequency $\Rightarrow \downarrow$ weight	v_k/V
constant	same weight for all commands	1

Table 1. Weighing functions

ROC curves were calculated using the statistic in Eqn. 4 and the five different weighting functions in Table 1. Instead of reproducing the curves as it has been done in Fig. 2, the behavior of each ROC curve has been summarized by means of the Area Under the Curve (AUC): the lower the AUC, the better is the discrimination power of the masquerader detection algorithm. The area under ROC curves are shown in Table 2, where, in the third column, a relative order score is shown (1 is the best, 5 is the worse). There, all weighting functions give similar results, except for the command use index. Therefore, the results

in Table 2 support the hypothesis that there is nothing special about using unpopular commands for masquerader detection.

Weight	**AUC**	**score**
uniqueness index	0.059	3
inverse uniqueness index	0.063	4
command use index	0.083	5
inverse command use index	0.056	2
constant	0.053	1

Table 2. Area under ROC curves.

4 Command Extraction

The approach of this paper to the dimension reduction problem is to simply select the proper command subset. Therefore, the difficulty is shifted to find the criterion for such selection. In Section 3.2 it has been shown that choosing only unpopular commands might not be the best choice. In this section other possibilities are explored.

The weighting functions in Table 1 may serve as motivation for several command subset selection criteria. Some of them are:

1. unpopular commands, i.e., those with high uniqueness index;
2. popular commands, with high inverse uniqueness index;
3. frequently used commands, with high command use index;
4. infrequently used commands, with high inverse command use index.

Note that there is no immediate translation of the constant weighting function to a criterion for command selection. Two reasonable alternatives are:

1. to select the subset of commands at random;
2. to select the subset of commands according to its alphanumerical order.

The selection of a subset of commands is more difficult for some of these criteria than others. Fig. 3 shows the (sorted) weights of each command in the training data according to each of the five weighting functions in Table 1. It can be seen that many commands are unpopular and have a similar (high) uniqueness index. Therefore, it is difficult to choose a subset of commands of a given size according to their uniqueness indices: at some point, a rather arbitrary choice between commands of (almost) the same index has to be done. A similar argument holds for infrequently used commands, i.e., with low command use index. On the other hand, clear distinctions can be made among popular (high inverse uniqueness index) and frequently used (high command use index) commands.

In order to compare the different selection criteria, we proceed as follows. For each criterion, the size of the selected subset is varied. For each subset,

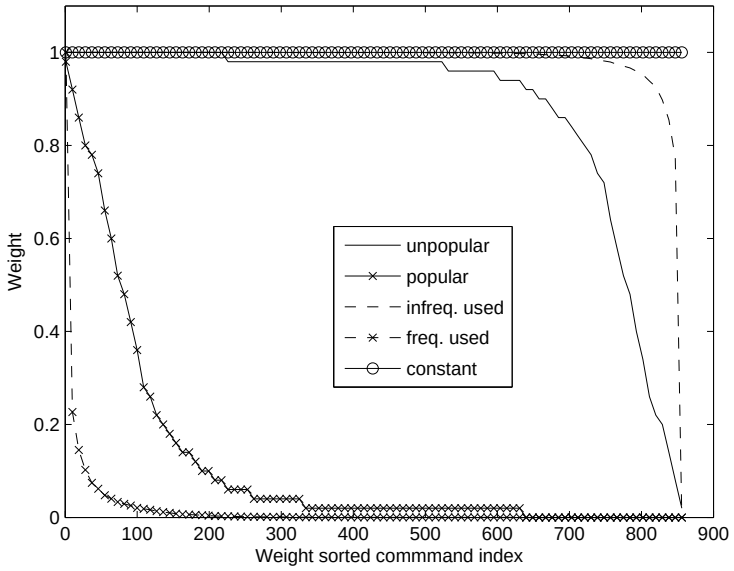


Fig. 3. Weights of training data commands.

the corresponding ROC curve is calculated, using the statistic in Eqn. 4 with *constant weight*[3] and changing the decision threshold. We finally summarize the information of each ROC curve by the area under it. Fig. 4 shows the results (AUC) as a function of the fraction of commands selected for each criterion. Note that all curves agree on two points. On one hand, all curves start at 0.5, i.e., when no commands are taken into account to make a decision, then the decision becomes completely random: either a session corresponds to a legitimate user or a masquerader with equal probability. On the other hand, as the fraction of selected users is increased, each curve tends to the value 0.053 reported in the last row of Table 2. In the particular case of random selection, the curve presented is the average of ten runs.

