SRI International

Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)[1]

Debra Anderson
Teresa F. Lunt
Harold Javitz
Ann Tamaru
Alfonso Valdes

Computer Science Laboratory
SRI-CSL-95-06, May 1995

[1]This report was prepared for Trusted Information Systems, Mountain View, California, Contract No. 910097C under RomeLab Contract F30602-91-C-0067.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As part of the Audit Data Analysis task of the Safeguard project, SRI has been under contract to Trusted Information Systems (TIS) to evaluate the ability of the Next-generation Intrusion Detection Expert System (NIDES) to detect the use of computer systems for other than authorized applications.

The principal goal of the Safeguard Audit Data Analysis task was to explore methods of detecting and reporting attempts by those whose use of exported computer systems is restricted to approved applications or classes of applications, to bypass or defeat those software execution restrictions. As part of this task, audit trail analysis techniques were considered as potentially useful in detecting misuse of restricted technology.

To date, NIDES has been used to monitor computer users by examining audit trail information using a custom non-parametric statistical component as well as a rule-based component. In the paradigm explored by SRI [1, 2, 4, 5], the subjects considered up to now have been computer users. For the present study, the methodology is adapted to consider applications as subjects. This directly addresses the stated goal of monitoring usage to detect unauthorized use of applications or application classes on restricted systems. With recent changes in export restrictions, this goal is perhaps less urgent now than when the study was undertaken. Nonetheless, the methodology as adapted here is useful in the detection of Trojan horses and other masquerading applications.

This report presents the findings of SRI's analysis of the NIDES statistical analysis component's ability to detect unauthorized application execution based on system-level audit trails.

1.1 Project Description

SRI developed a set of audit analysis techniques to support detection of the use of computer systems for other than authorized applications in environments in which users are permitted to develop and install their own application software. The analysis software developed for this project was a straightforward adaptation of the current NIDES software, specifically the NIDES statistical analysis component.

Our efforts were divided into the following general tasks:

In conjunction with TIS, we determined the available and appropriate audit data sets that could be collected for analysis purposes. As part of this process TIS converted the data into the NIDES audit record format. TIS provided all the audit data we used in developing profiles of acceptable application behavior and performing experiments in detecting abnormal activity.

We modified the NIDES statistical analysis software to support our experiments. Most of our modifications were to support the use of a set of statistical measures suitable for analyzing the audit data provided by TIS. These modifications included code changes as well as changes to the NIDES configuration files associated with the statistical analysis component.

We performed five complete experiments:

Three experiments - concept, verification, and refinement - were run to determine the most suitable configurations for the Safeguard application of NIDES. Each of these first three experiments included the following steps:

1. We set up a predetermined set of configurations for the experiment. Each experiment we ran had a different configuration.
2. We ran TIS-supplied audit data through the statistical analysis component of NIDES to develop baseline profiles of applications.
3. We performed false-positive tests on the profiles developed to determine if the profile was stable enough to be used in subsequent tests.
4. We ran tests using data that included masquerading applications to perform true-positive testing.
5. We reviewed the results of the experiment to determine the appropriate configuration for subsequent experiments. Our goal was to improve NIDES performance as our experiments progressed.

- For our fourth experiment we ran a cross-profiling experiment to perform additional true-positive testing, using the configuration setting of our refinement experiment.
- Lastly, we performed experiments with application groups using information gathered during the cross-profiling experiment.

Overall our results from the first (concept) experiment to our third (refinement) experiment improved. We had higher detection rates and generally lower false-positive rates. The cross-profiling experiment gave us additional insight into similarities of the applications in our test set and how easily one application in the test set could masquerade as another application in the test set. We also gained some understanding of the results one obtains when profiling application groups instead of individual applications - in general, the ability to detect anomalous activity (detection sensitivity) seemed to decrease when we worked with application groupings.

1.2 NIDES Statistical Algorithm Description

In this section, we provide an overview of the NIDES statistical algorithm, for a more detailed description, see [3].

The statistical approach used in NIDES is to compare a subject's short-term behavior with the subject's historical or long-term behavior. In the traditional NIDES framework, a subject is a user of a computer system. In the Safeguard context, a subject is an application. In comparing short-term behavior with long-term behavior, the statistical component is concerned both with long-term behaviors that do not appear in short-term behavior, and with short-term behaviors that are not typical of long-term behavior. Whenever short-term behavior is sufficiently unlike long-term behavior, a warning flag is raised. In general, short-term behavior is somewhat different from long-term behavior, because short-term behavior is more concentrated on specific activities and long-term behavior is distributed across many activities. To accommodate this expected deviation between short-term and long-term behavior, the NIDES statistical component keeps track of the amount of deviation that it has seen in the past between a subject's short-term behaviors and long-term behaviors. The NIDES statistical component issues a warning only if the current short-term behavior is very unlike long-term behavior relative to the amount of deviation between these types of behaviors that it has seen in the past.

The NIDES statistical approach requires no *a priori* knowledge about what type of behavior would result in compromised security. It simply compares short-term and long-term behaviors to determine whether they are statistically similar. This feature of the NIDES statistical component makes it ideally suited for the Safeguard task. A masquerading application does not compromise security in any of the typical ways, but nevertheless is an event that is anomalous and warrants detection. If the masquerading application has different characteristics from the application whose identity is being expropriated (and the short-term behavior for the application whose identity is being expropriated is not too erratic), the NIDES statistical component should be able to recognize these differences and identify the masquerade.

1.2.1 Measures

Aspects of subject behavior are represented as measures (e.g., file access, CPU usage, hour of use). For each measure, we construct a probability distribution of short-term and long-term behaviors. For example, for the measure of file access, the long-term probability distribution would consist of the historical probabilities with which different files have been accessed, and the short-term probability distribution would consist of the recent probabilities with which different files have been accessed. In this case, the categories to which probabilities are attached are the file names. In the case of continuous measures, such as CPU time, the categories to which probabilities are attached are ranges of values, which we sometimes refer to as "bins" (these ranges of values are mutually exclusive and span all values from 0 to infinity). We refer to the collection of measures and their long-term probability distributions as the subject's profile.

We have classified the NIDES measures into four groups: activity intensity, audit record distribution, categorical, and continuous. These different types of measures serve slightly different purposes. The activity intensity measures determine whether the volume of activity generated is normal. The audit record distribution measure determines whether, for recently observed activity (say, the last few hundred audit records generated), the types of actions being generated are normal. The categorical and continuous measures determine whether, within a type of activity (say, accessing a file), the types of actions being generated are normal.

1.2.2 Half-life

The number of audit records or days of audit record activity that constitute short-term and long-term behavior (i.e., that show up in the short-term and long-term probability distributions for a measure) can be set through the specification of a *half-life*. For the long-term probability distributions, we typically set the half-Iife at 30 days. With this setting, audit records that were gathered 30 days in the past contribute half as much weight toward the probability distribution as do the most recent records, audit records that were gathered 60 days in the past contribute one-quarter as much weight, and so forth. Thus, the most recent days of activity contribute more than more distant days of activity, and eventually the long-term profile "forgets" about very distant behavior.

The concepts of aging rate, half-life, and effective length of a profile are related as follows. The aging rate is a multiplicative factor less than or equal to unity, by which the existing information in a profile is aged. The smaller the rate, the more rapidly this information is "forgotten". The half-life is the number of audit records that need to transpire before the contribution of a given data item is decayed (downweighted) by one half. For example, if the aging rate is 0.8, the third most recent audit record has a weight of 0.8*0.8*0.8 or 0.512, so we would say that an aging rate of 0.8 corresponds to a short-term profile half-life of three audit records. The effective length of a profile is given by the series sum of all powers of the aging factor, which will converge if this factor is strictly less than 1. For example, an aging rate of 0.8 corresponds to an effective profile length of 5 audit records. As a rule of thumb, the effective length of the profile (in audit records) is approximately 1.5 times the half-life.

1.2.3 Differences between Long- and Short-term Profiles – The *Q* Statistic

The degree of difference between a long-term profile for a measure and the short-term profile for a measure is quantified using a chi-square-like statistic, with the long-term profile playing the "role" of the actual probability distribution and the short-term profile playing the "role" of the observations. We call the resultant numerical value Q. (There is a different Q value for each measure.) Large values of Q mean that the long-term and short- term profiles for the measure are very different from one another and therefore warrant some suspicion; a Q value of zero means that they agree completely.

In calculating the Q statistic,, we need to refer to the categories that have been defined for each measure. As with the chi-square statistic, it is desirable that categories that are rarely or never observed in the long-term distribution be collapsed together. In NIDES we have defined a parameter that controls how many categories are collapsed together. This parameter sets an upper limit on the combined probability of all category bins that are considered rare for scoring purposes. We continue to combine categories into the "rare" category, starting with the category with the smallest long-term probability, as long as the total long-term probability of all of the collapsed categories is less than the value for this parameter.

For each audit record, the NIDES statistical component generates a vector of Q values, with large values indicating suspicious activity. But, how large a Q value is too large? Unfortunately, it is not possible to refer Q directly to a chi-square table of cutoff values to determine if the difference between the two probability distributions is statistically significant. The traditional chi-square method cannot be applied in environments where the audit records that contribute to the short-term profiles are not independent, nor in systems where the short-term profiles are based upon very few audit records. Since the distribution of Q is not chi-squared, we need to track its values to determine what its distribution looks like. Every time another audit record arrives, a new value of Q is generated. We observe these values over a substantial portion of the profile-building period. We begin to observe Q values during the second stage of the profile-building period, which commences as soon as we have constructed reasonably stable historical profiles for the observed categories. We use these values for Q to build a long-term probability distribution for Q. (Each measure has its own Q and a long-term distribution for that Q statistic.) The Q distributions look somewhat like long-tailed and stretched-out chi-square distributions.

1.2.4 Scoring Anomalous Behavior - The *S* and T^2 Statistic

As with all other long-term distributions in NIDES, the long-term distribution of Q is categorized into ranges of values (called bins). From this distribution, we can obtain the tail probability of obtaining a value for Q at least as great as the observed value. We transform this tail probability and denote the transformed variable as S, and define the transformation so that S has a half-normal distribution. (A half-normal distribution looks like the right-hand side of a normal distribution, except that the height of the probability distribution is doubled so that there is still unit area under the curve. This is also the distribution of the absolute value of a normally distributed variable.) The mapping from tail probabilities of the Q distribution to half-normal values requires only a table of normal probabilities. For example, if the current Q statistic is at its 80th percentile (i.e., the tail probability is 20%), then the corresponding value for S is 1.28 (i.e., the same as the 90th percentile of a normal distribution). If the current Q statistic is at its 90th percentile, the corresponding value for S is 1.65 (i.e., the same as the 95th percentile of a normal distribution).

As each audit record is received, it is possible to generate the corresponding vector of S values. High S values correspond to measures that are unusual relative to the usual amount of discrepancy that occurs between long-term and short-term profiles. Small S values correspond to measures that are not unusual relative to the amount of discrepancy that typically occurs between long-term and short-term profiles. We combine the S scores into an overall statistic that we call T^2. The T^2 statistic is a summary judgment of the abnormality of many measures, and is in fact the sum of the squares of the S statistics (i.e., $S_1^2 + S_2^2 + ... + S_m^2$ where there are *m* measures).

Because each S statistic follows the same half-normal distribution (when the audit records being observed come from the subject who owns the account), the T^2 statistic is approximately distributed like a chi-squared variable with *m* degrees of freedom. However, the S_i are not necessarily independent, so rather than rely on the chi-square distribution to provide threshold values for T^2, we build a long-term distribution for T^2. We then observe the upper threshold values for T^2. (The long-term distribution for T^2 is built during the last stage of the profile building period, after reasonably stable long-term distributions for the Q statistics have been constructed.) We declare recent audit records to be anomalous at the "yellow" level whenever the T^2 value is over the 1% value of the long-term profile for T^2, and at the "red" level whenever the T^2 is over the 0.1% of the long-term profile.

Chapter 2

Customizations for Safeguard

In preparation for our experiments, we made some modifications to the NIDES statistical analysis component software and examined the various configurable items to determine the best configurations to use under the Safeguard context. These changes were needed because of three primary differences between the Safeguard and Sun-C2 applications of NIDES.

First, the volume of audit data under the Safeguard application was substantially less than the volume of data we see under the Sun-C2 environment. The number of audit records generated in the Safeguard application for a single activity is far fewer than the number in traditional NIDES applications for that same activity. Under BSM or C2 auditing, a single user will generate as many audit records in a single day as were obtained for the application with the most audit records over a single month in the Safeguard data.

Second, the nature of the Safeguard audit data is different in that it captures, in a single audit record, all the information for a single instance of an application use, while within the Sun-C2 context we would build the analogous information from a large number of audit records, each of which might affect one or a few of the possible measures. With the C2 audit data it would take tens or hundreds of audit records to represent'a single user activity, whereas in the Safeguard audit data each use of an application is reflected in a single audit record. Lastly, the safeguard data had applications, rather than users as subjects.

Furthermore, the objective of the Safeguard study was to flag small numbers or even single instances of anomalous activity, whereas Sun-C2 was tuned to detect a time interval (corresponding to hundreds of audit records or more) of such activity.

2.1 New Measures

The set of measures was modified to include behavior aspects that pertain specifically to applications, such as process time, detailed CPU and memory usage, and page faults. All of these new measures were continuous types.

It was decided upon consultation with TIS not to consider some measures, such as swap activity, that vary largely, due to system load and other factors external to the instance of the execution of the particular application. The complete list of measures used for the Safeguard task is described below. New measures are indicated with an asterisk (*).

UCPU* (continuous). User CPU time.

