Detecting Masquerades in Intrusion Detection Based on Unpopular Commands

Matthias Schonlau [a,1] Martin Theus [b,2]

[a] *National Institute of Statistical Sciences, 19 Alexander Drive, Research Triangle Park, NC 27709-4006, U.S.A.*

[b] *AT&T Labs - Research, Shannon Laboratory, 180 Park Avenue, Florham Park, NJ 07932-0971, U.S.A.*

Abstract

Computer intruders are modern day burglars: some of them steal information, some wreak havoc to the system, some just want to prove they can break in. Computer intrusion detection is concerned with designing alarm systems to prevent break-ins.

This paper presents a method for detecting intruders/users masqueradering as other users. We examine UNIX command streams of users and search for anomalies. We identify anomalies based on unpopular and uniquely used commands.

Key words: intrusion detection, anomaly, UNIX commands, test score, ROC curve

1 Introduction

The research in computer intrusion detection can be broadly divided into two categories: misuse detection and anomaly detection. In misuse detection one attempts to model the behavior of intruders in order to recognize and defend against an attack. There is a myriad of different ways in which software can be "misused". Often software bugs are exploited in ingenious ways.

Anomaly detection does not attempt to defend against specific system vulnerabilities. Rather, anomaly detection aims to spot anything out of the ordinary.

[1] Matthias Schonlau can now be reached at RAND, Statistics Group, 1700 Main Street, Santa Monica, CA 90407, U.S.A.

[2] Martin Theus can now be reached at VIAG Interkom, Georg-Brauchle Ring 23-25, 80992 München, Germany

Usually, a profile is generated for a user or aggregates of users, and a sufficiently large deviation from the profile is taken as potential evidence for the presence of an intruder. Anomaly detection is often considered a second line of defense. It can detect a wider array of intrusions than misuse detection since it is not motivated by specific misuse techniques, and because of its broad scope it is less powerful for any particular misuse technique. Intrusion detection systems often use both lines of defense.

The literature in computer intrusion detection is too vast to survey here. Important early papers were Denning (1986) and Lunt (1988) who did seminal work on the SRI statistical intrusion detection tools IDES, NIDES and, most recently, Emerald. More recent overviews are given, for example, in Denning and Denning (1997) and Amoroso (1999).

In this paper we construct a novel anomaly detection technique targeting masqueraders and investigate some of its properties, specifically the tradeoff between false alarms and missing alarms. As will be shown, our method is based on unpopular and uniquely used commands, and we therefore refer to it as the "uniqueness method".

Masqueraders are people who impersonate other people on the computer. They usually are insiders with malicious intent trying to hide their identity by impersonating other users. They could also be intruders from outside - although in practice most outside intruders immediately try to gain access to the account of the super-user and therefore are a special case. A computer crime and security survey (Computer Security Institute, 1998) ranking computer security problems in terms of their estimated financial damage found that unauthorized access by insiders was most damaging, accounting for about one-third of the total loss.

The uniqueness method can be used as one of the anomaly detection methods within an intrusion detection system. It is not meant to be a stand alone intrusion detection system by itself.

The outline of this article is as follows. To facilitate explanations in later sections we discuss our data in the next section. In Section 3 we motivate the importance of unpopular commands. Section 4 introduces a heuristic test statistic based on this idea. Section 5 describes an experiment to test the statistic and gives results. Section 6 concludes with a discussion.

2 Data

Command level data consisting of user names and commands (without arguments) were generated from output of the UNIX `acct` auditing mechanism. For illustration purposes a few data points are given in Table 1. Our platform is an SGI server running IRIX 6.2. Some commands recorded by the system are implicitly generated, rather than explicitly typed by the user. For example, each execution of the `.profile` file or a `make` file generates commands contained in these files that are also recorded in the data stream.

Command Name	User
`chmod`	matt
`more`	karr
`cat`	vardi
`whoami`	theus
`sendmail`	karr

Table 1
Example of data used

The first 10,000 commands for each of 53 users were recorded. For the analysis, we separated the commands into training data sets of 5000 commands and test data sets of 1000 commands. For some of the analysis reported below, we split each test data set into 10 replications of 100 commands, and into 2 replications of 500 commands.