Two important conclusions can be extracted from the results shown in Fig. 4. First of all, the degradation of the masquerader detection algorithm as the fraction of commands decreases, varies a lot depending on the criterion used. Indeed, *selection of popular and frequently used commands leads to a smooth degradation, while the selection of unpopular and infrequent commands produces a degradation which is worse than that of simple random selection.* Second, *the best performance of the algorithm may not necessarily correspond to accounting*

[3] In Section 3.2 it was shown that the particular weight chosen is not very important. Moreover, there is evidence in favor of a constant weight.

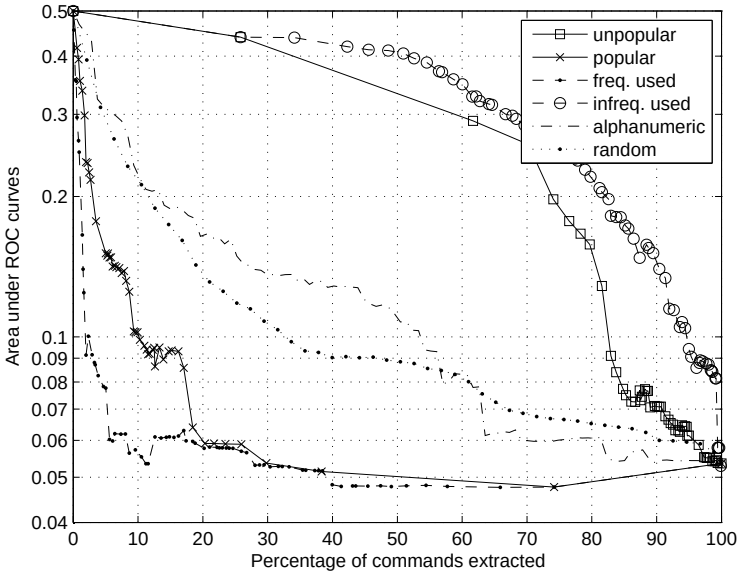


Fig. 4. Area under ROC curves vs. percentage of extracted commands.

for all commands in the training data, as the curves corresponding to the popular and frequently-used command selection criteria show.

5 Validation

Last section showed the results of calculating ROC curves by extracting commands from Schonlau's dataset and detecting masqueraders using Schonlau's statistic. In order to test the validity of the conclusions presented at the end of Section 4, further tests were performed with another publicly available command dataset, Greenberg's Dataset [12]. Moreover, another detection algorithm, based on Naive-Bayes, was used.

5.1 Greenberg's Data

This dataset[12] was collected by Saul Greenberg as part of a project of his PhD thesis in 1988. It contains data from 168 Unix users using *csh* (C shell) as command line shell. All of these users were unpaid volunteers and were either students or employees of the University of Calgary. Greenberg classified users in four groups as follows: Novice Programmers, Experienced Programmers, Computer Scientists and Non-programmers.

Greenberg's data was preprocessed in order to compare results with Schonlau's dataset in exactly the same way as Maxion did in his work [3]. Only truncated command-line data, that is, without command parameters, was preserved. Out of 168 Greenberg's users, 50 were selected as testing subjects having a total number of commands between 2000 and 5000. Out of the remaining 118 users, 25 were selected randomly to serve as a source of masquerader commands. The data of the subject users was truncated to 2000 command lines: the first 1000 commands to be used as training data and the other 1000 commands to be used as testing data. The last 100 commands from each of the 25 masquerader users were concatenated, from which 30 blocks of 10 commands each were selected randomly to be injected into the testing data of every user without replacement, resulting in a testing stream of 130 blocks of 10 command lines for each subject user. Although the masquerader blocks and injection positions were selected randomly, they were held constant across the 50 subject users for consistency purposes.