SCPU* (continuous). System CPU time.

IO (continuous). Amount of character IO during the application execution.

MEMCMB* (continuous). Combined process size - that is, the maximum size of the process during execution.

MEMUSE* (continuous). The integral of real memory usage over time, in units of KByte*seconds.

TEXTSZ* (continuous). The virtual size of the text segment.

OPENF* (continuous). The total number of ties opened during execution.

PGFLT* (continuous). The number of major and minor page faults.

PGIN* (continuous). The number of pages read in, a measure of disk IO.

PRCTIME* (continuous). Elapsed time in seconds - that is, the exit time minus the start time of the application.

SIGNAL* (continuous). Number of signals received during the execution of the application.

UID (categorical). Name of new user name (if the user name was changed during execution).

UIDB (binary). Whether the user name was changed during execution.

HOUR (categorical). Hour (O-23) in which activity was done.

RNET (binary). Whether the application executed was on a remote network host.

RNETHOST (categorical). Name of remote network host on which the application was invoked.

RNETTYPE (categorical). Name of application that was invoked on a remote network host.

LNET (binary). Whether the application executed was on a local network host.

LNETHOST (categorical). Name of local network host on which this application was invoked.

LNETTYPE (categorical). Name of application that was invoked on a local network host.

INTARR (continuous). Number of seconds elapsed since the last audit record for this application.

I60 (continuous/intensity). Rate of audit record activity every 60 seconds (1 minute).

I600 (continuous/intensity). Rate of audit record activity every 600 seconds (10 minutes).

I3600 (continuous/intensity). Rate of audit record activity every 3600 seconds (1 hour).

Only one module in the statistical component was changed to support these new measures, namely the function that converts the audit data into activity records. Table 2.1 lists the relevant data fields TIS provided in the audit data (NIDES format) and the corresponding measures to which the fields were mapped.

2.2 Configuration Items

A number of parameters within the statistical component can be configured to support any particular system. On the following list, those items marked with an asterisk were modified according to the particular experiment(s) we ran. These modifications are discussed in detail in Section 3.

- *Short-term profile half-life* *. This is the half-life of the short-term profile, measured in audit record counts.
- *Long-term profile half-life.* This is the half-life of the long-term profile measured in days.
- *Threshold percentages (yellow, red).* These percentages respectively represent the warning and critical thresholds at which anomalous activity is reported.
- *Minimum effective-N.* This is the effective minimum number of audit records necessary for a particular measure to be considered "trained" so that it may contribute to the overall scoring.
- *Training days.* This is the number of days required for the long-term profile to be trained. Scores produced by the statistical component during this training period are considered unstable and are thus ignored.
- *Scalar Values.* These numbers are used to scale observed values for continuous measures in order to assign the values to bins. The scalar values estimate a value beyond the maximum value that we ever expect to see for a continuous measure (i.e., the mean value plus four standard deviations). A scalar value was calculated for each continuous measure.

NIDES Audit Field	TIS Data Description	Corresponding Measure
atime	Exit Time	HOUR, INTARR I60, I600, I3600
hostname	'SOL"	RNET,RNETHOST,RNETTYPE LNET,LNETHOST,LNETTYPE
action	"IA_EXIT"	(None)
auname	command name	(NIDES subject name)
ouname	(empty)	UID, UIDB
pid	Integral of Memory Usage	MEMUSE
ttyname	Terminal Name	(None)
argcount	Text Size	TEXTSZ
syscall	Open Files	OPENF
errno	Page Faults	PGFLT
rval	Page Ins	PGIN
res_utime	User Time	UCPU
res_stime	System Time	SCPU
res_rtime	Elapsed Process Time	PRCTIME
res_mem	Combined Process Size	MEMCMB
res_io	Character IO	IO
res_rw	Signals	SIGNAL

Table **2.1:** Safeguard Audit Data Mappings to Measures

Q Max values. These numbers are used to scale Q values for all measures. These values play the same role for Q scores that the scalar values play for raw observations of continuous measures.

*Modes of operation**. The statistical component can be run in various modes, such as training or testing mode. One common testing mode is to disable the long-term profile updater, so that audit records are scored against trained profiles, but the profiles do not adapt to the new data.

2.3 Half-life of Short-term Profile

For monitoring of Sun audit data, we have set the half-life of the short-term profile at 250 audit records. Because the Sun auditing facility can generate tens of thousands of audit records per day, this means that the short-term profile is based (for all practical purposes) on minutes to at most hours of the most recent activity. In contrast, the Safeguard audit records are much more "concentrated"; a single audit record corresponds to the same amount of activity that might result in the generation of hundreds of Sun audit records (i.e, with the Sun audit data, a given user activity is represented by tens of hundreds of audit records, while with the Safeguard data, each activity is reflected in a single audit record). For this reason, and because we wanted to identify single audit records that were suspicious, we reduced the short-term profile half-life very substantially. In fact, most of our results are based on short-term profiles that consist of data from a single audit record. (In this case the short-term probability distribution has only one lump of probability in the category in which the single audit record occurred.) The short-term profile half-life length should be comparable in length to the stream of audit data (i.e., the number of audit records) that represent a single subject activity.

2.3.1 Number of Category Bins Reduced to 16

The NIDES statistical component generates category distributions for all measures. For categorical measures, there is a one-to-one correspondence between observed categories and category bins in the NIDES profile. For continuous measures, the range of values is broken up into a predefined number of category bins. These are scaled based on an observed maximum value (the scalar) so that the range of values is logarithmically assigned in a way that the ratio of the first to the last bin endpoint (the dynamic range that the profile can capture) is approximately 1000. For example, if 10 bins are available, the right endpoint of the 10th is chosen to be the data mean plus three to four standard deviations, and then each bin endpoint proceeding downward is obtained by halving the next higher. This gives a factor of ten powers of 2 (or 1024), as desired. The binning procedure uses fractional powers of two for any number of bins other than 10. This mechanism is sufficiently robust that the scaling maxima need not be accurately estimated; errors of a factor of 2 or even 4 are easily tolerated.

The Sun-C2 application of NIDES classifies continuous data into 32 bins with "third root of two" bin spacing, so that 30 bins give the desired dynamic range with some overflow bins left. This is appropriate with an effective profile length of hundreds of audit records, as on the average a reasonable number of the category bins will be populated. However, in the Safeguard application, we have very short effective profile lengths and the short-term profile has a very "lumpy" distribution (indeed, only one bin is touched per measure if the short-term profile consists of a single audit record). For this reason, we reduced the number of bins into which continuous measures are classified to 16 with the appropriate fractional logarithmic spacing to give a dynamic range of 1000 with some overflow bins. Examination of the long-term profiles generated using 16 bins indicated a reasonable population of a broad middle range of the bins and sparse or no population of the extremes, so that we believe the range, scaling, and number of bins were appropriate for the Safeguard data. The number of Q bins was also reduced from 32 to 16.

Scalars and QMaxes As part of the binning process, scalar and QMAX values are used to determine bin locations. To get the desired range of binning for continuous measures, we scanned all audit data files and noted the mean and variances of the observed values for each continuous measure. From these values, we computed the standard deviation and set the scalar value to be somewhere between 3 to 4 times the standard deviation (we ended up rounding off scalar values). Observed values were then placed in bins based upon this scalar value according to the approach discussed above.

Obtaining QMAX values required a small amount of statistical processing. The first stage of the training period was allowed to complete (typically in ten days worth of data) and then a day's worth of Q values were collected across all subjects. The QMAX value was then set at the Q score mean plus 3 to 4 standard deviations.

2.4 Statistical Component Wrapper

A special "wrapper" was developed to isolate the statistical component for experimentation (the current NIDES system includes a rulebased component, a resolver, and a user interface). This wrapper simplified the customization process of statistical configuration, and also enabled a relatively quick turn-around time for setting up and running experiments.

This wrapper is similar to the batch analysis version of NIDES, but does not contain the rulebased component. It can be invoked directly from the Unix shell, and reads NIDES audit records directly from a file or a list of files.

More than one option can be specified in one invocation of the wrapper, which has the following features:

Save results to file (-xscores option). By default, the wrapper dumps results onto the screen. The *-xscores* option makes the wrapper save out all the results (subject id, timestamp, audit record sequence number, score value, and alert level) to a file. The saved data can subsequently be used to plot histograms, graphs, and so forth.

T^2 summary data collection (-xt2 option). For the majority of our experiments, it is important to save the final distribution of T^2 scores to determine the rate of true or false positives. This option basically totals up the final count of anomalous records over the 1% and 0.1% (yellow and red) threshold levels, and calculates the percentage of these anomalous records over the total number of records for each subject.

Timestamp ranges (-t and -T options). With these options, the wrapper can process audit records within a specific time range. Timestamps are represented in the Unix long integer form. For example,

```
sgstat -t 742571460 -T 743113260
```

would mean to read audit records between the ranges of 7/13/93 06:51:00 through 7/19/93 13:21:00. Either one or both of the *-t* or *-T* can be specified. If neither of these options is specified, then all records will be processed.

Specific configuration files (-c option). This option allows the wrapper to read a prespecified configuration file (the default is called *IDES_statconfig).* The following example shows how to invoke the wrapper using a specific configuration file called *my_ides_config:*

```
sgstat -c my_ides_config
```

Specific instance (-i option). This option allows the wrapper to use a particular instance (i.e., set of profiles) for processing. Both this option and the *-c* option are useful when running parallel experiments. The following example shows how to invoke the wrapper using the instance called *my_instance:*

```
sgstat -i my_instance
```

Specific set of subjects (-S option). This option allows the wrapper to process audit records for a given list of subjects. The name of the file containing the list of relevant subjects must be specified, as follows:

```
sgstat -S test_subjects
```

2.5 Utilities

A number of software utilities were used or developed to aid in the execution and analysis of our Safeguard experiments. Many of these utilities already existed in some form, and were modified to accommodate the particular needs of this project.

arcount. This utility was used during the audit record analysis task. It reads a list of audit data files (in NIDES format) and summarizes the audit record distribution by printing out how many audit records were available for each subject on a daily basis. This utility allowed us to determine which subjects (applications) were eligible for our experiments (see Section 3.1).

tchange. This utility was used throughout our experiments. It shifts the timestamps in a given NIDES audit data file in any direction (later or earlier). This tool allowed us to select portions of the various data files and concatenate them together in a desired chronological order to represent continuous activity.

data_make. This utility was used to create masquerader data. Given an input file, output file, subject name that will be masquerading, and the subject it will pretend to be, this utility changes the identity of selected audit data. Audit record order is preserved and the resulting data set is written to a specified file.

change_prof. This utility changes the timestamp of any given profile, and also changes its subject name. The utility was particularly useful for the cross-profiling experiments (see Section 3.7), as well as for synchronizing profile dates to correspond with a particular audit data set.

dump_prof. This utility dumps out entire profile contents (both short-term snapshots and long-term) of any given subject. The contents can either be displayed on the screen or saved into a file.

bins and whichbins. These utilities were used to analyze continuous measures. *Bins* prints out the ranges of the bins given a maximum value for scaling, and *whichbins* indicates which bin a given value would fall into. These utilities were used as an aid to create profile histograms.

We also used a few COTS applications, such as *xgraph* and *wingz,* to create graphs and charts.

Chapter 3

Experiment Descriptions

TIS supplied approximately three months of audit data to SRI. We performed five major experiments using this data.

3.1 Audit Data Analysis and Organization

As part of our experiment preparation, we analyzed the audit data to ascertain the overall characteristics of the data (volume, daily activity, distribution of activity). From this preliminary analysis, we determined which applications were good subject candidates and which audit data fields (hence measures) were sufficiently available to conduct our experiments.

We organized the TIS audit data into three parts. Table 3.1 summarizes the audit data sets we used for our experiments. The first data set was used to train NIDES historical profiles, that is, historical bin probabilities, the empirical distribution of audit record Q and T^2 scores, and the yellow and red thresholds of the T^2 statistic. As it was essential to collect enough volume and variety of data to incorporate each subject's full spectrum of behavior into its profile, we used (approximately) two months of activity.

The second data set supplied previously unseen application data that was scored against that application's historical profile, and the rate of exceedances above the threshold tabulated. Such exceedances are termed "false-positives" because they represent records identified by NIDES as being above a detection threshold when it is known that they are not masqueraders. This is as opposed to "true-positives," that is, application records that when scored against another application's profile exceed the detection threshold. The false-positive rate for NIDES or any detection system that learns patterns should always be estimated using data not used in the learning phase, as we do here.

The third data set was used to conduct true-positive tests. TIS had indicated to us where two resource-intensive applications were being monitored, and we selected approximately 2 to 3 weeks of data surrounding these resource-intensive activities. We then created our own masquerader data by merging these anomalous records, with an appropriate name change, into each subject's original stream of audit data (using the same data that was used for the false-positive testing). Two data sets were created for true-positive testing — one with four audit records from one resource-intensive application and one with one audit record from another resource-intensive application.

Audit Data Set	Dates Represented	Number of Records
Training Data	6/22/93 to 7/19/93	72115
	8/2/93 to 9/23/93	72115
False-positive Test Data	7/19/93 to 8/2/93	20722
True-positive Test Data # 1	7/21/93	4
True-positive Test Data # 2	7/24/93	1

Table 3.1: Audit Data Set Summary

Subject Selection

After the data sets were constructed, we went through a subject selection process to determine which applications meet the following criteria and thus could be used for our experiments:

- The training data set must have at least 200 records for the application.
- The application must have at least 30 unique days of activity.
- The false-positive data set must have at least 100 records for the application.

