Masquerade Detection Using Enriched Command Lines

Roy A. Maxion
maxion@cs.cmu.edu

Dependable Systems Laboratory
Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 / USA

Abstract

A masquerade attack, in which one user impersonates another, is among the most serious forms of computer abuse, largely because such attacks are often mounted by insiders, and can be very difficult to detect. Automatic discovery of masqueraders is sometimes undertaken by detecting significant departures from normal user behavior, as represented by user profiles based on users' command histories. A series of experiments performed by Schonlau et al. [12] achieved moderate success in masquerade detection based on a data set comprised of truncated command lines, i.e., single commands, stripped of any accompanying flags, arguments or elements of shell grammar such as pipes or semi-colons. Using the same data, Maxion and Townsend [8] improved on the Schonlau et al. results by 56%, raising the detection rate from 39.4% to 61.5% at false-alarm rates near 1%. The present paper extends this work by testing the hypothesis that a limitation of these approaches is the use of truncated command-line data, as opposed to command lines enriched with flags, shell grammar, arguments and information about aliases. Enriched command lines were found to facilitate correct detection at the 82% level, far exceeding previous results, with a corresponding 30% reduction in the overall cost of errors, and only a small increase in false alarms. Descriptions of pathological cases illustrate strengths and limitations of both the data and the detection algorithm.

1. Introduction

Colloquially, masquerading is the act of substituting oneself for another. To masquerade is to disguise; to assume the appearance of something one is not; to furnish with a false appearance or an assumed identity; or to obscure the existence or true state or character of something. The computer masquerade problem is exemplified in the following scenario. A legitimate user takes a coffee break, leaving his/her terminal open and logged in. During the user's brief absence, an interloper assumes control of the keyboard, and enters commands, taking advantage of the legitimate user's privileges and access to programs and data. The interloper's commands may comprise read or write access to private data, acquisition of system privileges, installation of malicious software, etc. Because the interloper is impersonating a legitimate user (or some other computer identity, such as a program), s/he is commonly known as a masquerader. There are many ways for a masquerader to gain access to legitimate user accounts, e.g., through a purloined password or a hacker's break in. The term may also be extended to encompass abuse of legitimate privileges – the case in which a user "masquerades" as himself; such a person is sometimes termed an "insider," especially when the person is an accepted member of the organization sponsoring the target system.

Masquerading can be a serious threat to the security of computer systems and computational infrastructures. A well-known instance of masquerader activity is the case of Robert P. Hanssen, the FBI mole who allegedly used agency computers to ferret out information later sold to his co-conspirators [4]. Hanssen was a legitimate user, but his behavior was improper. In another example, in November of 2002, thirty thousand credit histories were reported stolen in what a federal attorney called "the biggest case of identity theft in U.S. history [2]." This was an insider undertaking in which a help-desk employee sold misappropriated passwords to someone who used them to masquerade as a legitimate user downloading credit reports. Information in the credit reports was used to withdraw funds from bank accounts, make illegitimate charges to credit cards, etc. Insider masquerading, in various forms, is an enormously costly problem reported by more than 60% of companies surveyed in 1996 [13]. Of even greater concern than economic losses, of course, are attacks on national security.

Detecting masqueraders has long been a challenge, dating as far back as 1988 for practical detection systems [6]. The typical approach is based on the idea that masquerader activity is unusual activity that will manifest as significant excursions from normal user profiles. User profiles are constructed from monitored system-log or accounting-log data. Examples of the kinds of information derived from these (and other) logs are: time of login, physical location of login, duration of user session, cumulative CPU time, particular programs executed, names of files accessed, user commands issued, and so forth [5]. When a deviation from normal behavior is observed, a masquerade (or other misuse) attempt is suspected.

2 Background and related work

There have been several attempts to tackle the problem of detecting masqueraders, one of the earliest of which was the IDES system [6]. About twelve years later, Schonlau and his colleagues [12] presented a nice collection of masquerader research, in which a number of masquerade-detection techniques were applied to the same data set. In terms of detecting masqueraders, the best detection result reported in the Schonlau et al. work was for a Bayes One-Step Markov model, which achieved a hit rate of 69.3% with a corresponding false-alarm rate of 6.7%. In terms of minimizing false alarms (targeted at 1%), their best result was obtained by using a uniqueness metric that achieved a 39.4% hit rate with a corresponding false-alarm rate of 1.4%. Although these results may seem disappointing, they are in fact quite good, considering the extreme difficulty of the problem.

Taking the work of Schonlau et al. as a point of departure, Maxion and Townsend [8] used the Schonlau data to demonstrate a new approach to masquerade detection. Their technique was based on naive Bayes classification, which has enjoyed considerable success in the field of text classification [10]. Compared to the Schonlau et al. results, they achieved a 56% improvement in correct detection (61.5%) at a false-alarm rate (1.3%) that is the lowest reported in the literature so far. They also amplified their results with an analysis of the errors made by the detector as applied to an alternative configuration of the Schonlau et al. data set. The error analysis exposed certain limitations of the Schonlau et al. framework, and suggested various things that might improve future results, including the use of better command-line data.

Both Schonlau et al. [12] and Maxion and Townsend [8] studied masquerade detection using truncated command lines, with each command line comprising just a single command, stripped of any accompanying flags, arguments or elements of shell grammar (such as pipes or semicolons). In all of these experiments, 5000 lines of training data were available for each user, plus 10,000 lines of testing data. The unit of classification was a block of 100 contiguous command lines, which means that in a practical setting the detector would need to wait for 100 user commands before deciding whether or not those commands were typed by the legitimate user or by a masquerader. While this was only an experimental scenario, the delay involved in waiting one hundred command lines before classification is clearly undesirable. However, Maxion and Townsend [9] report in further work that reduction of the unit of classification to a block of only 10 commands leads to a 20-point reduction in the hit rate (to just 47.1%) and a corresponding false alarm rate similar to that obtained with block size 100 (1.6%). This result is based on using a factor of 10 fewer commands, and yet it retains nearly the same false-alarm rate as earlier work using blocks of 100 commands.