5.2 Naive-Bayes algorithm

Maxion et al. [3,4] have shown that Naive-Bayes classifiers may be successfully used for detection of masqueraders. The detection algorithm implemented in this work is the no-update Naive-Bayes detector described in [4]. For each user u and for each command k in the training data, the conditional probability of appearance of command k in a block of user u is estimated as

$$P(k|u) = \frac{N_{uk} + \delta}{N_u + K \times \delta},$$

where N_{uk}, N_u and K are defined as in Section 3.1 and δ is a small value which is introduced to avoid zero counts. Given a test sequence of commands, $k_1 k_2 \cdots k_m$, where m is the length of the sequence, the probability that the command k was generated by user u is given by[4]:

$$P(u|k_1 k_2 \cdots k_m) = \frac{P(u) \times \prod_{i=1}^{m} P(k_i|u)}{P(k_1 k_2 \cdots k_m)}.$$

For each user u, a model of "Not User u" ($\overline{u}$) may also be built based on the training data of the remaining users. Indeed, we may estimate the probability

$$P(k|\overline{u}) = \frac{\sum_{i \neq u} N_{ik} + \delta}{\sum_{i \neq u} N_i + K \times \delta}.$$

Given a test sequence of commands, $k_1 k_2 \cdots k_m$, we may compute the probability

$$P(\overline{u}|k_1 k_2 \cdots k_m) = \frac{P(\overline{u}) \times \prod_{i=1}^{m} P(k_i|\overline{u})}{P(k_1 k_2 \cdots k_m)}.$$

[4] Bayes Theorem and conditional independence are used, that is, the hypotheses behind Naive-Bayes classifiers.

Therefore, the relative likelihood

$$L\,(\,u|\,k_1k_2\cdots k_m) = \frac{P\,(\,u|\,k_1k_2\cdots k_m)}{P\,(\,\overline{u}|\,k_1k_2\cdots k_m)}$$

is a measure of how likely it is that the given sequence of commands was generated by user u. Since Maxion et al. [4] assume, although not explicitly, that $P(u) = P\,(\overline{u})$, the relative likelihood simplifies to

$$L\,(\,u|\,k_1k_2\cdots k_m) = \frac{\prod_{i=1}^{m} P\,(\,k_i|\,u)}{\prod_{i=1}^{m} P\,(\,k_i|\,\overline{u})} = \prod_{i=1}^{m} \frac{P\,(\,k_i|\,u)}{P\,(\,k_i|\,\overline{u})}.$$

The detection algorithm decides whether the sequence corresponds to a masquerader by comparing the likelihood ratio to a thresholod: if the likelihood ratio is too small, the algorithm assumes that the sequence was generated by a masquerader. If threshold is varied, then a ROC curve can be obtained.

5.3 Experiments

Following the same criteria than in Sec. 4, experiments were performed processing Schonlau's and Greenberg's datasets.