Out of 288 applications available in the data sets, 30 met the above criteria. Table 3.2 lists the applications we selected. Those subjects marked with an asterisk (*) fell a little short of the criteria, but were close enough for inclusion in the experiments.

3.2 Overview of Experiments

We performed five major experiments during our analysis. The first three experiments were run to determine the best possible configuration for NIDES under the Safeguard concept. These three experiments evolved as we progressed, where each iteration provided additional feedback to determine subsequent experiment configurations.

The following experiments were conducted:

Subject (Application)	Total Records	Training Records	Testing Records	Unique Days
as	1688	1539	149	39
cat	1195	1058	137	68
ccom	886	736	150	36
compile	1010	838	172	43
cp	378	273	105	60
cpp	2625	2470	155	44
csh	909	709	200	57
diff *	690	596	94	46
discuss	1328	1040	288	60
emacs	7929	6227	1702	84
finger *	619	537	82	78
fmt	1819	1522	297	64
gawk *	613	530	83	56
getfullnm *	353	269	84	52
ghostview *	320	225	95	50
grep	5685	3474	2211	60
latex	928	758	170	52
less	5409	4709	700	80
ls	9020	7368	1652	78
mail *	613	527	86	60
make	1251	1095	156	52
man	938	708	230	60
more	8015	6497	1518	68
mymoreproc	3901	3406	495	77
pwd	1405	1181	224	62
rm	2539	2184	355	82
sed	1801	1464	337	64
sort	891	702	189	61
stty	1003	871	132	68
vi	5452	4663	789	77

This table lists the number of audit records used for our experiments for each application profiled. The training record counts represent the number of records that were used to train profiles; the test record counts represent the number of records that were used during the false-positive testing phase of the experiments. The column showing unique days lists the number of days for each subject for which any audit record data were available.
Note: * indicates a borderline subject.

Table 3.2: Experiment Subjects and Numbers of Audit Records

- *Concept Experiment.* This experiment served as our baseline using default NIDES configuration settings. We used 24 measures (see Table 3.3), and short-term profile half-life settings of 50, 20, 10, and 1 audit record(s).
- *Verification Experiment.* Based on the results of the concept experiment, we made adjustments to the statistical parameters (such as turning off measures, adjusting half-lives), and reran the same set of tests.
- *Refinement Experiment.* Based on the results of the verification experiment, we made final adjustments to the configuration, and completed the same procedure as the first two experiments.
- *Cross Profiling.* With the configuration used during the refinement experiment, we examined the rate of anomalous records as a result of running each application's audit records through another application's profile.
- *Group Profiling.* Based upon the analysis done in the cross-profiling experiment and performing a historical profile overlap analysis, we explored potential application groupings. We conducted a few sample groupings to evaluate whether such groupings are cohesive and useful.

Table 3.3 indicates the measures that were used for each of the five experiments. Note that only the first experiment used the original 24 measures defined for the project, and the remaining four experiments used a configuration consisting of a subset of 12 measures.

Because the procedures followed for the first three experiments were essentially the same, those procedures will be described first, then the results of each experiment will be discussed. Following these discussions will be descriptions of the remaining two experiments: the cross profiling and grouping experiments.

3.3 Procedures of Concept, Verification, and Refinement Experiments

As mentioned earlier, we performed three sets of experiments to determine the best possible configuration for NIDES under the Safeguard environment. Table 3.4 summarizes the various configurations used for the concept, verification, and refinement experiments.

The procedures used during the concept, verification, and refinement experiments were divided into three main steps: profile building, false-positive testing, and true-positive testing.

3.3.1 Profile Building

The baseline set of subjects used for the experiments is listed in Table 3.2. To conduct this phase of the experiment, we cycled through the training data set (see Section 3.1) three times to simulate

Measure	Concept	Verification	Refinement	Cross-profiling	Grouping
UCPU	X	X	X	X	X
SCPU	X	X	X	X	X
IO	X	X	X	X	X
MEMCMB	X	X	X	X	X
MEMUSE	X	X	X	X	X
TEXTSZ	X	X	X	X	X
OPENF	X	X	X	X	X
PGFLT	X	X	X	X	X
PGIN	X	X	X	X	X
PRCTIME	X	X	X	X	X
SIGNAL	X	X	X	X	X
UID	X				
UIDB	X				
HOUR	X	X	X	X	X
RNET	X				
RNETHOST	X				
RNETTYPE	X				
LNET	X				
LNETHOST	X				
LNETTYPE	X				
INTARR	X				
I60	X				
I600	X				
I3600	X				

Table 3.3: Measures Used for Experiments

Experiment	No. of Measures	Half-lives	Significant Differences in Experiments
Concept	24	50,20,10,1	Standard NIDES configuration
Verification	12	2,1,0.1	Reduced number of measures; smaller short-term profile half-lives
Refinement	12	2,1,0.1	Rare probability sum reduced from 10% to 1%; nonobservations counted

This table summarizes the configurations used during our first three experiments: concept, verification and refinement. The half-life column lists the short-term profile half-life configurations used. The significant changes column highlights the major differences between the experiments.

Table 3.4: Safeguard Experiment Configuration Summary

approximately 180 days (6 months) worth of data. We felt that this was sufficient to produce reasonably stable historical profiles for each applications. These profiles were then saved to serve as baseline profiles for the false-positive and true-positive tests.

The following configurations were used during all the profile-building processes for all experiments where profiles were developed:

- False-positive thresholds were set to 1% (warning/yellow) and 0.1% (critical/red).
- The long-term half-life was set to 20 days.
- The profile-training period was set to 20 days.
- The minimum effective- N was set to 100 records. This value is essentially the minimum number of observations required for training before a measure can contribute to the statistical scoring.

·From the historical profiles that were created, we obtained the yellow and red score thresholds and recorded them into tables. These are the score thresholds used in determining the false-positive and true-positive rates.

Because the threshold values are empirically determined, it is not possible to infer something about the subject's behavior by comparison to, say, a suitably normalized chi-square. Furthermore, with longer profile half-lives, the training data set contains a number of audit records that is only a few times as long as the effective short-term profile length (see the glossary for a description of profile length). For example, with a short-term profile half-life of 50 records (which corresponds to an effective short-term profile length of about 75), a training data set of 300 records contains only four independent realizations of the short-term profile. This renders estimates of the thresholds, particularly the red threshold (99.9 percentile) somewhat unstable.

For these reasons, comparing the score threshold values for an application from one half-life configuration to another does not give useful information. Within a given configuration (i.e. a set of measures and a single half-life), it is qualitatively true that the subjects with higher thresholds are more forgiving of unusual activity, so that applications masquerading as these subjects would be harder to detect.

3.3.2 False-positive Analysis

Our false-positive tests were performed using a set of never-seen-before data (approximately two weeks worth), that is, data not used as part of the training process. Using the trained profiles built earlier, we processed this new data and recorded the total number of records that exceeded the yellow and red thresholds. This number was then converted into a percentage that we refer to as the *false-positive rate* for that subject.

The false-positive rate should be low, since we expect a subject's behavior to be consistent with its past behavior. The false positive rate obtained from data unseen by the training mechanism will generally be higher than the nominal rate NIDES uses when setting its threshold values. As such, we expect to see rates somewhat above the nominal 1% and 0.1% false positives (as expected for our nominal yellow and red threshold values). However, rates significantly higher than these indicate some change in the nature of activity for the subject in question between the training and test set (which we sometimes refer to as an "unstable" profile). A change in behavior is indicated by a qualitative change in the graph of T^2 versus time (Figure 3.1 in Section 3.4.5 is an example). Subjects with unacceptable false-positive rates were identified at this step.

For verification purposes, we also ran an additional false-positive test using the training data set that was used to build the profiles. We expected to find, and did find, that the false-positive rates for the training set were very close to nominal levels (i.e., 1% and 0.1%), except when there were insufficient numbers of audit records in the training set.

During the false-positive test, long-term profile updating was disabled so that potentially anomalous data would not be incorporated into the historical profile.

3.3.3 True-positive Analysis

We used the results of the false-positive tests to determine which subjects had reasonably stable profiles with which to perform true-positive tests. The purpose of true-positive testing was to see if or how soon anomalous records could be detected using the trained profiles created in the earlier profile-building phase.

We had two data sets available for the true-positive testing (see Section 3.1). The first data set contained four known anomalous records (produced by the same resource-intensive application), and the second data set contained one known anomalous record produced by another resource-intensive application.

For each subject we obtained a "score" value for each masquerading audit record and determined if that score was above the red (0.1%) threshold, above the yellow (1%) threshold, or below both thresholds. If the score was above either boundary we considered it a successful detection. For each application used in the true-positive test, we indicated for each anomalous audit record whether the record was detected as being abnormal (Table 3.7 in Section 3.4.5 is an example).

3.4 Concept Experiment

3.4.1 Configuration

Our baseline experiment was conducted using the following configurations:

- Twenty-four measures were used (see Table 3.3).
- Four different half-lives for the short-term profile were used: 50, 20, 10, and 1. In contrast to the tens of thousands of audit records that are generated in a Sun-C2 UNIX environment, Safeguard audit data records are much more concentrated; a single audit record corresponds to a complete instance of an activity. For this reason, and because we wanted to identify a handful of (or single) audit records that were suspicious, we started our baseline with significantly shorter half-lives when compared to the half-lives of hundreds of audit records used for the Sun-C2 data.

Using these configurations, profiles for 30 applications were built. Table 3.5 shows the score thresholds for the applications profiled.

3.4.2 False-positive Test Results

Table 3.6 shows the resultant false-positive rates for the applications profiled. The false-positive rates were unacceptably large for a number of applications (e.g., `ccom`, `cp`, `ghostview`, `mail`, `man`, and `sed`), particularly when the short-term profile half-lives were large (i.e., 50 or 20). The longer half-lives (50, 20, and even 10) displayed much higher false-positive rates because the longer half-lives split the data into too few independent observations for statistical stability meaning that the short-term profile (i.e, there were only a small number of independent realizations of the short-term profile - see Section 3.3.1 and the glossary for discussion of profile realizations).

This aspect of the data (i.e., decreasing false-positive rates with decreasing half-life rates) also caused us to conclude that there were small shifts in the behavior of the subjects during the testing period. It is these kind of shifts that are most easily detected when larger half-life configurations are used.

To establish the existence of pattern shifts, we looked at plots of T^2 (the detection score) versus time. Figure 3.1 shows such a plot for `ccom`, in which there is an apparent shift in behavior toward the last part of the graph. This plot indicates a period in the testing data when there were marked changes in the subject's behavior.

We discussed this finding with TIS staff, and they could offer no explanation of why such changes would occur. We decided that these changes were sufficiently small with the shorter half-lives and that we could continue on to true-positive testing if we used shorter half-life (on the order of one audit record) configurations. In addition, given the objective of this project, such a reduction in half-life would be desirable for reasons other than ensuring a reasonable false-positive rate. However, five applications (`ccom`, `cp`, `ghostview`, `man`, and `sed`) had sufficiently large false-positive rates at the red threshold with a half-life of 1 that we believed that these applications had temporarily unstable profiles and hence would not be used during our true-positive tests.

3.4.3 True-positive Test Results

Using the results from our false-positive test, as shown in Table 3.6, we excluded the applications `ccom`, `cp`, `ghostviev`, `man`, and `sed` from the true-positive tests. These applications exhibited an unusually high false-positive rate, and were deemed unusable for the true-positive tests (unpredictable results).

Tables 3.7 and 3.8 summarize the results of the two true-positive tests. Each subject contains information regarding the number of records in its anomalous data-set, and the position of the four (or one) anomalous record(s) within the set. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection. Shorter half-lives were generally more successful than longer half-lives in detecting masqueraders, primarily because there were so few masquerader audit records. (If there had been more masquerader audit records, the longer half-lives would have improved success in detecting the masquerading episodes.) Because the results with smaller short-term profile half-lives were favorable, we decided to concentrate on shorter half-lives in the next experiment.

A half-life of 1 seemed to show some residual effects after the masquerader data disappeared. This effect occurred because with a half-life of 1 it takes about four audit records for a given audit record to be erased from the short-term profile. To allow the short-term profile to contain a single audit record, we decided to set the short-term half-life to some number smaller than 1 (say 0.1). This causes the previous audit record to contribute $\frac{1}{1024}$ to the short-term profile, making the short-term profile essentially equivalent to the current audit record.

3.4.4 Conclusions

Our decision to use smaller short-term profile half-lives in our next experiment also convinced us to drop the intensity and interarrival measures. These measures are less meaningful with smaller short-term profile half-lives and they can be affected by how the masquerader data was injected, and thus give spurious detections. For the Sun version of NIDES, we were concerned about bursts of thousands of audit records that could indicate an intrusion attempt. By contrast, for the Safeguard task, we are actually trying to identify individual audit records that are masqueraders. If there were thousands (or even hundreds or tens) of masquerading audit records, we do not want to identify them through increased volume; we would want to identify a large percentage of them individually as masqueraders.

3.4.5 Concept Experiment Figures and Tables

The following pages contain figures and tables showing the results of the concept experiments.