The use of further information such as time stamps or command arguments is also of interest, but was not considered in this investigation.

3 Motivation

A command has popularity i if exactly i users in the training data use that command. A command with popularity 1 is also called a uniquely used command. In this section we explore unpopular and uniquely used commands.

We first look at the question of what fraction of different commands in a community of U users are of popularity 1, 2, etc. We plot $1-i/U$, $i = 1, \ldots, U$, versus the fraction of commands with popularity greater than i. This plot is shown in Figure 1 for user communities of size $U = 4$, $U = 10$ and $U = 53$. The vertical axis (labeled "Uniqueness Index") thus attaches low values for

popular commands and high values for unpopular commands. Because each of the three curves reaches the highest level on the vertical axis after just over half of the commands, almost half of all commands are uniquely used: they are only used by one out of the 4, 10 or 53 users, respectively.

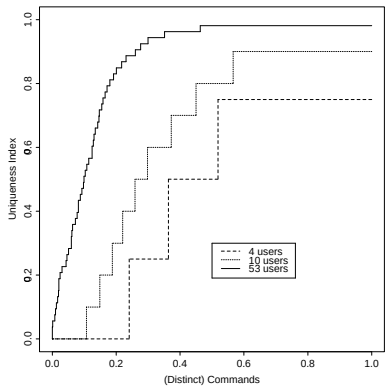

Fig. 1. Stepplot of uniqueness index (1− fraction of number of users) versus the fraction of distinct commands for 4, 10 and 53 users.

While Figure 1 shows that there are many unique and unpopular commands, it is unclear whether these commands account for a large proportion of the total commands. Instead of the fraction of distinct commands, Figure 2 displays the fraction of the total frequencies on the horizontal axis. Figure 2 shows that the percentage of the data consisting of commands used only by a certain percentage of users is approximately the same regardless of whether the user community consists of 4, 10 or 53 users. Also, a substantial fraction of the total command frequencies has a high value on the vertical axis, i.e. is unpopular.

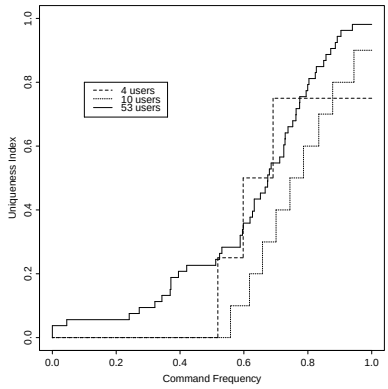

Fig. 2. Stepplot of uniqueness index (1− fraction of number of users) versus the fraction of command frequency for 4, 10, and 53 users.

The commands can be grouped such that each group contains all commands that are used by exactly i users ($1 \leq i \leq U$). We assign an ID to each command such that commands from groups with unpopular commands are assigned

lower ID's than commands from groups with more popular commands. The order within a group is arbitrary. When plotting the command ID versus order in the audit stream the usage pattern of unique/unpopular commands emerges. Such a plot is shown in Figure 3 for each of 53 users. Plots of different users are clearly distinguishable from each other. The first panel corresponds to the super-user, the second is an account that only permits a certain ftp command to be run. The remaining panels correspond to ordinary users.