Correct detection near 99% with a corresponding false-alarm rate under 1% would be considered worthy outcomes. Given the comparatively poor results obtained with the various algorithms tried by Schonlau et al., even using the larger block size, as well as the substantially improved, yet not spectacular results achieved by Maxion and Townsend, it seems reasonable to conclude that acceptable masquerade-detection capability is unlikely to be achieved with the information-impoverished data used, i.e., command lines stripped of any information beyond the basic command. The obvious questions, then, are these: Would data containing more information about a user's command-line behavior improve profiling and thus detection? Compared to the "truncated" data used so far, would data "enriched" with command-line flags and arguments facilitate better masquerade detection? It is from these questions that the current work arises.

3 Objective and approach

The objective of the present work is to determine the extent to which information-enriched command-line data can improve masquerade detection over truncated command-line data. Examples of truncated and enriched versions of the same data are shown in Table 1.

Truncated	Enriched
cd	cd cpsc504
more	more susan.lst
diff	diff susan.lst julie.lst
lpr	lpr -Pjp susan.lst
setenv	setenv TERM amb amb
rwho	rwho -a
set	set prompt set prompt = "VAXC{!} [$cwd:t] $-->$"
nroff	nroff -me proposal | more
ls	ls -F -l ∼candym/.em* l -l ∼candym/.em*
enscript	enscript -2Gr -L66 -Palw -h *.c print66*.c

Table 1: Examples of truncated (left) and enriched (right) command-line data.

Reaching this determination in a way that effects fair comparisons with previous work requires acquisition of new data, because the data used in earlier studies were already truncated, so they cannot be fairly compared with enriched data sets from alternative sources. Once the new, enriched data (replete with full command-line information) are obtained, a truncated data set will be constructed from the enriched data. Then both data sets will be divided into training and testing data, masquerade "intrusions" will be injected into the testing data, and a naive Bayes classifier will be used to detect the intrusions. Results are expected to show that enriched data, containing more information, better support masquerade detection than truncated data do.

The following sections describe the data, the experimental methodology and the results.

4 Data

The data used by Schonlau et al. did not allow for any enrichment - no information about flags, aliases, arguments or shell grammar was provided. This necessitated procuring data from elsewhere. The data were obtained from an earlier study whose original purposes included describing how people use commands in Unix, and reporting on the statistics of the complete command as entered by the user, as opposed to just the command itself. These data, collected by Saul Greenberg, comprised full command-line entries from 168 unpaid, volunteer users of the Unix *csh* system, and are documented in [3]. All data sets were rendered anonymous by replacing user-confidential information (e.g., user names) with dummy information that retained the semantics of the original data. The original data are split into four groups comprising 55 novice users, 36 experienced users, 52 computer-scientist users, and 25 non-programmer users, all of whom were affiliated with the University of Calgary (Canada) as students, faculty, researchers or staff. Greenberg defined these categories carefully, but they will not be repeated here due to space limitations and the lack of a compelling need to do so.

The Greenberg data provide much more complete information about the user's command line behavior than do the Schonlau et al. data. Greenberg supplies the command line as typed, including flags, grammar and arguments, along with a note of whether the typed command was an alias for anything, and if so, what. In addition, a record of history use and errors is kept, and each set of commands executed in the same terminal window is stamped with the time at which that window was opened and closed.

An example of an original Greenberg data entry appears in Table 2. S-lines give the day, date and time of opening of the Xterm in which the command line was issued. E-lines give the day, date and time of closing (or ending) of the Xterm in which the command line was issued, or NIL if not available. C-lines give the command line as typed by the user. D-lines give the path of the working directory. A-lines give the command executed by the shell if different from the one that the user typed, i.e., the aliased command line. H-lines indicate whether history was used. X-lines indicate whether an error was made, and if so, of what sort.

```
S Fri Feb 20 23:39:46 1987
E NIL
C purge
D /user/cpsc500/l01b91/xxxxxx/c500
A rm -i -i .ed_[0-9]* .emacs_[0-9]* .*.CKP
  .*.BAK *.CKP *.BAK ; clear ; /bin/ls -al | more
H NIL
X NIL
```

Table 2: Example of Greenberg raw data.

Because the current experiments focus on profiling user behavior through command-line activity, the Xterm time, history and error categories are ignored for the purpose of constructing the enriched command-line data employed in this work. After removal of directory and filename-type arguments, and ignoring the current directory, history, and error categories, the Greenberg entry would be transformed as follows:

```
C purge
A rm -i -i  ; clear ; /bin/ls -al | more
```

Truncated command lines consist of only the first item typed, with no information about aliases, flags or arguments; hence the truncated command line version of the data for this entry would become:

```
purge
```

An enriched command line is a concatenation of the whole command line typed, including flags, arguments and items of shell grammar (such as & or $>>$) together with the expansion of any alias employed; hence the enriched command-line version of this entry would be:

```
purge  rm -i -i  ; clear ; /bin/ls -al | more
```

Side-by-side examples of the two types of data are shown in Table 1 of Section 3. Note that the enriched data must be stripped of directory and filename arguments, because normal user data is used as a proxy for masquerader data, and it is clearly unreasonable to inject John's data with data from Jack, whilst leaving in file and directory names specific to Jack. Thus, the final version of the two types of data would be as shown in Table 3.