		Half-life=50		Half-life=20		Half-life=10		Half-life=1	
Subject	# of Records	Yellow	Red	Yellow	Red	Yellow	Red	Yellow	Red
as	1539	2.1	2.3	1.9	2.2	2.0	2.5	1.4	2.0
cat	1058	1.9	5.6	1.9	4.9	1.7	4.2	1.8	3.1
ccom	736	1.8	2.1	1.9	2.5	2.1	2.8	1.5	2.0
compile	838	3.2	4.7	3.2	4.6	3.1	4.1	3.0	3.7
cp	273	1.6	2.2	2.0	2.0	1.9	2.0	2.4	2.6
cpp	2470	7.1	8.3	5.0	6.2	3.7	5.4	2.2	3.4
csh	709	2.3	3.9	2.8	4.0	2.9	3.8	2.4	4.2
diff	596	2.2	2.7	2.6	3.6	2.7	3.4	1.9	3.1
discuss	1040	2.0	4.8	1.9	4.4	2.1	4.2	2.1	3.2
emacs	6227	4.2	5.9	3.8	6.2	3.3	5.5	2.5	3.5
finger	537	4.8	4.8	3.7	4.0	3.5	3.5	2.4	2.8
fmt	1522	2.9	5.7	3.0	5.4	2.9	4.3	2.1	3.3
gawk	530	2.7	3.7	2.7	3.1	2.3	2.8	2.0	2.8
getfullnm	269	1.1	1.6	1.1	1.6	1.0	1.6	1.3	2.0
ghostview	225	2.4	2.5	2.5	2.9	3.3	3.5	2.1	3.2
grep	3474	3.6	7.1	3.9	5.3	3.0	4.7	2.6	3.9
latex	758	2.9	3.9	2.7	3.6	2.5	3.0	1.7	2.5
less	4709	4.5	5.5	3.7	5.1	3.4	4.4	2.2	3.1
ls	7368	4.0	4.8	4.1	4.8	4.1	4.6	2.5	3.6
mail	527	2.1	2.5	2.2	2.6	2.5	3.1	2.2	3.9
make	1095	2.4	7.3	2.3	6.3	2.2	5.6	2.2	4.0
man	708	2.4	4.6	3.2	5.4	3.2	6.1	1.9	4.0
more	6497	2.6	3.1	2.6	3.1	2.4	2.8	1.7	2.5
mymoreproc	3406	2.2	3.7	2.5	4.0	2.4	3.9	1.7	2.4
pwd	1181	2.0	2.9	2.3	2.6	2.4	2.9	1.7	1.9
rm	2184	4.1	4.7	3.8	5.1	3.2	5.1	2.5	4.5
sed	1464	2.1	3.0	2.1	2.5	2.1	2.6	1.7	3.0
sort	702	2.7	3.2	2.5	3.8	2.6	4.0	1.9	3.2
stty	871	1.7	2.2	1.7	2.3	1.8	2.4	1.4	2.1
vi	4663	6.6	6.8	6.0	6.3	5.0	5.2	2.4	3.1

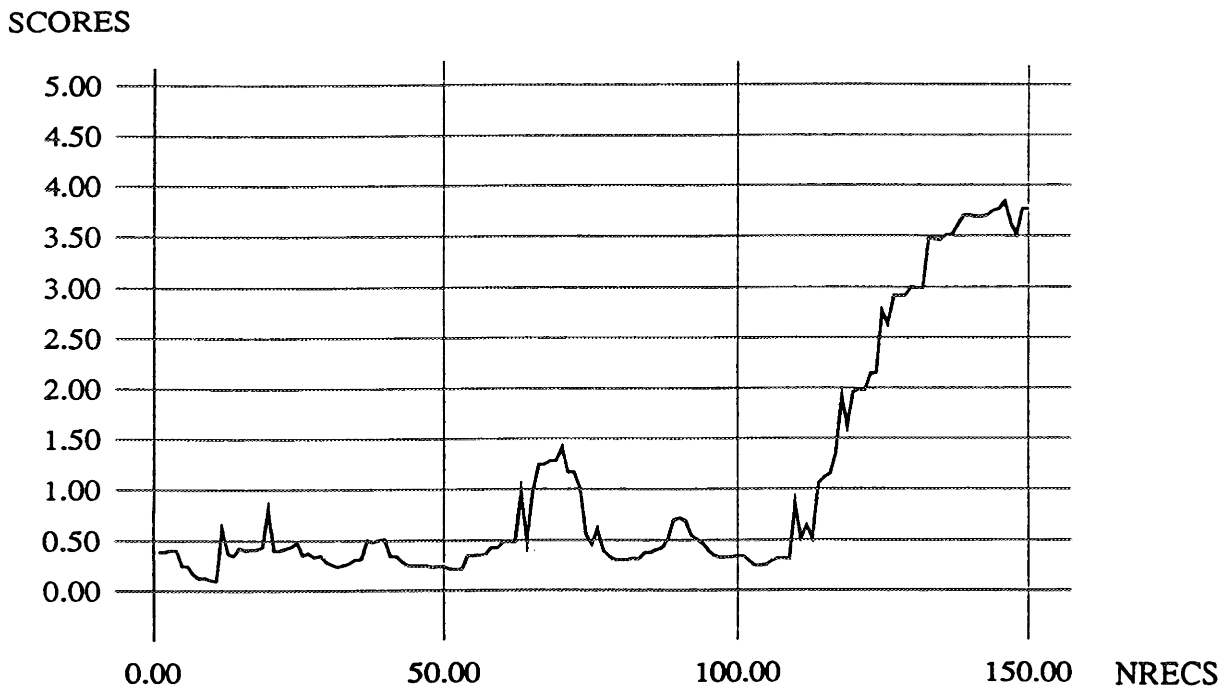
This table shows the score thresholds computed for each application during the profile-training process of the concept experiment. Each application subject is listed with the number of records used for profile-training (note that this number is actually one-third of the total number of records used for the training period, as we cycled through the training data three times). For each application subject, we have recorded the yellow and red threshold score values calculated for each of the short-term profile half-lives used for the experiment (50, 20, 10, and 1). The "yellow" column represents the score value at which records are flagged at a warning level, and the "red" column indicates the score value at which records are flagged at a critical level.

Table 3.5: Score Thresholds for the Concept Experiment

		Half-life=50		Half-life=20		Half-life=10		Half-life=1	
Subject	# Records	(%) Yellow	(%) Red	(%) Yellow	(%) Red	(%) Yellow	(%) Red	(%) Yellow	(%) Red
as	149	0.0	0.0	8.5	0.0	8.1	0.0	1.3	0.7
cat	137	0.0	0.0	0.0	0.0	2.5	0.0	2.6	0.7
ccom	150	21.3	18.7	19.3	29.3	20.0	5.3	12.0	5.3
compile	172	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
cp	105	95.3	93.5	74.8	74.8	43.9	43.9	13.1	10.3
cpp	2470	0.0	0.0	0.0	0.0	0.0	0.0	1.9	0.0
csh	200	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
diff	94	6.4	3.2	6.4	0.0	6.4	1.2	10.6	4.3
discuss	288	12.2	0.0	3.8	0.0	0.8	0.0	0.0	0.0
emacs	1702	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.1
finger	82	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mt	297	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gawk	83	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
getfullnm	84	7.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ghostview	95	89.9	89.0	56.9	46.8	14.7	11.9	2.8	0.0
grep	2211	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0
latex	170	0.0	0.0	0.0	0.0	0.0	0.0	5.0	1.1
less	700	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ls	1652	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0
mail	86	25.8	9.6	4.1	2.1	0.0	0.0	0.0	0.0
make	156	0.0	0.0	0.0	0.0	0.0	0.0	4.4	0.0
man	230	52.8	7.3	10.3	3.0	9.9	0.9	6.4	0.0
more	1518	9.8	5.5	4.9	2.3	3.5	2.5	2.5	0.2
mymoreproc	495	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0
pwd	224	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rm	355	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0
sed	337	50.2	34.4	18.1	15.1	14.5	9.8	8.9	0.0
sort	189	0.0	0.0	3.2	0.0	2.1	0.0	1.6	0.0
stty	132	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
vi	789	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

This table contains false-positive rates (percentages) for the concept experiment. The total number of audit records used for each application subject in the false-positive tests are listed in the second column. There are two columns of data for each half-life used (50, 20, 10, and 1). The columns labeled "yellow" indicate the false-positive rates at the warning threshold level (i.e., the percentage of records flagged at this level) based on the score thresholds listed Table 3.5, while the columns labeled "red" show the false-positive rates at the critical level.

Table 3.6: False-positive Rates (percent) for the Concept Experiment



The graph shown here represents a plot of the scores from the false-positive test for the application `ccom`. The x-axis represents the audit record number (starting from 0), and the y-axis indicates the anomaly score value of each record for that subject. (Note the behavior shift from about record 110 onward.)

Figure 3.1: Plot of Application `ccom` Scores during Concept Experiment False-positive Testing

Subject	#Records	Location	HL=50	HL=20	HL=10	HL=1
as	19	12-15	****	****	****	**+*
cat	69	36-39	- - - -	- - - +	- +++	- ++*
compile	27	18-21	- - - -	- - - -	- - - -	- - - -
cpp	21	12-15	- - - -	- - - -	- - - -	- - - -
csh	76	45-48	**++	+++*	+++*	+++*
diff	30	5-8	- - - -	- - - -	- - - -	- - +*
discuss	121	70-73	++++	++++	+++*	****
emacs	621	291-294	- - - -	- - - -	- - - -	- ++ -
finger	41	22-25	- - - -	- - - -	- - - -	- - - -
fmt	120	41-44	- - - +	- - - +	- - - +	- - +*
gawk	18	11-14	- - - -	- - - -	- - - -	+- ++
getfullnm	31	21-24	****	****	****	****
grep	2124	2058,2096,2102,2103	- - - -	- - - -	- - - -	- - - -
latex	69	4-7	- - - -	- - - -	- - - -	****
less	238	137-140	- - - -	- - - -	- - - -	- - - +
ls	637	350,358,359,367	- - - -	- - - -	- - - -	- - - -
mail	45	13-16	****	****	****	+*+*
make	680	24-27	- - - -	- - - -	- - - -	- - - -
more	597	267-270	- - - -	- - - -	- - - -	+++*
mymoreproc	166	92-95	- - - +	- - +*	- - +*	++**
pwd	88	54-56,58	+++*	++**	+***	****
rm	155	92-94,96	- - - -	- - - -	- - - -	- - - +
sort	79	45-48	- - - -	- - - -	- - - -	- - - +
stty	58	19-22	- - +*	- - +*	- - **	- ++*
vi	362	134-137	- - - -	- - - -	- - - -	- - - -

This table shows the results of the first of two true-positive tests performed during the concept experiment. The second column indicates the number of records used during the true-positive testing; the next column shows the positions of the masquerader data in the application's audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (50, 20, 10, and 1). Each half-life column contains four characters indicating the results of each of the four masquerading records data going through the application's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

Table 3.7: True-positive Results for the Concept Experiment (Masquerading Application #l)

Subject	#Records	Location	HL=50	HL=20	HL=10	HL=1
as	134	1	-	+	+	*
cat	70	9	+	+	+	*
compile	149	1	-	-	-	-
cpp	139	1	-	-	-	-
csh	118	8	+	+	+	+
diff	68	5	-	-	-	+
discuss	159	11	+	+	+	+
emacs	1029	80	-	-	-	-
finger	42	3	-	-	-	*
fmt	147	2	-	+	+	*
gawk	67	6	-	-	-	*
getfullnm	46	1	*	*	*	*
grep	98	3	-	-	-	+
latex	69	4	-	-	-	*
less	382	20	-	-	-	*
ls	961	56	-	-	-	+
mail	46	2	-	-	-	+
make	85	2	-	-	-	-
more	926	17	-	-	-	+
mymoreproc	279	10	*	*	*	*
pwd	135	1	+	*	*	*
rm	206	9	-	-	-	-
sort	106	8	*	+	+	*
stty	82	16	*	*	*	*
vi	337	19	-	-	-	*

This table shows the results of the second of two true-positive tests performed during the concept experiment. The second column indicates the number of records used during the true-positive testing; the next column shows the position of the masquerader data in the application's audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (50, 20, 10, and 1). Each half-life column contains one character indicating the results of the masquerading record going through the application's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

Table 3.8: True-positive Results for the Concept Experiment (Masquerading Application #2)

3.5 Verification Experiment

3.5.1 Configuration

Based on the observations we noted during the concept experiment, we deactivated the following twelve measures (half of our original 24):

- Network-related measures (RNET, RNETHOST, RNETTYPE, LNET, LNETHOST, LNETTYPE) were dropped because there was no variance in these measures across all applications.
- User-related measures (UID, UIDB) were also dropped as they were never observed for any of the applications.
- Intensity measures (INT60, INT600, INT3600), and the interarrival measure (INTARR) were dropped because they would be sensitive to where we placed the masquerading records.

This configuration left us with the 12 measures listed in Table 3.3.

We also changed the short-term profile half-lives to even smaller numbers of audit records. Because of the nature of the masquerader data (small number of records in a short period of time, or a single bad record), we decided to conduct further experiments using half-lives of 2, 1, and 0.1 (for the latter, the short-term profile consists of essentially a single audit record).

Using these configurations, profiles for the 30 applications were built. Table 3.9 shows the score thresholds for the applications profiled.

3.5.2 False-positive Test Results

Table 3.10 shows the false-positive rates for the applications profiled during the verification experiment. As before, several applications had rather high false-positive rates, that were indicative of behavioral shifts over time. We attributed the higher false-positive rates to the removal of measures that were generally unobserved or irrelevant, and which served in the concept experiment to dilute the overall score and reduce sensitivity. Their removal enabled NIDES to focus on a smaller, more meaningful set of measures.

Occasionally, as with `cpp`, the resulting greater sensitivity, relative to the concept experiment, amplified the unusual stretches of audit data and resulted in higher false-positive rates at the yellow threshold. In contrast with the activity shown in Figure 3.2 for the concept experiment, Figure 3.3 illustrates this amplification of the unusual activity in the final 20% to 30% of the data.