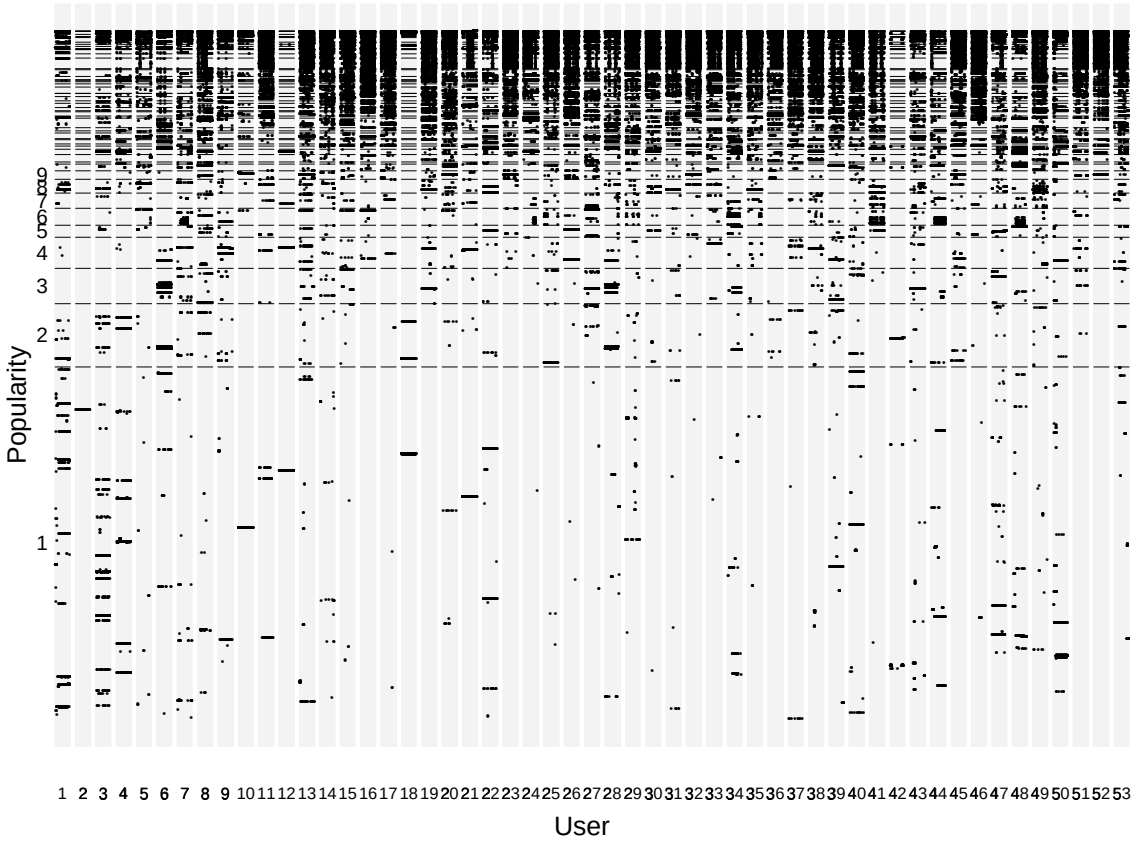

Fig. 3. Visualization of command popularity. Each panel corresponds to one user. Within each user's panel, each command is represented by a single dot. The popularity corresponding to each command is plotted versus "time" (different commands with the same popularity have different ordinates within their popularity group).

4 Method

Before we define a test statistic to detect masqueraders on the basis of unpopular commands we briefly introduce some notation: Let N_u (n_u) denote the length of user u's training (test) data, and N_{uk} (n_{uk}) the number of times user u used command k in the training (test) data. Let K denote the total number of distinct commands and U the total number of users. Further, let U_k denote the popularity of command k.

We define a test statistic that builds on the notion of unpopular and uniquely used commands:

$$x_u = \frac{1}{n_u} \sum_{k=1}^{K} W_{uk} \ (1 - U_k/U) \ n_{uk} \qquad , \tag{1}$$

where the weights W_{uk} are

$$W_{uk} = \begin{cases} v_{uk}/v_k \text{ if user } u\text{'s training data contains command } k \\ -1 \qquad \text{otherwise} \end{cases}$$

where $v_{uk} = N_{uk}/N_u$ and $v_k = \sum_u v_{uk}$.

The fraction $(1 - U_k/U)$ acts as a uniqueness index: it is zero if all users have used this command before, it is 1 if none of the users has used it before. The weights W_{uk} control whether the uniqueness index should be subtracted or added, depending on whether the command was seen before or not. Hence a user who uses commands similar to the ones used in the training data will tend to score high. The order in which the commands appear does not matter. The quantity v_{uk}/v_k represents the command usage relative to other users. It reduces the score contribution of commands that other users often use and this user rarely.