Truncated	Enriched
cd	cd
more	more
diff	diff
lpr	lpr -Pjp
setenv	setenv
rwho	rwho -a
set	set
nroff	nroff -me | more
ls	ls -F -l -l
enscript	enscript -2Gr -L66 -Palw -h

Table 3: Examples of truncated (left) and enriched (right) command-line data, after removal of file/directory names.

5 Experimental method

Two experiments were conducted – one with truncated command lines and one with enriched command lines. Readers familiar with previous work will note differences in experimental methodologies between earlier studies and this one. These differences are due primarily to constraints imposed by the data available. For example, the amount of data used in training and testing sets differed here, compared to those used in earlier work, primarily due to the contents of the original data sets.

5.1 Selection of subjects

The Greenberg data contains 168 users, not all of which were well suited for the present masquerade study. A subset of users – fifty victims and 25 masqueraders – was selected in consideration of various issues, detailed below.

In earlier studies using the Schonlau et al. data, each of the 50 users had 15,000 commands; these were separated into a set of training data (the first 5000 of the 15,000 commands) and testing data (the remaining 10,000 commands). It would be ideal to match the numbers of commands in these earlier data sets, but unfortunately this could not be done, because most of the Greenberg users did not have that many commands.

Figure 1 shows the decision diagram for selecting experimental users from the pool of 168. Of the 168 users in the Greenberg data, 112 produced fewer than 2000 commands; the remaining 56 users produced between 2024 and 12,056 commands, depending on the user. Of these 56 users, 6 had generated markedly more command lines than the others, having in excess of 5000 command lines each. These 6, due to their being in essence, outliers, were removed from consideration, leaving 50 subjects, a group coincidentally the same size as was used in previous work. These 50 users were distributed across the user categories as follows: 13 novices, 15 experienced, 21 scientists and 1 non-programmer. A further 25 users were selected at random from the remaining pool of 118 users to serve as a source of masquerader commands.

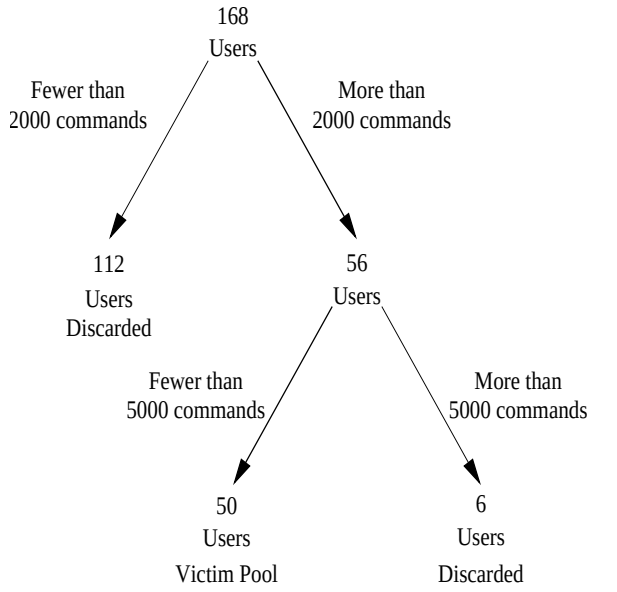

Figure 1: Decision diagram for selecting the 50 subjects comprising the victim pool.

5.2 Training and testing data

The data for the subject users was truncated to 2000 command lines. The first 1000 lines for each user were kept aside as training data upon which to base a profile of self. The second 1000 command lines were kept as unlabelled self data with which to test self-recognition capacity.

The last 100 command lines from each of the 25 masquerader users were concatenated to give a stream of 2500 command lines which constituted a pool from which to draw "masquerade" data. These 2500 command lines were processed into 250 nonoverlapping testing units of 10 command lines each. Thirty of these units were then selected at random and injected at randomly selected positions, without replacement, into the stream of 1000 self command lines, resulting in 130 blocks of 10 command lines for testing self and non-self recognition capacity. Note that although the masquerade blocks and injection positions were randomly selected, they were held constant over the 50 users, i.e., the test data for each user contains the same masqueraders in the same positions. For example, each victim might have been injected at position 2 with the same masquerader block, thus ensuring consistency throughout the injected data.

5.3 Detection/classification algorithm

The naive Bayes algorithm with simple updating (passing each approved block of self back to the self model for X) was employed. Naive Bayes classifiers are simple probabilistic classifiers known for inherent robustness to noise and fast learning time (learning time is linear in the number of training examples). These classifiers have a history of successful use in text classification, where the task is to assign a document to a particular class, typically using the so-called "bag of words" approach, which profiles document classes based simply on word frequencies [10]. Deciding whether a newspaper article is about sports, health or politics, based on the counts of words in the article, is similar to the task of deciding whether or not a stream of command lines issued at a computer terminal belongs to a particular authorized user or not.