3.5.3 True-positive Test Results

Using the results from our false-positive tests as shown in Table 3.10, we excluded the applications `ccom`, `cpp`, `diff`, `latex`, and `sed` from the true-positive tests.

Tables 3.11 and 3.12 show the true-positive results for the applications tested during the verification experiment. These tables show comparable results as compared to the concept testing experiment. For example, comparing Tables 3.7 and 3.11 with a short-term profile half-life of 1 audit record, the number of anomalies at the red threshold has increased only slightly from 29 to 32 and the number of anomalies at the yellow threshold has increased only slightly from 24 to 27. For the second resource-intensive application the detection results appear slightly worse in the verification experiment.

The shortest half-life (i.e., 0.1 - where the short-term profile consists of a single audit record) generally did the best job of detecting masqueraders. For example, the total number of anomalies across all five masquerading audit records at the red threshold value increases from 41 to 73 as the short-term profile half-life decreases from 2 to 0.1. We believe that this is a consequence of the small number of masqueraders and their substantial differences from the host applications, as well as the greater inherent stability of the thresholds on which detection is based.

3.5.4 Conclusions

Some measures (most notably SIGNAL) did not reach a minimum number of observations to qualify as a trained measure in the profiles (a minimum effective number of 100 records), and hence did not contribute to the overall scoring, because a value of zero was not counted as an observation. We decided to change the statistical component to accept zero as an observed value to allow this measure (and possibly others) to contribute to the score in future experiments.

Rare Probability Threshold. The scoring of the short-term profile is based on a chi-square-like squared difference of expected and observed data counts. This statistic tends to be unstable in some applications when expected counts are unacceptably small. For this reason, Sun-C2 NIDES used a rare probability mechanism whereby at profile update time rare categories were combined into a special category and the sum of their probabilities was used to score an observation of any of these categories. The upper threshold for combining category probabilities, known as *MAXSUM-RAREPROB,* was moderately high (10% for Sun-C2 NIDES). For the present study with smaller effective lengths for the short-term profile, it was not possible to prevent expected counts from being small. Moreover, the high rare probability threshold caused some counter-intuitive scoring (such as never-seen categories possibly appearing more normal than categories that barely fail to make the "rare" combination). With a *MAXSUMRAREPROB* equal to 10%, it was frequently the case that an audit record that had values in never- before-seen categories contributed less toward the detection statistics (i.e., the Q's and T^2) than audit records that had measure values in seen-before categories.

As a result of our findings, the *MAXSUMRAREPROB* value was changed from 10% to 1% for the refinement experiment. This new value of *MAXSUMRAREPROB* prevents the counter-intuitive possibility of bins with small but real probabilities being scored as more unusual than never-seen-before bins.

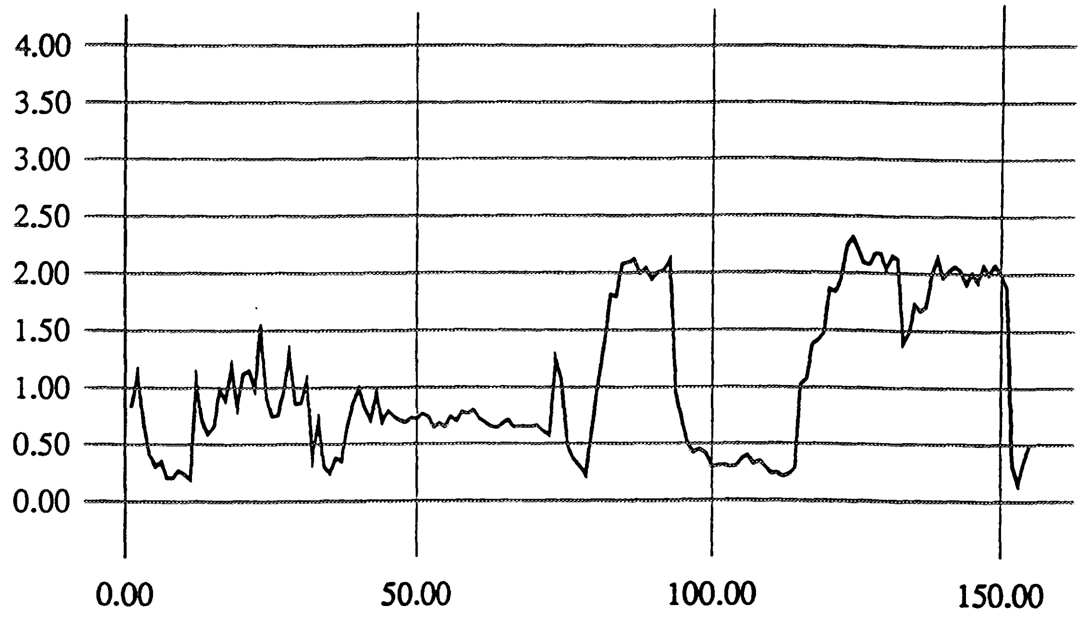
3.5.5 Verification Experiment Figures and Tables

The following pages contain figures and tables showing the results of the verification experiments.

		Half-life=2		Half-life=1		Half-life=0.1	
Subject	# of Records	Yellow	Red	Yellow	Red	Yellow	Red
as	1539	2.6	3.1	2.0	3.0	1.4	2.8
cat	1058	2.4	5.2	2.6	4.9	2.2	2.8
ccom	736	2.9	3.8	2.3	3.2	1.8	2.8
compile	838	4.6	6.1	4.6	5.8	2.6	3.4
cp	273	3.6	4.4	3.7	4.2	2.2	3.4
cpp	2470	3.0	5.0	2.6	3.7	2.1	3.1
csh	709	3.9	6.9	3.9	6.7	2.9	3.6
diff	596	3.0	4.9	2.9	4.6	2.4	3.5
discuss	1040	2.8	5.0	3.0	5.2	2.8	3.7
emacs	6227	3.5	4.7	3.2	5.0	2.4	3.2
finger	537	3.9	5.5	2.8	4.0	2.3	3.4
fmt	1522	3.0	4.6	2.3	4.8	1.7	2.8
gawk	530	2.9	4.9	2.9	4.5	2.7	4.5
getfullnm	269	1.6	2.3	1.5	3.1	1.7	2.0
ghostview	225	3.1	4.3	3.2	3.3	2.4	2.5
grep	3474	3.5	6.2	3.5	5.4	2.9	4.0
latex	758	3.3	4.3	2.4	4.1	1.8	3.9
less	4709	3.5	4.7	3.1	4.7	1.9	2.4
ls	7368	4.8	6.3	4.1	5.9	2.2	3.8
mail	527	3.9	6.2	3.1	6.5	2.1	5.1
make	1095	2.9	4.8	3.0	5.2	2.9	5.4
man	708	2.8	5.5	2.4	5.2	1.5	1.8
more	6497	2.5	4.3	2.5	4.0	2.0	2.6
mymoreproc	3406	2.1	3.1	1.8	2.8	1.5	2.7
pwd	1181	2.4	3.3	2.5	2.9	2.0	2.5
rm	2184	4.4	7.5	3.8	7.2	2.4	3.3
sed	1464	2.6	5.0	2.4	4.9	2.2	3.3
sort	702	2.7	6.0	2.7	5.1	2.4	3.0
stty	871	1.9	2.2	1.9	2.2	1.7	2.4
vi	4663	2.6	4.6	2.5	4.2	1.9	2.8

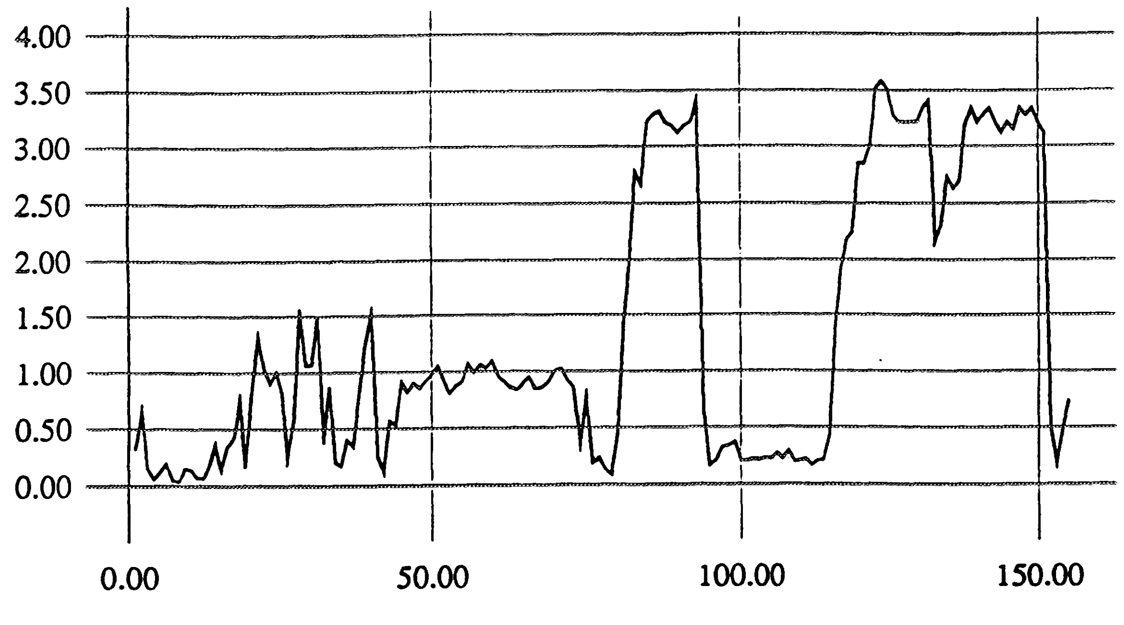
This table shows the score thresholds computed for each application during the profile-training process of the verification experiment. Each application subject is listed with the number of records used for training (note that this number is actually one-third of the total number of records used for the training period, as we cycled through the training data three times). For each application subject, we have recorded the yellow and red threshold score values calculated for each of the short-term profile half-lives used for the experiment (2, 1, and 0.1). The "yellow" column represents the score value at which records are flagged at a warning level, and the "red" column indicates the score value at which records are flagged at a critical level.

Table 3.9: Score Thresholds for the Verification Experiment



The graph shown here represents a plot of the scores from the false-positive test for the application cpp during the concept experiment. The x-axis represents the audit record number (starting from 0), and the y-axis indicates the anomaly score value of each record for that subject.

Figure 3.2: Plot of Application cpp Scores during Concept Experiment False-positive Testing



The graph shown here represents a plot of the scores from the false-positive test for the application cpp during the verification experiment. The x-axis represents the audit record number (starting from 0), and the y-axis indicates the anomaly score value of each record for that subject. Note the amplification of the abnormal data stretches with the reduced measure set used during the verification experiment vs. the concept experiment.

Figure 3.3: Plot of Application cpp Scores during Verification Experiment False-positive Testing

Subject	#Records	Location	HL=2	HL=1	HL=0.1
as	19	12-15	****	**+*	*+++
cat	69	36-39	+++*	-++*	-***
compile	27	18-21	- - - -	- - - -	- -**
cp	21	18-20,22	- - -*	- - -*	-+**
csh	76	45-48	- -+*	- - **	*+**
discuss	121	70-73	****	****	****
emacs	621	291-294	- -+-	-++-	*+++
finger	41	22-25	- - -+	- - -+	- - - -
fmt	120	41-44	- - -*	+***	+***
gawk	18	11-14	++++	++++	+++-
getfullnm	31	21-24	****	****	****
ghostview	61	35-38	-***	-***	****
grep	2124	2058,2096,2102,2103	- - - -	- - - -	- - - -
less	238	137-140	- - -+	+-++	****
ls	637	350,358,359,367	- - - -	- - - -	- - -+
mail	45	13-16	++++	++++	++++
make	680	24-27	- - - -	- - -+	+- -+
man	132	57-60	- -++	-+++	-+**
more	597	267-270	+++*	+++*	****
mymoreproc	166	92-95	-***	+***	****
pwd	88	54-56,58	****	****	****
rm	155	92-94,96	- - - -	- - - -	- -**
sort	79	45-48	- - -*	- - -+	- - - -
stty	58	19-22	-***	-***	-***
vi	362	134-137	- - - -	+- - -	*- - -

This table shows the results of the first of two true-positive tests performed during the verification experiment. The second column indicates the number of records used during the true-positive testing; the next column shows the positions of the masquerader data in the application's audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (2, 1, and 0.1). Each half-life column contains four characters indicating the results of each of the four masquerading records data going through the application's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

Table 3.11: True-positive Results from Verification Experiment (Masquerading Application #l)

Subject	#Records	Location	HL=2	HL=1	HL=0.1
as	134	1	*	*	*
cat	70	9	*	*	*
compile	149	1	-	-	*
cp	85	6	*	*	*
csh	118	8	+	+	*
discuss	159	11	+	+	*
emacs	1029	80	-	-	+
finger	42	3	*	*	*
fmt	147	2	*	*	*
gawk	67	6	*	*	*
getfullnm	46	1	*	*	*
ghostview	39	3	-	-	*
grep	98	3	+	+	+
less	382	20	+	+	*
ls	961	56	-	-	*
mail	46	2	+	+	*
make	85	2	-	-	+
man	103	8	+	+	*
more	926	17	+	+	*
mymoreproc	279	10	*	*	*
pwd	135	1	*	*	*
rm	206	9	-	-	*
sort	106	8	+	+	*
stty	82	16	*	*	*
vi	337	19	+	+	*

This table shows the results of the second of two true-positive tests performed during the verification experiment. For each subject we list the number of records used during the true-positive testing; the next column shows the position of the masquerader data in the application's audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (2, 1, and 0.1). Each half-life column contains one character indicating the result of the masquerading record going through the application's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

Table 3.12: True-positive Results for the Verification Experiment (Masquerading Application #2)

3.6 Refinement Experiment

3.6.1 Configuration

Results from the verification experiment indicated that we needed to make two additional adjustments before we were convinced that the statistical component was satisfactorily adapted to the Safeguard audit data. First, because we noticed that the probability for the rare categories seemed rather high, and thus masked the contribution of categories with small-but-not-rare probabilities during the verification experiment, we reduced the *MAXSUMRAREPROB* value from 10% to 1%.