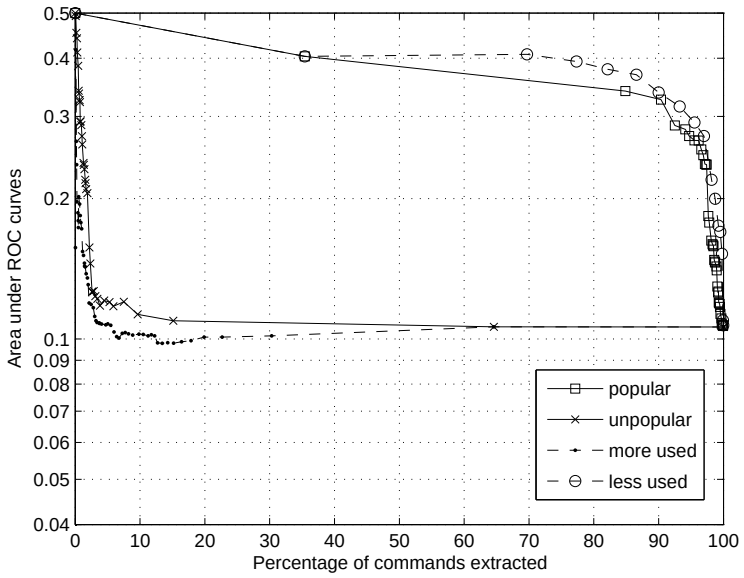
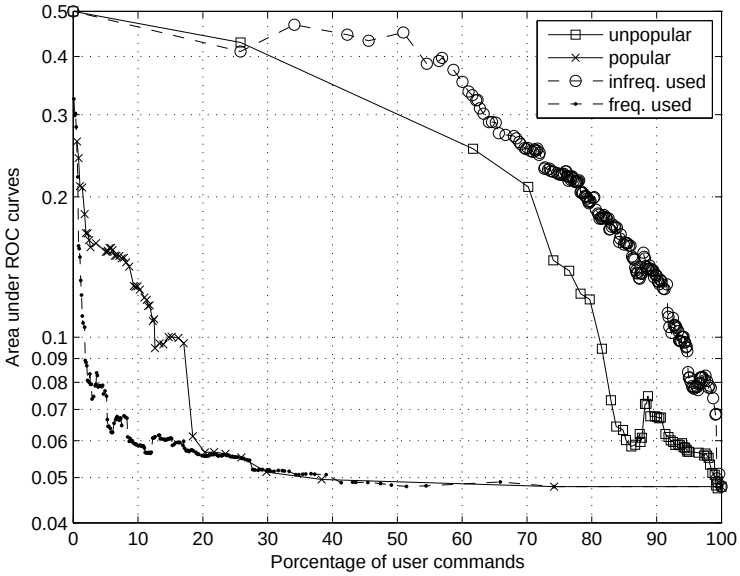
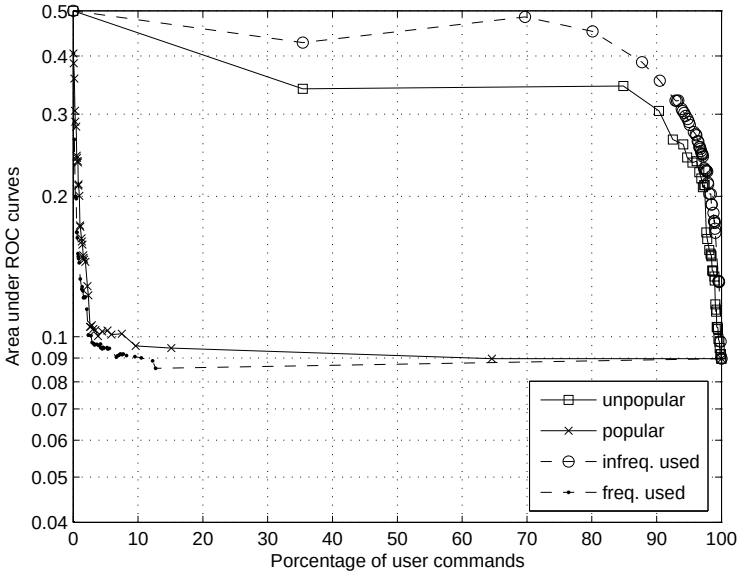


Fig. 5. Area under ROC curves for Greenberg dataset using Schonlau detector



(a) Schonlau dataset



(b) Greenberg dataset

Fig. 6. Area under ROC curves using Naive-Bayes detector

Figures 4, 5, 6(a), and 6(b) show a similar behavior of the Area under ROC curve as function of the fraction of commands extracted for the two datasets and the two detection algorithms. Moreover, it can be verified that the conclusions at the end of Secion 4 are valid for both datasets and both detection algorithms. It should be noticed that random or alphabetical order for comparison are included only in 4.

6 Conclusions and future work

Command space dimension reduction in masquerader detection was tested. Figures 4, 5, 6(a), and 6(b) show that a small number of commands can be used without considerable degradation of the masquerader detection algorithm. According to the experiments, 20% of the most popular commands or 10% of the most frequently used commands are enough to get results comparable to those obtained using all commands. This reduction of the number of commands taken into account, improves masquerader detection by reducing computational complexity.