In the present context, the classifier works as follows. The model assumes that the user generates a sequence of commands lines, one line at a time, each with a fixed probability that is independent of the command lines preceding it (this independence assumption is the "naive" part of naive Bayes). The probability for each command line c for a given user u is based on the frequency with which that command line was seen in the training data, and is given by:

$$P_{c,u} = \frac{\text{Training Count}_{c,u} + \alpha}{\text{Training Data Length} + (\alpha \times A)}$$

where α is a pseudocount and A is the number of distinct command lines (i.e., the alphabet) in the data. The pseudocount can be any real number larger than zero (0.01 in this study), and is added to ensure that there are no zero counts; the lower the pseudocount, the more sensitive the detector is to previously unseen commands. The pseudocount term in the denominator compensates for the addition of a pseudocount in the numerator. The probability that a test sequence of the five command lines "a a b b b" was generated by a particular user, say User 1, denoted as $u1$, is:

$$P_{u1,a} * P_{u1,a} * P_{u1,b} * P_{u1,b} * P_{u1,b}$$

or $(P_{u1,a})^2 * (P_{u1,b})^3$ where $P_{u1,a}$ is the probability that User1 typed the command line a. For each User X, a model of Not X can also be built using training data from all other victims. The probability of the test sequence having been generated by Not X can then be assessed in the same way as the probability of its having been generated by User X. The larger the ratio of the probability of originating with X to the probability of originating with Not X, the greater the evidence in favor of assigning the test sequence to X. The exact cut-off for classification as X, that is the ratio of probabilities below which the likelihood that the sequence was generated by X is deemed too low, can be determined by a cross-validation experiment during which probability ratios for sequences which are known to have been generated by self are calculated, and the range of values these legitimate sequences cover is examined.

The success of naive Bayes has often struck researchers as surprising, given the unrealistic assumption of attribute independence which underlies the naive Bayes approach. However, [1] demonstrates that naive Bayes can be optimal even when this assumption is violated. Further general details regarding naive Bayes can be found in [7] and [11]. For a more detailed description of the algorithm in the context of masquerade detection, the reader is referred to [8].

5.3.1 Training procedure

The goal of the training procedure is to establish profiles of self and nonself, and to determine a decision threshold for discriminating between examples of self and nonself.

For each user, the detector builds a profile of self based on the command line frequencies from the 1000 command lines for that user; it builds a separate profile of nonself based on the training data for the other 49 users, i.e., 49 x 1000 command lines.

Five-fold cross-validation, with sets of 200 and 800 command lines, was performed in order to determine the appropriate threshold for the detector. In each round, the larger set was used to build a profile of self and the smaller set was used to determine the range of values the detector would assign to material generated by the authorized user. The test data for each round consisted of 20 blocks of ten command lines; thus a total of 100 scores (five rounds of validation times 20 blocks per round) for self were generated during cross-validation. For each user the individual threshold was taken to be the value associated with the block of data deemed least likely to have come from that user. Prior experience indicates that better performance with respect to the false alarm problem is achieved with a single generic threshold for all users, as opposed to an individual threshold customized to each user. This generic threshold was taken to be the mean of the 50 individual thresholds.

5.3.2 Testing procedure

The detector was presented with 130 blocks of mixed, unlabelled self and non-self data; this is approximately 3/4 self (100 blocks) to 1/4 nonself (30 blocks), as described earlier. For purposes of updating, any block deemed to be consistent with the profile of self was passed back to the detector to update the model of self for the user in question. Thus, updating of the profile of self (but not of the threshold, which was not updated at all) occurred after each block accepted as self.

6 Results and analysis

The experimental results show clearly that including information about aliases, flags and items of shell grammar brings about a dramatic improvement in masquerade detection. In the present study there was an improvement from 70.9% hits to 82.1% hits using truncated and enriched data, respectively; this is an increase of 15.8%. The false alarm rates of the two data types are comparable, at 4.7 and 5.7% respectively, an increase of 21.3%. These figures can also be interpreted in terms of the cost of errors, namely misses and false alarms. If the costs of these errors are equal, then the cost function (weighted combination of misses and false alarms) goes from 33.8 to 23.6, an improvement of 30.2%. In previous work, the costs of errors were not equal; misses cost six times as much as false alarms [8]. Using this cost measure, the improvement was 9.1%, from 57.3 to 52.1. Table 4 summarizes the results.

Data	Greenberg	Greenberg
Type of data	Truncated	Enriched
Amount of training data	1000	1000
Block size	10	10
Hits %	70.9	82.1
Misses %	29.1	17.9
False Alarms %	4.7	5.7
Cost (equal weights)	33.8	23.6
Cost (FA=6*Miss)	57.3	52.1

Table 4: Results for naive Bayes on Greenberg data, averaged across all users.

The average rates portrayed by the table do not tell the whole story. It is important to note that not only is the average hit rate higher for enriched command lines, but that this increase in the average reflects an improvement across the board, rather than just for one or two outliers. With enriched command lines, there is only one user (User 36) for whom the hit rate is less than 66%, compared with 20 such users when truncated command lines are employed.

6.1 ROC curve

In assessing the results of a masquerade detector, one is concerned with the trade-off between correct detections (hits, or true positives) and false detections (false alarms, or false positives). These are often depicted on a receiver operating characteristic curve (called an ROC curve) where the percentages of hits and false alarms are shown on the y-axis and the x-axis, respectively.[1] ROC curves for the naive Bayes classifier (with updating) on the truncated and the enriched versions of the Greenberg data (segmented into sequences of length 10) are presented in Figure 2. The ROC curves were obtained by stepping the value of the threshold through a range bounded at one end by the threshold for which 100% hits were obtained, and on the other end by the threshold for which no false alarms were observed, in increments of 0.05. The dotted curve applies to the truncated data; the bold curve applies to the enriched data. Lenient decision criteria allow a higher hit rate, but also a higher false-alarm rate; more stringent criteria tend to reduce both rates. Each point on the curves indicates a particular trade-off between hits and false alarms. Points nearer to the upper left corner of the graph are the most desirable, as they indicate higher hit rates and lower false-alarm rates.