Because some measures did not reach a minimum number of observations to qualify as a trained measure, and hence did not contribute to the overall scoring during our verification experiment, we modified the code such that an observed value of zero (i.e., a non-event) was counted into the profile.

We used the same 12 measures as in the verification experiment, and the same half-life set: 2, 1, and 0.1 (single). Using these configurations, we built profiles for the 30 applications. Table 3.13 shows the score thresholds for the applications profiled.

3.6.2 False-positive Test Results

Table 3.14 shows the false-positive rates for the applications profiled. We noted considerable improvements in the false-positive rates. The progressive refinements achieved our objective with respect to false-positive rates, especially for shorter profile half-lives. Most subjects with false detection rates significantly above nominal levels have a relatively small amount of data, such as `diff.` The one exception is `cpp,` which, as we discussed in Subsection 3.5.2, appears to have an unacceptably unstable profile. Otherwise, false-positive rates were well within acceptable limits, especially as we proceeded to the smallest short-term profile half-life values.

3.6.3 True-positive Tests Results

Applications `ccom, diff, cpp,` and `sed` were eliminated from the true-positive testing because of high false-positive rates as indicated in Table 3.14.

Tables 3.15 and 3.16 show the true-positive results for the applications tested. The results for the different half-lives are quite comparable. For between 18 and 19 of the 26 applications, NIDES was able to identify one or more of the masquerading audit records using the red threshold. In three to five of the remaining applications there were two or more audit records that exceeded the yellow threshold. Only two applications (`vi` and `grep`) appeared to be quite tolerant of the masquerading applications. These results suggest that masquerades by these resource-intensive applications for any appreciable number of audit records would almost certainly be identified, except possibly when masquerading as `vi` or `grep.`

For the short-term profile half-life of 0.1 (i.e., the single audit record profile length) and the four masquerader records from the first application, NIDES in this final configuration declared a red alert on 37 of 104 opportunities and a yellow alert on an additional 38, detecting as unusual 72% of the masquerader data. In addition, this application generated at least one red alert when attempting to masquerade 19 of the 26 other applications, indicating a high probability of eventual detection with continued use. For only one host application, `vi,` was it the case that all four audit records from the unauthorized application escaped detection altogether. The results for the second resource-intensive application were even more impressive, with 25 red- and 1 yellow-level detections in 26 opportunities (again, with the short-term profile half-life of 0.1). For both masquerading applications, we observed 101 detections in 130 opportunities (approximately 77%). Failure to detect is due either to the fact that the masquerader data hit some of the same category bins that the host application hits or to the large variation of the short-term profile about historical behavior for the host application (this latter effect inflates the T^2 threshold and may consider records normal that overlap the historical profile minimally in so far as category bins).

As seen in Table 3.15, whether the masquerading application can go undetected depends on the particular application that is being "impersonated." For example, with a half-life of 0.1, all audit records of the resource-intensive application are flagged if that application masquerades as latex. In contrast, only the last audit record engenders a raised suspicion level if that application masquerades as `grep.` This situation is explored in more depth in Figure 3.4. Referring to the `latex` graph, we see that the four masquerading audit records receive very high S-values (in excess of 3.5) on 7, 8, 7, and 4 measures, respectively. Referring to the `grep` graph, we see substantially fewer large S-values. The first masquerading audit record possesses no large S-values, and therefore looks very much like a typical `grep` record. The next two masquerading audit records have only single large S-values and are not very different from typical `grep` records. The fourth masquerading audit record demonstrates four S-values in excess of 3.5 and therefore raises some suspicion. However, as seen from the relatively high values for the yellow and red thresholds for `grep` in Table 3.13, individual `grep` records show sufficient variability that although suspicious, the presence of four large S-values is not enough to declare an audit record anomalous using the red threshold criteria. The fourth masquerading audit record violates only the yellow, or warning, threshold.

Even greater insight can be gained by examining Figures 3.5 through 3.10, which show the long-term distribution (in gray) for each measure of the subject `grep.` In addition, the black bars represent the histogram of the four masquerading audit records. Examining the graph for the MEMUSE (memory use) measure, we see that only one of the four masquerading audit records "resides" in a bin that is sparsely populated with `grep` records, and as seen in Figure 3.4, that audit record receives an S-value of about 3.6. The other three masquerading audit records reside in bins that do not infrequently see `grep` records, and therefore these masquerading audit records receive much lower S-values for MEMUSE. We also see a number of measures with many bins with small amounts of probability in the `grep` profile. For example, the I/O measure possesses ten bins (bins 1 through 7 and 14 through 16) for which there are small amounts of probability. This indicates that `grep` records themselves are widely distributed across bins, and it is not too uncommon for `grep` records to appear in sparsely populated bins. (The total probability in these sparsely populated bins is approximately 8%) This is one of the reasons why we see a relatively large red threshold for `grep`. It is also the reason why the *S*-values for the three masquerading audit records that occur in sparsely-populated bins achieve only a moderately unusual value of approximately 2.5.

The first three masquerading audit records do look essentially like `grep` audit records, or at least display no more dissimilarity to `grep` audit records than occasionally seen in the record-to-record dissimilarity among `grep` records. The fourth masquerading audit record is somewhat suspicious and deserves to be flagged at the yellow (warning) level. Note that `grep` is probably not the type of application that will need export control.

3.6.4 Conclusions

Overall, we were satisfied with the results of this third experiment, and determined that this final configuration setting was sufficient to conduct further experiments (such as cross-profiling and subject group analysis). Table 3.17 shows the comparison in true-positive performance over the three experiments.

Figure 3.11 shows the average values of the *S* statistics for the 12 common measures used in the concept, verification, and refinement experiments. The height of the bars can be used to gauge the relative contribution of the measures to identifying the five masquerader audit records. The most important measures were OPENF, MEMCMB, I/O, and MEMUSE. The masquerading records looked very similar to the profiled applications (non-resource-intensive) with respect to TEXTSZ SIGNAL, PGIN, and HOUR. Results for UCPU and SCPU were mixed.

3.6.5 Refinement Experiment Figures and Tables

The following pages contain figures and tables showing the results of the refinement experiments.

		Half-life=2		Half-life=1		Half-life=0.1	
Subject	# of Records	Yellow	Red	Yellow	Red	Yellow	Red
as	1539	3.7	5.2	3.7	6.3	4.3	7.0
cat	1058	3.0	6.1	3.2	5.5	3.2	5.7
ccom	736	2.9	4.6	3.2	5.4	3.5	5.8
compile	838	6.3	7.6	6.0	7.2	4.7	5.5
cp	273	3.7	6.4	4.1	6.5	4.5	5.7
cpp	2470	4.1	6.5	4.1	6.5	4.1	6.5
csh	709	4.4	6.9	4.6	7.0	4.6	7.0
diff	596	4.0	5.9	4.3	6.9	4.3	7.7
discuss	1040	4.1	6.9	4.0	7.0	3.9	6.8
emacs	6227	4.5	8.1	4.5	8.1	4.1	8.0
finger	537	3.2	4.2	3.2	4.2	2.1	3.2
fmt	1522	3.2	4.8	2.7	4.9	1.8	4.2
gawk	530	3.5	5.0	3.3	6.3	2.6	6.8
getfullnm	269	1.5	2.3	1.5	2.3	1.5	2.0
ghostview	225	4.0	4.6	3.7	4.4	4.1	5.0
grep	3474	5.4	8.2	5.8	8.3	5.3	7.5
latex	758	2.9	4.5	2.8	4.3	2.6	4.3
less	4709	3.7	5.9	3.8	5.8	3.5	5.2
ls	7368	4.3	6.8	4.0	6.7	3.3	6.3
mail	527	3.5	7.1	2.9	7.1	3.0	7.5
make	1095	4.1	7.6	4.2	7.9	4.3	8.3
man	708	3.0	5.4	3.0	4.2	2.5	4.4
more	6497	3.7	5.8	3.6	5.6	3.6	5.6
mymoreproc	3406	2.2	3.3	2.1	3.2	1.9	3.0
pwd	1181	2.4	3.3	2.4	3.2	1.8	2.6
rm	2184	5.0	7.2	4.5	6.8	4.5	6.7
sed	1464	3.4	7.5	3.5	7.4	3.1	6.0
sort	702	4.4	8.5	4.1	8.5	2.8	8.3
stty	871	2.0	2.7	2.1	2.9	1.7	2.9
vi	4663	4.2	6.5	4.6	7.5	4.8	7.4

This table shows the score thresholds computed for each application during the profile-training process for the refinement experiment. Each application subject is listed with the number of records used for training (note that this number is actually one-third of the total number of records used for the training period, as we cycled through the training data three times). For each application subject, we have recorded the yellow and red threshold score values calculated for each of the short-term profile half-lives used for the experiment (2, 1, and 0.1). The "yellow"column represents the score value at which records are flagged at a warning level, and the "red" column indicates the score value at which records are flagged at a critical level.

Table 3.13: Score Thresholds for the Refinement Experiment

		Half-life=2		Half-life=1		Half-life=0.1	
Subject	# of Records	(%) Yellow	(%) Red	(%) Yellow	(%) Red	(%) Yellow	(%) Red
as	149	0.0	0.0	0.7	0.0	0.0	0.0
cat	137	3.2	1.3	3.2	1.3	3.9	1.9
ccom	150	12.0	0.0	7.3	1.3	1.3	1.3
compile	172	0.0	0.0	0.0	0.0	0.0	0.0
cp	105	0.0	0.0	0.0	0.0	0.0	0.0
cpp	2470	11.6	0.0	6.5	0.0	5.2	0.0
csh	200	0.5	0.0	0.5	0.0	0.5	0.5
diff	94	11.7	5.3	8.5	3.2	5.3	0.0
discuss	288	1.0	0.4	0.7	0.0	0.7	0.0
emacs	1702	2.3	0.2	1.5	0.3	2.0	0.3
finger	82	0.0	0.0	0.0	0.0	0.0	0.0
fmt	297	0.0	0.0	0.0	0.0	0.3	0.0
gawk	83	0.0	0.0	0.0	0.0	1.3	0.0
getfullnm	84	0.0	0.0	1.3	0.0	2.6	1.3
ghostview	95	0.9	0.9	3.7	1.8	0.9	0.0
grep	2211	0.1	0.0	0.1	0.0	0.1	0.0
latex	170	4.4	1.1	3.9	0.6	3.9	0.0
less	700	0.7	0.0	0.7	0.3	0.7	0.4
ls	1652	0.6	0.1	1.0	0.1	1.0	0.1
mail	86	1.0	0.0	1.0	0.0	0.0	0.0
make	156	4.4	0.0	4.4	0.0	2.2	0.0
man	230	0.4	0.0	0.0	0.0	0.9	0.0
more	1518	0.7	0.0	0.9	0.0	0.7	0.0
mymoreproc	495	0.6	0.0	1.0	0.0	0.8	0.0
pwd	224	0.4	0.0	0.4	0.0	0.4	0.4
rm	355	0.3	0.0	0.5	0.3	0.3	0.0
sed	337	10.7	0.0	3.9	0.0	0.3	0.0
sort	189	1.6	0.0	1.1	0.0	1.1	0.0
stty	132	1.4	0.0	0.0	0.0	0.0	0.0
vi	789	1.3	0.1	1.4	0.1	1.3	0.1

This table contains false-positive rates (percentages) for the refinement experiment. The total number of audit records used for each application subject in the false-positive tests are listed in the second column. Next are two columns of data for each half-life used (2, 1, and 0.1). The columns labeled "yellow" indicate the false-positive rates at the warning threshold level (i.e., the percentage of records flagged at this level) based on the score thresholds listed in Table 3.13, while the columns labeled "red" show the false-positive rates at the critical level.