Posadas et al. [10] studied the case of a masquerader that knows the profile of the legitimate user. Such a knowledge is proved to be very detrimental for most masquerader detection algorithms, with the exception of the algorithm presented by the authors. It must be noted that one of the drawbacks of extracting just the most popular or frequently used commands is precisely that impersonation of a legitimate user is made easier for a masquerader with some knowledge of user's profile: the masquerader needs to know only the user's profile corresponding to a limited number of commands.

In the present work only Schonlau's statistic test and Naive-Bayes are used in order to compute masquerader detection efficiency using different subsets of extracted commands as input. In order to validate the dimension reduction techniques introduced here, also other detection algorithms like Normalized Compression Distance ([8],[9]), Grammar Extraction ([13]), Bayes One-Step Markov and Hybrid Multistep Markov ([1]) should be used. This is a matter of future work.

Dimension reduction analysis has been performed using truncated command line data, but Maxion [3] and Bertacchini and Benitez [9] shown that command line data enriched with command parameters allows a better performance in masquerader detection. Therefore, work needs to be done in order to extend the ideas of this paper to the case of enriched commands.

7 Acknowledgments

This work is part of a more extendend research project which is supported in part by ANPCyT (National Agency of Scientific Promotion and Technology) under the grant PICTO CITEFA 2004 18623. We gratefully acknowledge financial support from ANPCyT. We would also like to thank all members of CITEFA's Si6 Lab for their support specially to Luciano Bello and also to Cecilia Oriolo

for their suggestions and corrections. We would also like to thank the reviewers of this paper for their useful comments.

References

1. Schonlau, M., DuMouchel, W., Ju, W., M. Theus, A., Vardi, Y.: Computer intrusion: Detecting masquerades. Statistical Science **16** (2001) 58–74
2. Schonlau, M., Theus, M.: Detecting masquerades in intrusion detection based on unpopular commands. Inf. Process. Lett. **76**(1-2) (2000) 33–38
3. Maxion, R.: Masquerade detection using enriched command lines. In: International Conference on Dependable Systems and Networks, San Francisco, CA, IEEE (June 2003)
4. Maxion, R.A., Townsend, T.N.: Masquerade detection using truncated command lines. In: International Conference on Dependable Systems and Networks (DSN-02), Washington, IEEE Computer Society Press (June 2002) 219–228
5. Wan, M.D., Wu, H.C., Kuo, Y.W., Marshall, J., Huang, S.H.S.: Detecting masqueraders using high frequency commands as signatures. In: Advanced Information Networking and Applications - Workshops. (March 2008) 596–601
6. Wang, W., Guan, X., Zhang, X.: Profiling program and user behaviors for anomaly intrusion detection based on non-negative matrix factorization. In: 43rd IEEE Conference on Decision and Control, Pardise Island, Bahamas (Dec 2004) 99–104
7. Kim, H.S., Cha, S.D.: Empirical evaluation of svm-based masquerade detection using unix commands. Computers & Security **24**(2) (2005) 160–168
8. Bertacchini, M., Fierens, P.: Preliminary results on masquerader detection using compression based similarity metrics. Electronic Journal of SADIO **7**(1) (February 2007)
9. Bertacchini, M., Benitez, C.: NCD based masquerader detection using enriched command lines. In: Proc. of the IV Congreso Iberoamericano de Seguridad Informatica (CIBSI'07), Mar del Plata, Argentina (November 2007) 329–338
10. Posadas, R., Max-Perera, C., Monroy, R., Nolazco, J.A.: Hybrid method for detecting masqueraders using session folding and hidden markov models. In: MICAI 2006: Advances in Artificial Intelligence, Mexico (November 2006) 622–631
11. Schonlau, M.: Masquerading user data. `http://www.schonlau.net/intrusion.html` (1998)
12. Greenberg, S.: Using unix: Collected traces of 168 users. Technical Report 1988-333-45, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada (December 1988)
13. Latendresse, M.: Masquerade detection via customized grammars. In Julisch, K., Krgel, C., eds.: DIMVA. Volume 3548 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2005) 141–159