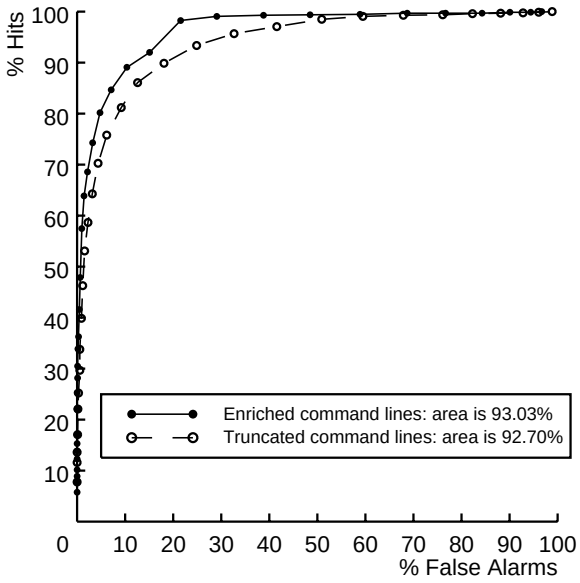

Figure 2: Receiver operating characteristic (ROC) curve for the naive Bayes classifier (with updating) as applied to truncated and enriched Greenberg data.

Although the difference in area under the two curves is very small (93.03% for enriched data, and 92.70% for truncated data), it is still highly significant for performance. For example, it is possible to obtain almost 60% hits at a false alarm rate of around 1% with the enriched data, whilst less than 50% hits can be obtained at that level of false alarms with the truncated data. At the high hit-rate end of the continuum, enriched data allows 85% hits at 7% false alarms, whereas a similar hit rate on truncated data results in nearly doubling the false-alarm rate to almost 13% false alarms. The curve for the enriched data hits 100% detection at a false alarm rate of 22%, but the curve for the truncated data hits 100% only at a false alarm rate of 58%, more than dou-

[1] For a thorough exposition of ROC curves, see [14].

ble the false alarms. If the costs of misses and false alarms are the same, then using the truncated data comes at an enormous expense. The enriched data enable a higher hit rate at a much lower false alarm rate.

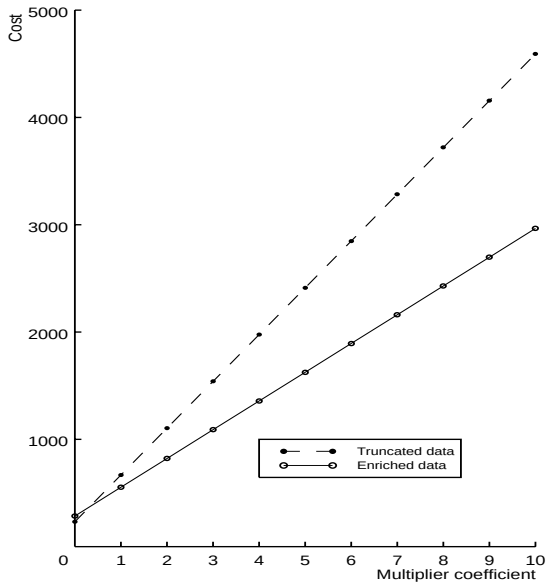

Figure 3: The cost of error in which a miss is N times the cost of a false alarm. The point of equal cost of using either truncated or enriched data is .3155, where the two lines cross.

Figure 3 shows the cost of error for cases in which a miss is N times the cost of a false alarm, with N shown along the x-axis. When N is 1, the cost of a miss is the same as the cost of a false alarm. When N is .3155, the cost of using either truncated or enriched data is the same. As the relative cost of a miss to a false alarm goes up, the cost of using truncated data becomes much higher than the cost of using enriched data, as shown by the diverging lines in the graph.

For the masquerade detection problem, it seems reasonable to estimate costs in this way, because a missed masquerader may cause damage worth far more than the mere cost of investigating a false alarm. Of course this depends on how many false alarms occur for every missed masquerader, but given equal base rates we can see that once the cost of a miss rises to being more than about one third (.3155) the cost of a false alarm, then using enriched command-line data entails an enormous advantage which escalates as the cost differential grows.

6.2 Transition table

While ROC curves and cost functions can portray the gross differences between using truncated and enriched data, examining the transitions of events from one kind of data to another can also be informative. The objective of using enriched data was to facilitate as many transitions as possible, from misses in the truncated data to hits in the enriched data. How many transitions of various types were there, and what were the interesting cases? Table 5 summarizes the total transitions from truncated to enriched data. For example, of all the masquerade injections that were missed in the truncated data, 67.43% of them transitioned to hits in the enriched data. Of all the injections that were correctly detected in the truncated data, only 11.84% of them transitioned to misses in the enriched data, for a net gain of 55.59%. Of the hits in the truncated data, 88.16% of them remained hits in the enriched data.

Transition type	Percent change
Miss to hit	67.43
Hit to miss	11.84
Hit to hit	88.16
Miss to miss	32.57

Table 5: Overall transitions from truncated to enriched data.

A complete presentation of the transitions between the truncated and enriched conditions, resulting from all 30 injections into each of 50 victims, is shown in Table 6. These were run in two conditions, truncated and enriched, with the anticipation that missed detections in the truncated condition would transition to correct detections in the enriched condition. For example, column 1 and row 1 of the table shows the effect of masquerader 1 on victim (user) 1. The symbol * indicates that this injection was correctly detected in both the truncated and enriched conditions. Column 1 row 7 (+) shows a case in which a miss in the truncated condition transitioned to a hit in the enriched condition, which is exactly what was hoped for. Of course there were also a few cases, such as for masquerader 7, victim 1, in which the enriched data effected a reduction in correct detections, but overall these were in the minority.