Table 3.14: False-positive Rates (percent) for the Refinement Experiment

Subject	#Records	Location	HL=2	HL=1	HL=0.1
as	19	12-15	**+*	*+++	++-+
cat	69	36-39	-+**	--**	--**
compile	27	18-21	---+	---+	--+*
cp	21	18-20,22	--++	--++	---*
csh	76	45-48	+++*	+++*	+++*
discuss	121	70-73	****	**+*	**++
emacs	621	291-294	***+	***+	**++
finger	41	22-25	---*	---*	--+*
fmt	120	41-44	-***	-***	+***
gawk	18	11-14	*++*	+++*	++++
getfullnm	31	21-24	****	****	****
ghostview	61	35-38	****	****	**++
grep	2124	2058,2096,2102,2103	----	----	---+
latex	106	44-47	****	****	****
less	238	137-140	-++*	+++*	+++*
ls	637	350,358,359,367	---+	---+	---*
mail	45	13-16	+++*	+++*	++++
make	680	24-27	---+	++-+	+--+
man	132	57-60	-++*	--+*	--+*
more	597	267-270	--+*	--+*	-++*
mymoreproc	166	92-95	+***	+***	+***
pwd	88	54-56,58	****	****	****
rm	155	92-94,96	--++	--+*	--+*
sort	79	45-48	--++	--++	--++
stty	58	19-22	+***	+***	++**
vi	362	134-137	---+	---+	----

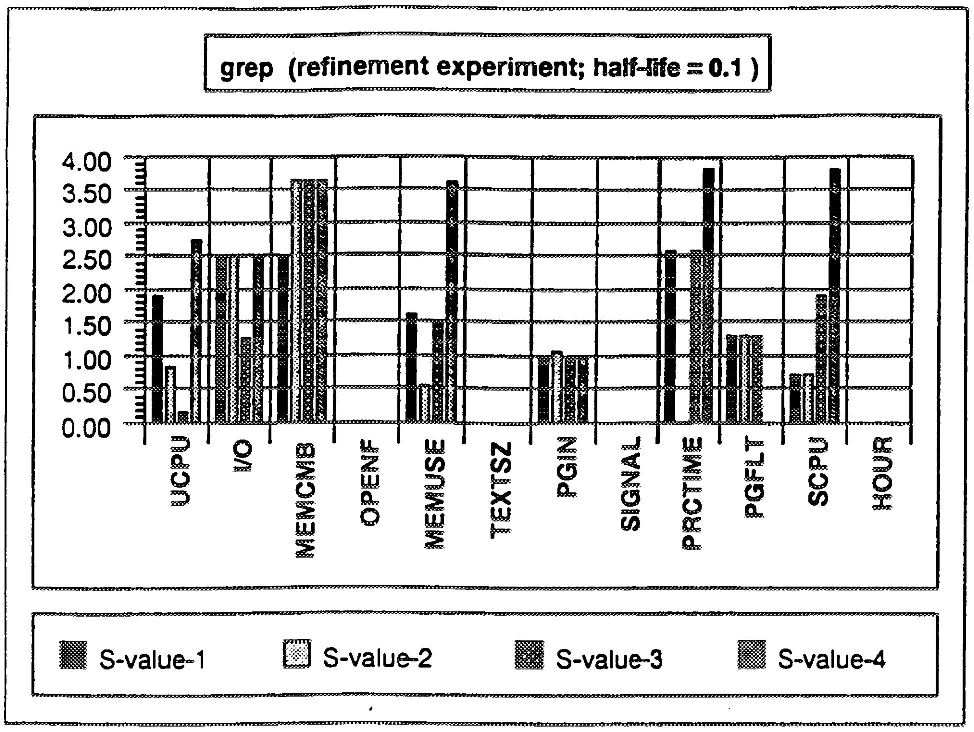
This table shows the results of the first of two true-positive tests performed during the refinement experiment. The second column indicates the number of records used during the true-positive testing. The next column shows the positions of the masquerader data in the application's audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (2, 1, and 0.1). Each half-life column contains four characters indicating the results of each of the four masquerading records data going through the application's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

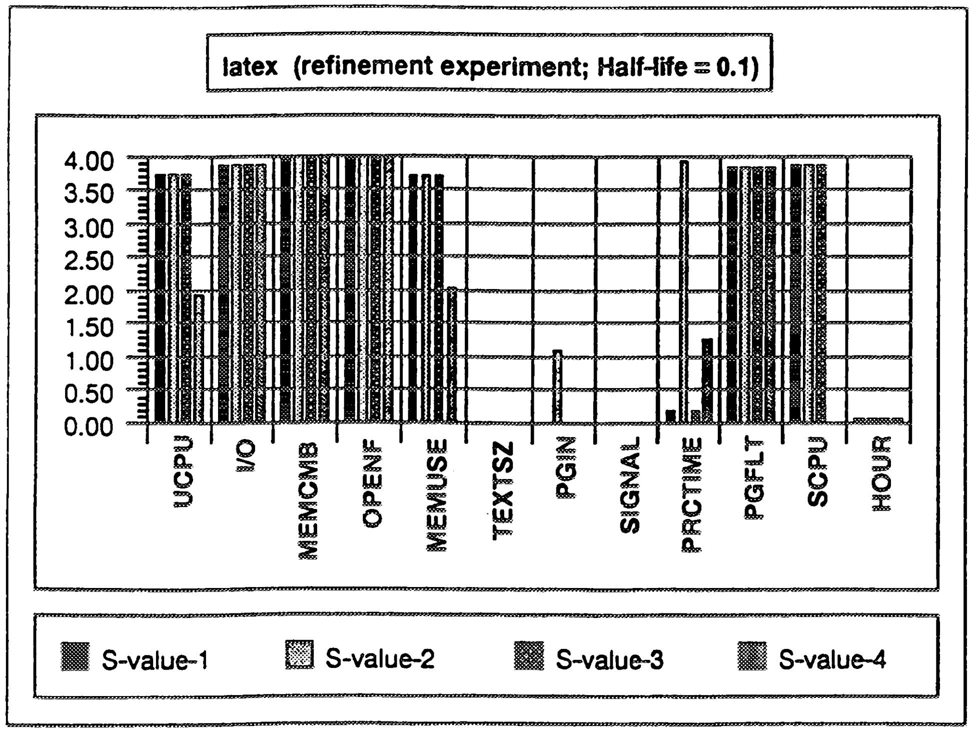
Table 3.15: True-positive Results from Refinement Experiment (Masquerading Application #1)

Subject	#Records	Location	HL=2	HL=1	HL=0.1
as	134	1	*	*	*
cat	70	9	*	*	*
compile	149	1	*	*	*
cp	85	6	*	*	*
csh	118	8	*	*	*
discuss	159	11	*	*	*
emacs	1029	80	+	+	+
finger	42	3	*	*	*
fmt	147	2	*	*	*
gawk	67	6	*	*	*
getfullnm	46	1	*	*	*
ghostview	39	3	*	*	*
grep	98	3	+	+	*
latex	69	4	*	*	*
less	382	20	*	*	*
ls	961	56	*	*	*
mail	46	2	*	*	*
make	85	2	+	*	*
man	103	8	*	*	*
more	926	17	*	*	*
mymoreproc	279	10	*	*	*
pwd	135	1	*	*	*
rm	206	9	*	*	*
sort	106	8	*	*	*
stty	82	16	*	*	*
vi	337	19	*	*	*

This table shows the results of the second of two true-positive tests performed during the refinement experiment. The second column indicates list the number of records used during the true-positive testing. The next column shows the positions of the masquerader data in the applications audit record stream. The remaining columns summarize the results of the test for each short-term profile half-life used (2, 1, and 0.1). Each half-life column contains one character indicating the results of the masquerading record going through the applications profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

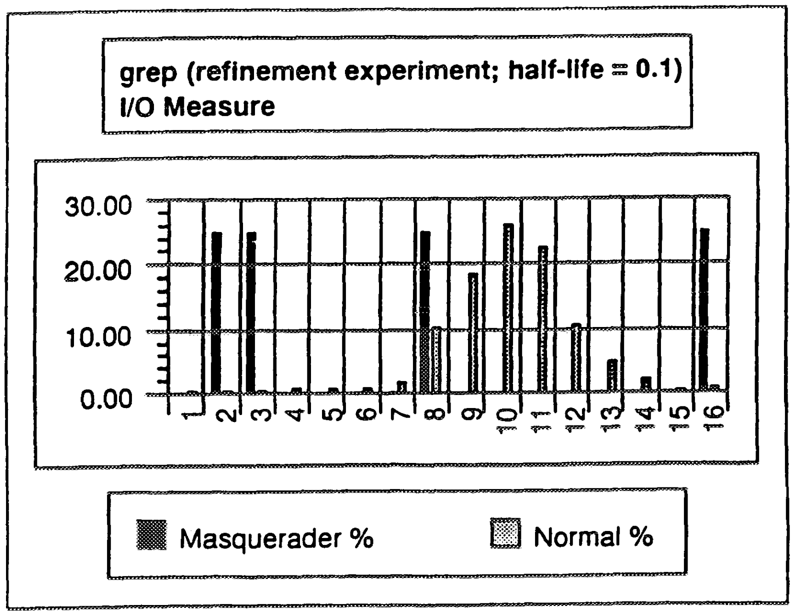
Table 3.16: True-positive Results from Refinement Experiment (Masquerading Application #2)

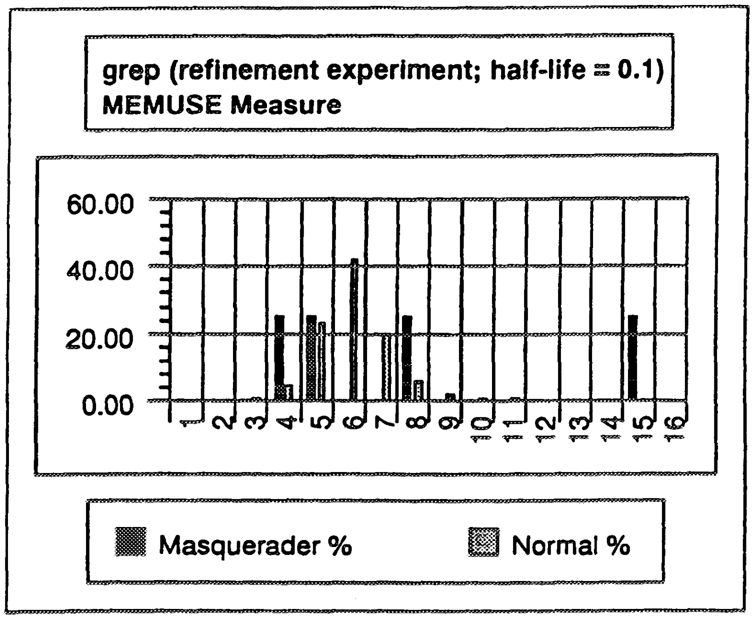




This figure shows the S scores by measure for the four records from the first resource-intensive application as processed through the profiles for `grep` and `latex`, respectively. Recall that the S score is obtained for each measure by a half-normal transformation of the Q distribution. High S values indicate that the audit record in question differs significantly from the underlying historical profile. Our half-normal inversion returns a maximum value of 4. The top figure shows moderately elevated S scores for some of the four records against the `grep` profile for the measures MEMCB, MEMUSE, PRCTIME, and SCPU. The lower figure shows that many of the measures have elevated S scores (including several at the theoretical maximum) for the four records scored against `latex`. This indicates that these records appear more unusual against the `latex` profile than against that for `grep`. The final step in the scoring mechanism compares this "unusualness" versus the "unusualness" of records against their own profiles (i.e., scores for `grep` against the `grep` profile).

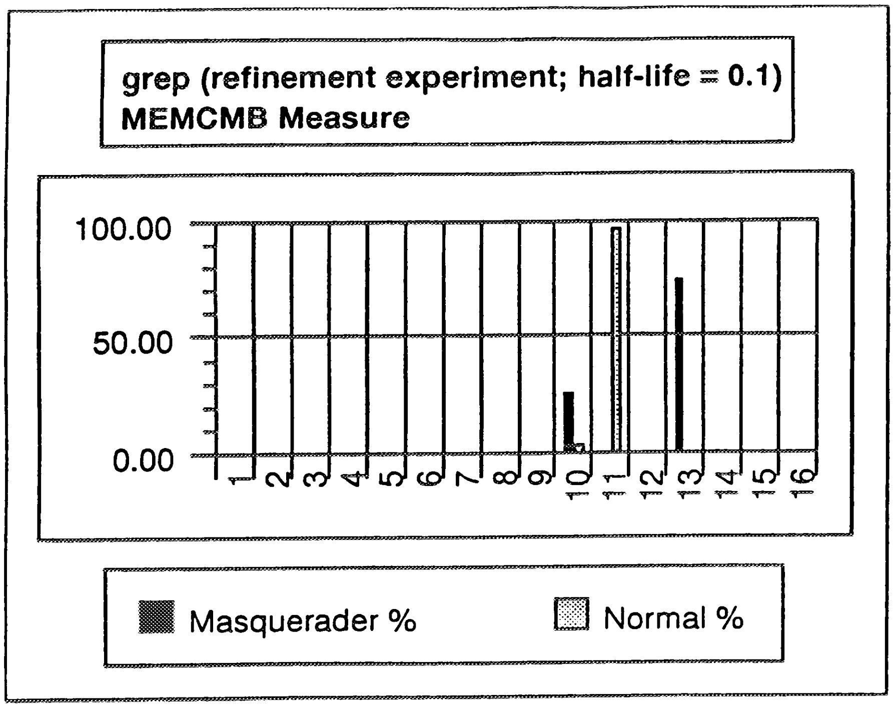
Figure 3.4: Measure S-values for `Grep` and `Latex`