When (1) based on test data exceeds a threshold value a masquerader is suspected and an alarm is generated. The threshold value is preset. The choice of the threshold value is determined by the desired false alarm/ missing alarm rate. The tradeoff between false alarms and missing alarms is discussed in the next section.

5 Results

We evaluate the uniqueness method in two ways: (a) we study to what extent the method can identify the true user among 53 users on fresh data and (b) we study the method's tradeoff between false alarms and missing alarms.

We compute the test data score (1) of user i ($i = 1, \ldots, 53$) based on the training data for user j ($j = 1, \ldots, 53$) for all pairs of users. The resulting 53×53 scores are graphically displayed in Figure 4 (only scores greater than zero are shown). Darker shading corresponds to higher scores, lighter shading to lower scores. For the diagonal entries both the test data and the training data correspond to the same user.

Ideally, we would be able to perfectly discriminate between diagonal entries and off-diagonal entries, that is all diagonal entries should be darker than all off-diagonal elements. As Figure 4 shows, only few off-diagonal elements are darker than some diagonal elements.

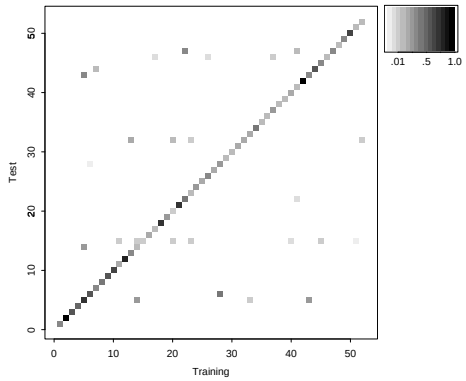

Fig. 4. Visual display of the 53×53 scores from (1) based on 5000 training and 1000 test commands. Only scores greater than zero are shown. The darker the shading, the higher the score.

By varying the threshold at which an alarm is raised one can obtain different rates of false alarms (false positives) and missing alarms (false negatives), obtaining so-called ROC-curves. Figure 5 shows this tradeoff between false alarms and missing alarms for scores based on test data sequences of 100 commands (left), 500 commands (center) and 1000 commands (right).

Each panel in Figure 5 contains five curves to give an idea of the variability of the curves for different data sets; each curve corresponds to 5000 commands training data and 1000 commands test data. The 1000 commands are split up into replications of test data sequences of length 100 (left), 500 (center), and 1000 (right) commands, respectively. The five test data sets correspond to commands 5001 through 10000. For each test data set, the immediately preceding 5000 commands are used as training data.

The lower left corner on this plot represents the ideal situation: no false alarms, and no missing alarms. For the uniqueness method a false alarm rate of 5% corresponds to about 10% missing alarms based on 100 commands, about 2% based on 500 commands, or about 1% based on 1000 commands. As expected, the discrimination among users is easier the longer the test data sequence is.

6 Discussion

Intrusion detection based on command uniqueness is conceptionally very simple. It requires very little storage (W_{ij} for all i, j and U_j for all j) and, since

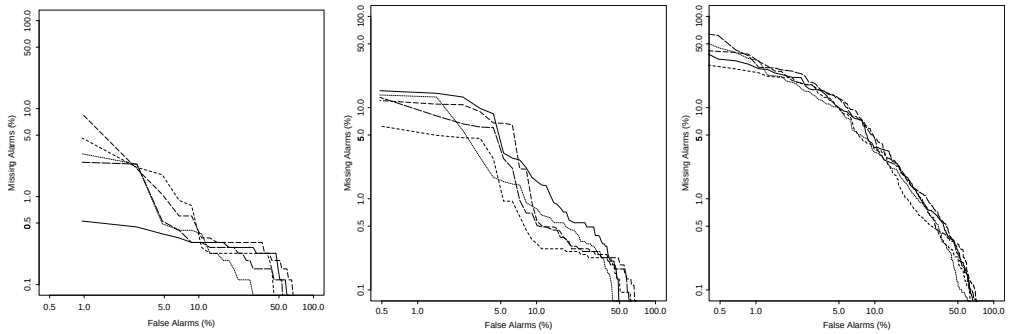

Fig. 5. Tradeoff between false alarms and missing alarms (ROC Curve) based on (1). Each plot displays 5 sets of training and test data. The training and test data sets are of length 5000 and 1000 commands, respectively. The test data are split into sequences of 1000 (left), 500 (center) and 100 (right) commands. All axes are on a logarithmic scale.