6.3 Pathologies showing effects of enriched data

Close examination of Table 6 shows the details of classifications that improved from a miss to a hit or deteriorated from a hit to a miss upon moving from truncated data to enriched data. As seen in the table, the changes indicated in the columns corresponding to masqueraders 2, 16 and 26 stand out, because they show the largest numbers of miss-to-hit transitions. Detection of these injections improved more than two-fold with the introduction of enriched data;

Victim	Injection/Masquerader Number																														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	*	+	*	*	+	*	-	*	*	*	*	*	*	*	*	+	*	*	*	*	*	!	*	-	+	+	+	*	*	*
2	*	+	*	*	*	*	!	+	*	*	*	+	*	*	*	*	*	*	*	*	+	+	*	!	*	+	*	+	*	*
3	*	*	*	*	!	*	-	*	*	*	*	*	*	*	*	+	*	*	+	*	*	!	*	-	+	!	+	*	!	*
4	*	!	*	*	*	*	!	*	*	*	*	*	*	*	*	*	*	*	*	*	+	+	*	!	*	+	*	*	-	*
5	*	*	!	*	*	-	-	*	*	*	*	*	-	*	+	!	-	*	+	-	*	*	!	!	+	+	*	*	*	*
6	*	!	*	*	*	-	!	+	*	*	*	*	*	*	*	+	-	*	+	*	!	*	*	!	+	+	*	+	-	-
7	+	!	*	+	!	*	!	+	*	*	*	*	*	*	*	+	*	*	!	*	!	!	*	!	*	+	+	+	-	*
8	+	+	*	+	-	+	!	*	*	*	*	*	+	*	*	+	+	*	!	+	+	+	*	!	+	+	+	+	!	*
9	*	+	*	*	*	-	-	*	*	*	*	*	-	*	*	+	-	*	+	-	+	+	*	!	*	*	*	+	*	*
10	*	+	*	+	*	!	-	*	*	*	*	*	!	*	*	+	!	*	+	!	*	+	*	-	*	+	*	*	*	*
11	*	+	*	*	-	-	!	+	*	*	+	*	-	*	*	+	-	*	*	-	!	+	*	!	+	+	*	+	-	+
12	*	*	*	*	+	*	-	*	*	*	*	*	*	*	*	+	*	*	+	*	*	+	*	-	+	!	+	*	*	*
13	*	+	*	*	*	-	-	*	*	*	!	*	-	*	*	+	*	*	+	-	!	*	*	-	*	+	*	+	*	*
14	+	+	*	*	+	*	-	+	*	*	*	*	*	*	*	+	-	+	+	*	*	+	*	!	+	+	+	+	*	*
15	*	+	*	+	!	*	-	*	*	*	*	*	*	*	*	+	*	*	!	*	*	!	*	-	+	+	+	*	!	*
16	+	!	+	+	*	!	-	*	*	*	*	*	+	-	*	+	!	+	+	+	+	!	*	-	+	+	+	*	*	*
17	+	!	*	*	+	*	!	+	*	*	*	*	*	-	*	!	*	*	+	*	+	!	*	!	+	+	+	+	*	*
18	*	+	+	+	!	*	!	*	*	+	*	*	*	*	+	+	*	*	!	*	*	!	*	!	+	+	!	+	!	+
19	+	!	*	*	+	*	-	*	*	*	*	*	*	*	*	+	*	*	*	*	!	!	*	-	*	!	+	+	*	*
20	+	!	*	+	+	*	-	*	*	*	*	*	*	-	*	!	*	*	*	*	+	!	*	-	+	+	+	*	+	+
21	*	!	*	*	*	*	!	+	*	*	*	*	*	*	*	+	-	+	+	*	!	-	*	!	*	*	+	*	*	*
22	*	+	*	*	*	*	!	*	*	*	*	*	*	*	*	+	*	*	*	*	*	+	*	!	*	+	+	+	*	*
23	*	!	!	*	*	-	!	*	*	*	*	*	-	*	*	+	-	*	*	-	+	*	*	!	*	*	*	*	*	*
24	*	+	*	*	*	*	!	*	*	*	*	*	*	*	*	+	*	*	*	*	+	*	*	!	*	*	*	*	*	*
25	*	+	*	*	-	-	!	+	*	*	*	*	-	*	*	+	-	*	!	-	+	+	*	!	+	+	*	*	-	*
26	+	+	*	!	+	*	!	+	*	*	*	*	*	*	*	+	*	*	+	*	+	!	*	!	*	+	*	+	*	*
27	+	!	*	+	+	+	!	+	*	*	*	*	+	-	*	!	!	*	+	+	+	!	*	!	+	+	+	*	+	*
28	*	+	*	*	*	*	-	*	*	*	+	*	*	*	*	+	*	*	*	*	+	!	*	-	*	+	+	+	*	+
29	*	+	+	*	*	+	-	*	*	*	*	*	*	*	*	*	*	*	*	*	*	+	+	-	*	+	+	*	*	*
30	*	!	+	*	-	*	-	*	*	*	*	*	*	*	*	+	-	*	!	*	*	*	*	-	+	+	*	*	-	*
31	*	+	*	*	*	*	-	*	*	*	*	*	*	*	*	+	-	*	+	*	+	!	*	-	*	+	*	+	*	*
32	+	!	+	*	*	*	-	*	*	*	*	*	*	*	*	+	*	*	*	*	*	!	*	-	+	+	+	*	*	*
33	+	+	+	*	+	*	-	*	*	*	*	*	*	-	*	!	*	*	+	*	*	!	*	-	+	+	+	*	!	*
34	+	+	+	*	+	-	-	*	*	*	*	*	-	*	*	!	-	*	+	-	+	!	*	-	+	+	+	*	*	*
35	*	+	+	*	-	+	-	*	*	*	*	*	+	*	*	+	+	*	!	+	*	*	*	-	+	*	*	*	-	*
36	*	!	*	*	*	-	-	*	*	*	!	*	-	!	*	!	-	*	+	-	!	+	*	-	*	*	*	*	!	*
37	*	*	+	+	*	+	-	*	*	*	*	*	+	*	*	+	+	*	*	+	*	+	*	-	*	+	*	+	*	*
38	*	+	*	*	*	*	-	*	*	*	*	*	*	*	*	+	*	*	*	*	*	+	*	-	*	+	+	+	*	*
39	*	*	+	*	*	*	-	*	!	-	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
40	*	!	*	*	*	*	-	*	+	!	+	!	*	*	+	!	*	*	*	*	+	*	*	-	+	*	*	*	-	+
41	*	+	*	*	*	*	-	*	!	!	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
42	*	+	*	*	*	*	-	*	+	+	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
43	*	+	*	*	*	*	-	*	+	+	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
44	*	+	!	*	*	*	-	*	+	!	*	!	*	*	*	*	*	*	*	*	+	*	*	-	*	*	*	*	*	*
45	*	*	!	*	*	*	-	*	!	!	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
46	*	+	*	*	*	*	-	*	*	*	*	!	*	*	*	*	*	*	*	*	+	*	*	-	*	*	*	*	*	*
47	*	*	*	*	*	*	-	*	+	!	*	!	*	*	+	+	*	*	*	*	*	*	*	-	*	+	*	*	*	*
48	*	!	*	*	*	!	-	*	*	*	*	*	!	*	*	+	!	!	+	!	*	*	*	-	+	*	*	*	*	*
49	*	*	+	*	*	*	-	*	!	!	*	!	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
50	*	*	!	*	*	*	-	*	*	-	*	*	*	*	*	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*