(1) is a weighted average, it can be easily updated. Moreover, each update is computationally fast, requiring only a few multiplications.

It is interesting to reflect on how the uniqueness method tries to distinguish between the true and the masqueradering user in the absence of training data from that masquerading user. Commands not used by the community of users are indicative of a masquerader. For all other commands, we borrow strength from data from other users; thus making the implicit assumption that a masquerader is more likely to resemble a combination of the (many) other users than the (one) true user.

For comparison consider the following alternative approach: user u's score equals the percentage of never used commands, i.e. the percentage of commands in the test data that user u did not use in the training data. This alternative approach can be derived as a special case from (1) with $U'_k = 0$ (i.e. the uniqueness index is constant, i.e. all commands are equally popular) and weights

$$W'_{uk} = \begin{cases} 0 \text{ if user } u\text{'s training data contains command } k \\ 1 \text{ otherwise} \end{cases}$$

$W_{uk} = 0$ represents the limiting case where the community of users is infinite such that $v_{uk}/v_u \to 0$ for any one user. This alternate approach turns out to be less powerful than the uniqueness approach. We believe there are two main reasons for this: 1) The Uniqueness method successfully borrows strength from the community of users rather than looking at individual users in isolation. 2) The distinction between various degrees of popularity is preferable to a binary classification "used before"/"never used before".

The uniqueness approach may work especially well in the diversity of a research environment. The UNIX operating system allows and even encourages this diversity. The greater the diversity, the larger training data sets are needed to capture that diversity. For our environment it turned out that in order to capture the most regularly used commands the training data needed to be of size 5000 or larger.

For some applications, e.g., in some military applications, a zero or low missing alarm rate is very important. However, when resources to act on alarms are more limited, a low false alarm rate is desirable. For a sequence of 1000 commands and a missing alarm rate of 5%, the uniqueness method corresponds to about 1% false alarms.

It is difficult to compare various intrusion detection methods due to the lack of a common yardstick (ROC curves like the ones in Figure 5 could provide such a common yardstick for future comparisons). Different methods are based on different data sources and are often too complicated to re-implement for comparison purposes. Based on Figure 5, the uniqueness approach compares favorably with DuMouchel and Schonlau (1999), who report missing alarm rates between 10% and 50% at 5% false alarms for comparable data sources, but also had a considerable time gap between training and testing data. A discussion about the uniqueness method and its relationship to structural zeroes can be found in Theus and Schonlau (1998).

Acknowledgements

M. Schonlau's work is funded in part by NSF grants DMS-9700867 and DMS-9208758. We are grateful for feedback from our network intrusion group with members from AT&T Labs Research, The National Institute of Statistical Sciences and Rutgers University.

References

[1] Amoroso, E. (1999), *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*, Intrusion.Net Books, Sparta, New Jersey.

[2] Computer Security Institute (1998), "CSI/FBI Computer Crime and Security Survey Results Quantify Financial Losses", *Computer Security Alert*, no. 181, April.

[3] Denning, D. (1986), "An Intrusion-Detection Model", *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, pp 222-232.

[4] Denning, D. E. and Denning P.J. (eds.) (1997), *Internet Besieged* , ACM Press, New York.

[5] DuMouchel, W. and Schonlau, M. (1999), A Comparison of Test Statistics for Computer Intrusion Detection Based on Principal Components Regression of Transition Probabilities, In *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*, 30, 404-413.

[6] Lunt, T. (1988), "Automated Audit Trail Analysis and Intrusion Detection", *Proceedings of the 11th National Computer Security Conference".*

[7] Theus, M. and Schonlau, M. (1998), "Intrusion Detection Based on Structural Zeroes", *Statistical Computing & Graphics Newsletter.* Vol. 9, No 1, 12-17.