Table 6: Transition in classifications of nonself blocks between truncated and enriched command line data. For example, the + in column 2 of row 1 indicates that masquerader 2 was missed in victim 1's truncated data, but correctly detected in the enriched data. Key: (+) miss to hit; (-) hit to miss; (*) hit to hit; (!) miss to miss.

masquerader 2 was detected in 26 more instances, and masqueraders 16 and 26 were detected in 30 more instances. All of the misses for these three masqueraders were missed in both the truncated and enriched conditions. All of the transitions were from miss to hit, and none of the transitions were from hit to miss. Masqueraders 7 and 24 experienced the largest numbers of hit-to-miss transitions. The next few subsections briefly discuss selected pathological cases to illustrate their causal behaviors; unfortunately, not all of the interesting cases can be discussed here, due to space limitations.

6.4 Masquerader 02 revealed by unusual flags

Masquerader 2 experienced 26 miss-to-hit transitions and no hit-to-miss transitions. Enriching the data caused this masquerader to go from an 18% detection rate across 50 victims to a 70% detection rate due to the masquerader's use of an unusual command-line flag. Most victims regularly use the commands ls (list files) and cat (view files). Through enrichment, ls became ls -al (list detailed info about all files) and cat became cat -n (view files and add line numbering). These flags are unusual, and naive Bayes noticed this. Enrichment allows unusual behavior in the injection to be seen by naive Bayes, which is precisely the intention of using enriched data. The truncated and enriched blocks of data that revealed the masquerader are shown in Table 7.

6.5 Novel commands conceal Masquerader 07

Masquerader 7 experienced 35 miss-to-hit transitions and no hit-to-miss transitions. Enrichment causes this masquerader to go from a 70% detection rate to a 0% detection rate. The truncated rlogin command (remote login) enriches to rlogin /usr/ucb/rlogin -8 (an alias of rlogin, the -8 means to enable the 8th bit (parity bit) in the network connection). The enriched form of the command was not seen in the training data. Naive Bayes treats never-before-seen-commands as more likely to be self than non-self, because of the higher command-frequency ratio in the nonself data

Truncated	Enriched
ls	ls -al
cat	cat -n
sc	sc -D -o
a5	a5
a5	a5
a5	a5
mail	mail
ftp	ftp
cat	cat
mail	mail

Table 7: Masquerader 2, revealed by unusual flags.

as opposed to the self data; that is, the probability of occurrence is lower in the nonself data. Because 80% of the enriched block was never-before-seen, the block was classified as self. This kind of behavior, typical of naive Bayes, is vulnerable to exploitation by a clever masquerader to elude detection. Note that masquerader 24's similar outcome can be explained by the same mechanism. See Table 8.

Truncated	Enriched
rlogin	rlogin /usr/ucb/rlogin -8
rlogin	rlogin /usr/ucb/rlogin -8
rlogin	rlogin /usr/ucb/rlogin -8
rlogin	rlogin /usr/ucb/rlogin -8
rlogin	rlogin /usr/ucb/rlogin -8
mail	mail
rlogin	rlogin /usr/ucb/rlogin -8
rlogin	rlogin /usr/ucb/rlogin -8
mail	mail
rlogin	rlogin /usr/ucb/rlogin -8

Table 8: Masquerader 7, concealed by novel commands.

6.6 Simplistic commands reveal Masquerader 16.

Masquerader 16 experienced 30 miss-to-hit transitions and no hit-to-miss transitions. Enrichment moves the 24% detection rate to 84%. Yet, enrichment did not reveal anything new about the masquerader's command line. This is not an error, because the masquerader's behavior is too simple for most victims. Most victims have ls (list files) aliased to use some set of preferred flags. The lack of these flags reveals the masquerader to naive Bayes. Counter-intuitively, enrichment can help even when it may not reveal anything new about the injection. Because the truncated and enriched data in this case are the same (the differences were in the training data, not the test data), they are not shown.

6.7 Masquerader 26 revealed by printer choice

Masquerader 26 experienced 30 miss-to-hit transitions and no hit-to-miss transitions. A 34% detection rate in the truncated data turned into a 94% detection rate through enrichment. The lpq command (used to check the print queue length) became lpq -Palw2 through enrichment (check queue on printer named alw2). Many victims check print queues, so lpq is not suspicious. Different victims use different printers, however, and the alw2 specification, revealed through enrichment, is suspicious. In this example, naive Bayes learns indicators of victim preferences (e.g., preferred printers). See Table 9.

Truncated	Enriched
ptroff	ptroff /userc/offstaff/group.bin/lwpp
lpq	lpq -Palw2
lpq	lpq -Palw2
ptroff	ptroff /userc/offstaff/group.bin/lwpp
lpq	lpq -Palw2
e	e emacs
e	e emacs
lpq	lpq -Palw2
ptroff	ptroff /userc/offstaff/group.bin/lwpp
lpq	lpq -Palw2

Table 9: Masquerader 26, revealed through choice of printer.

7 Discussion

The hypothesis that enriched command-line data can enhance detection of masqueraders has been confirmed. The use of enriched data:

- Increased hits by 15.79%
- Reduced misses by 38.53%
- Increased false alarms from 4.7% to 5.7%
- Reduced equal-basis cost of error by 30.02%

The study improved hit rates to 82.1%, which is a 32.2% increase over the previous best masquerade-detection achievement [8], and more than 100% improvement over the best of the results from [12]. This was accomplished using the same number of masquerade victims (50) as in these two previous studies, but with 5 times less training data, and 10 times less information at each decision point (block size of only 10 commands).

Some general observations can be made on the basis of the selected examinations of conspicuous masqueraders. The detector is able to learn subtle user idiosyncrasies that exist in real data, and it can leverage that knowledge in detecting masqueraders who don't share those idiosyncrasies (e.g., a victim uses "ls" less often than the attacker does).

Because of the mathematics of the naive Bayes detector, it is easy to understand why the detector makes the decisions it makes; and, unlike neural nets, its internals are accessible and can be brought out to ancillary processes such as alarm mitigators, displaying evidence to operators. The detector is amenable to fine-grained analysis; that is, one can determine the exact elements of the attacker and victim environments that influence hits, misses and false alarms, hence providing a better comprehension of coverage. The detector works reasonably well on small block sizes (10 commands).

A point worth noting is the vulnerability of the system in respect of unique, previously unseen masquerader command lines. A block containing a high proportion of (or nothing but) previously unseen commands will have an exceedingly low probability of having been generated by either the model of self or the model of non-self. However, due to the larger amount of data in the non-self model, the probability of nonself will be even lower than that of self, and since classification is done on the basis of the ratio of self to nonself, an intruder issuing such a block of commands may beat the detector (depending upon the threshold). One solution to this would be a two-tiered thresholding system, in which an absolute limit on the likelihood of self is applied as a primary filter, before the ratio test. Tolerance of previously unseen commands is important, because the data show a pronounced tendency for users to embark on sudden spurts of new command usage.

8 Conclusion

This work represents an advance over previous masquerade-detection research, both in terms of improved detection statistics, as well as in achieving an understanding of what works and what doesn't work, and why. There is now at least some comprehension of the conditions that lead the detector to failure. Possibly through design diversity (multiple diverse detectors), there will be ways to compensate for these failures, now that at least some details of these failure characteristics are now known in some detail.

9 Acknowledgements

The Defense Advanced Research Projects Agency (DARPA) supported this work under contracts F30602-99-2-0537 and F30602-00-2-0528; thanks to Cathy McCollum for encouraging this research. Tahlia Townsend (now at Yale University) performed the initial data analyses. Kevin Killourhy helped with data management and analysis. Imre Kondor (now at Columbia University) helped with a visualization system for interpreting naive Bayes classification results. This study would not have been possible without Saul Greenberg's gracious contribution of his data.

References

[1] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In L. Saitta, editor, *13th International Conference on Machine Learning (ICML-96)*, pages 105–112, 03-06 July 1996, Bari, Italy. Morgan Kaufmann, San Francisco, California, 1996.

[2] M. Fan. Massive identity theft alleged: Credit fraud affects 30,000, authorities say. San Jose Mercury News, San Jose, California, 26 November 2002.

[3] S. Greenberg. Using Unix: Collected traces of 168 users. Technical report 88/333/45, Department of Computer Science, University of Calgary, Calgary, Canada. 1988.

[4] V. Loeb. Spy case prompts computer search. *Washington Post*, 05 March 2001, page A01.

[5] T. F. Lunt. A survey of intrusion-detection techniques. *Computers & Security*, 12(4):405–418, June 1993.

[6] T. F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *IEEE Symposium on Security and Privacy*, pages 59–66, 18-21 April 1988, Oakland, California. IEEE Computer Society Press, Washington, DC, 1988.

[7] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999. Fourth printing, 2001.

[8] R. A. Maxion and T. N. Townsend. Masquerade detection using truncated command lines. In *International Conference on Dependable Systems & Networks*, pages 219–228, 23-26 June 2002, Washington, DC, IEEE Computer Society, Los Alamitos, California, 2002. .

[9] R. A. Maxion and T. N. Townsend. Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability*, In press, 2003.

[10] A. McCallum and K. Nigam. A comparison of event models for naive 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.

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

[12] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):58–74, February 2001.

[13] E. D. Shaw, K. G. Ruby and J. M. Post. The insider threat to information systems: The psychology of the dangerous insider. *Security Awareness Bulletin*, 2-98, Department of Defense Security Institute, Richmond, Virginia. September 1998.

[14] J. Swets and R. Pickett. *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory*. Academic Press, New York, 1992.