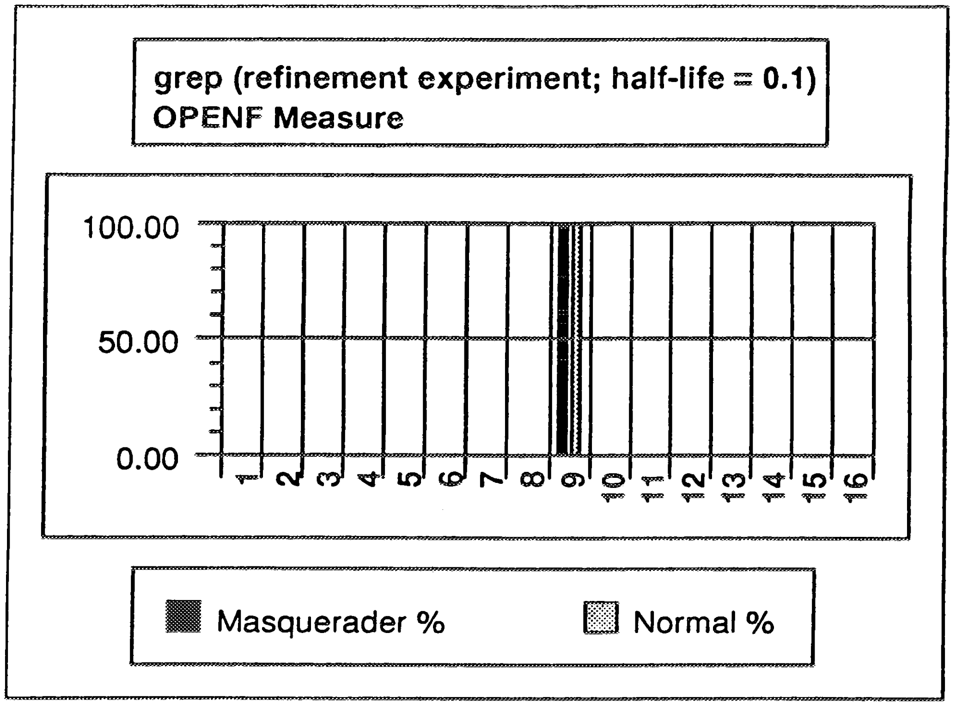




Historical grep profile data are shown in gray for the bin probabilities of the I/O and MEMUSE measures. The observed bin values for the four masquerader records are shown in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the I/O and MEMUSE measures will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records. For the I/O measure, we observe that grep's activity usually falls between bins 7 and 14, although there are a number of bins with small but non-zero probability. This indicates that it is not unusual for grep observations to fall into some bin with fairly small probability. Three of the four masquerader records fall into such bins. The fourth lands in bin 8, which has moderate historical probability. We thus expect three of the masquerader records to have somewhat high S scores for I/O, but scores by no means off the chart due to grep s frequency of landing in low-probability bins. The fourth masquerader audit record will appear relatively normal in S. This is what is observed in Figure 3.4, with three masquerader S scores at 2.5 and one at 1.3 for this measure against the grep profile. Thus it is seen that the S scores do not go to their maximum when unusual bins are observed if the underlying subject regularly hits unusual bins. The lower figure shows that for MEMUSE, three of the four masquerader records fall into bins with quite reasonable probability, so that the corresponding S scores would be on the low side. The record in bin 15 should receive a moderately high S. Again, this is confirmed in Figure 3.4.

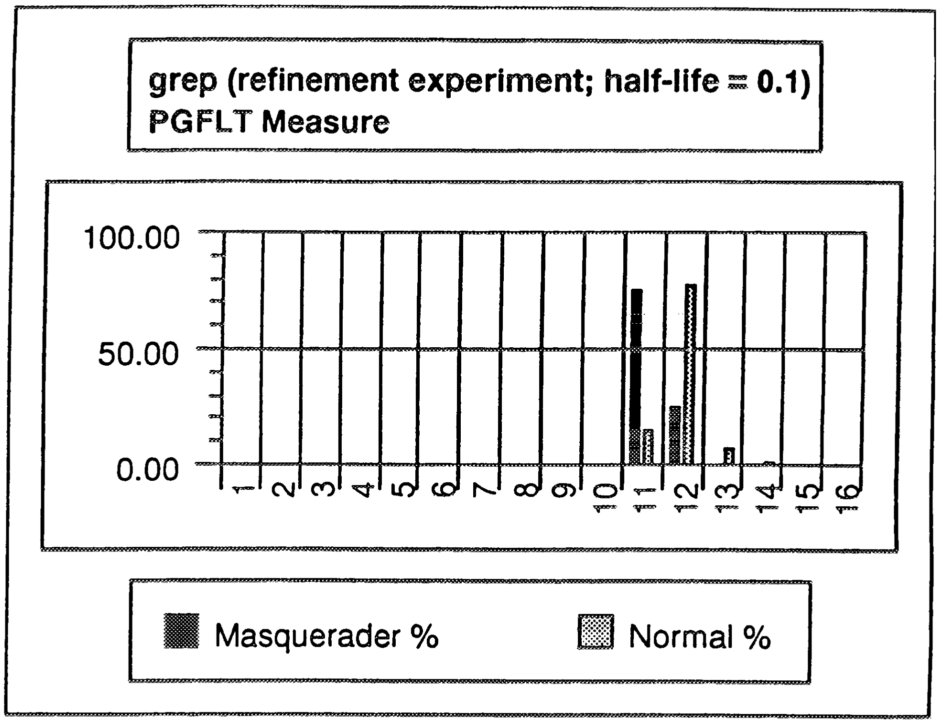
Figure 3.5: Long-term Profile Category Bins for Application Grep (Measures I/O and MEMUSE)

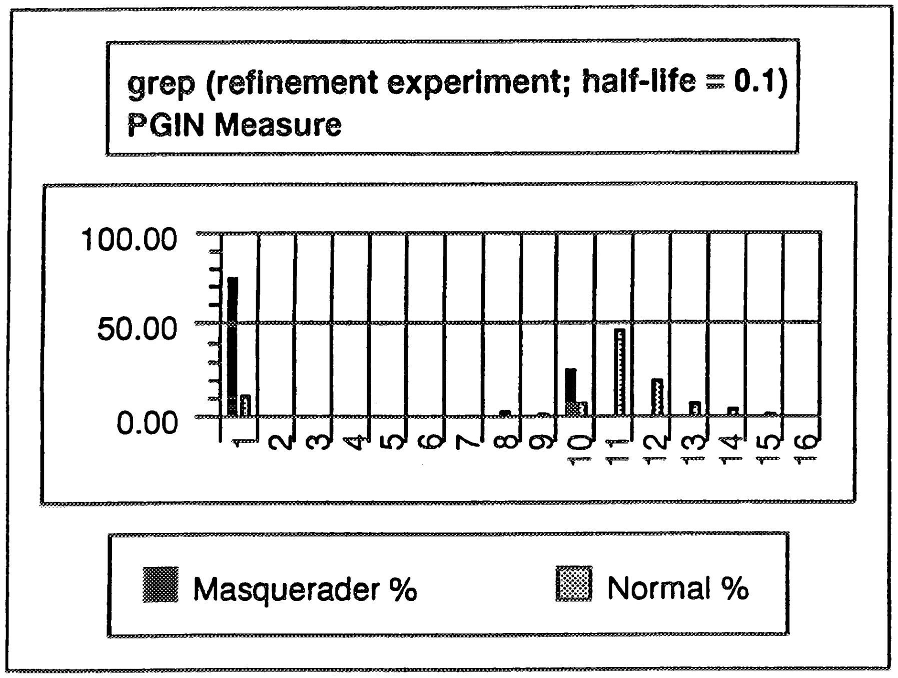




Historical grep profile data are shown in gray for the bin probabilities of the MEMCMB and OPENF measures. The observed bin values for the four masquerader records shown in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the MEMCMB and OPENF measures will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records.

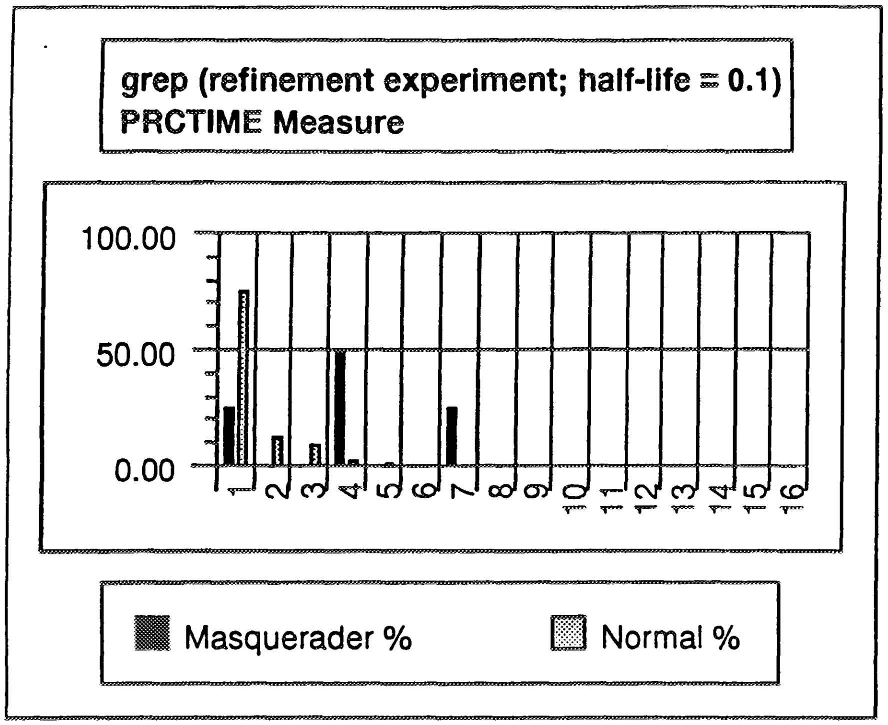
Figure 3.6: Long-term Profile Category Bins for Application Grep (Measures MEMCMB and OPENF)

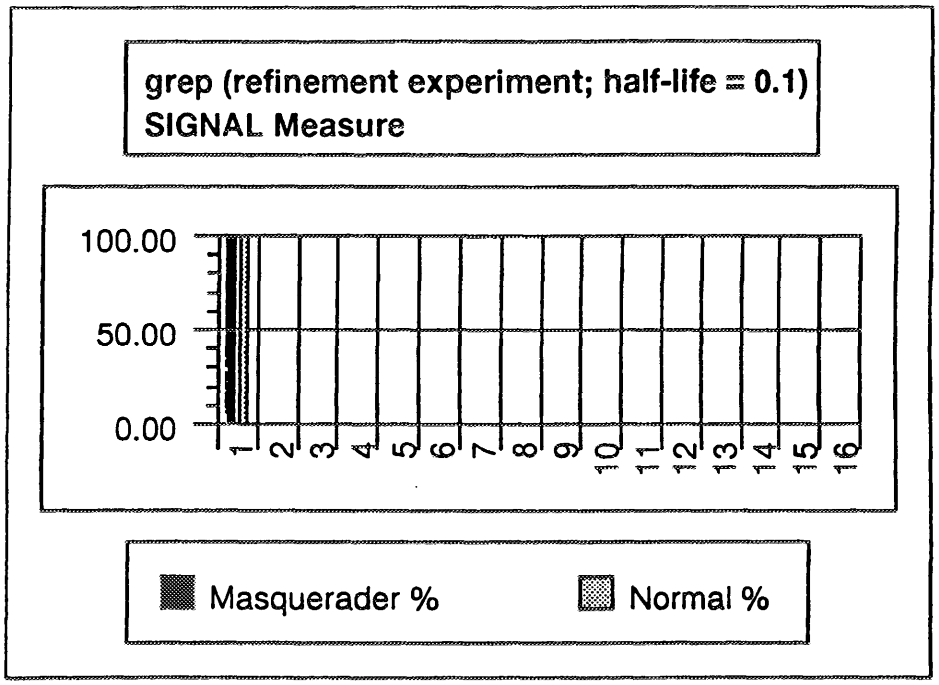




Historical grep profile data are shown in gray for the bin probabilities of the PGFLT and PGIN measures. The observed bin values for the four masquerader records are shown in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the PGFLT and PGIN measures will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records.

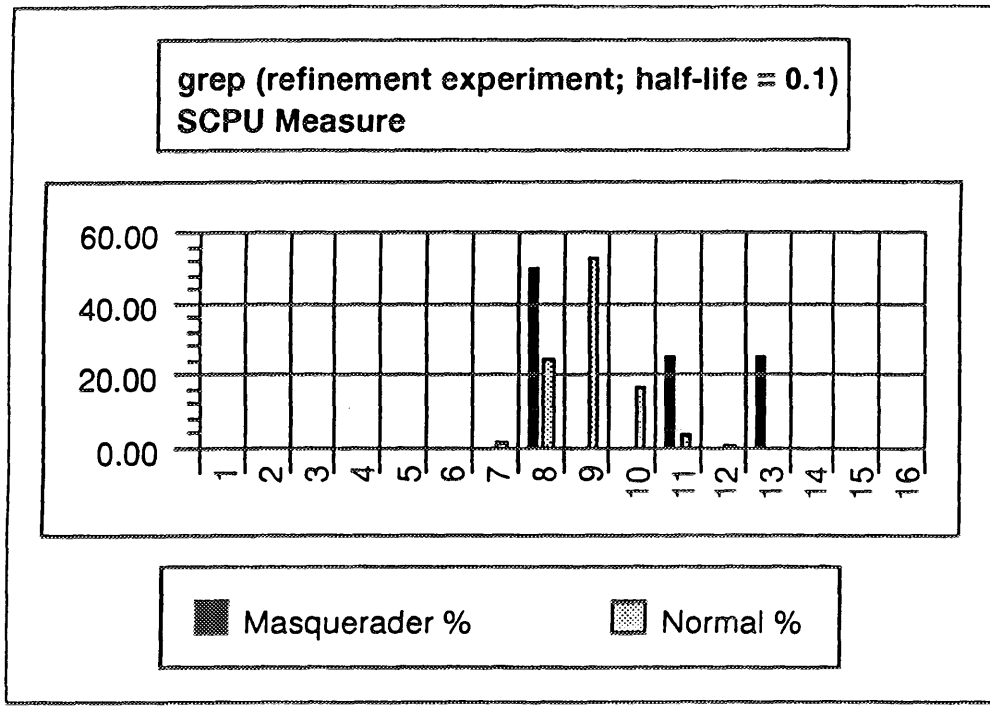
Figure 3.7: Long-term Profiles Category Bins for Application Grep (Measures PGFLT and PGIN)

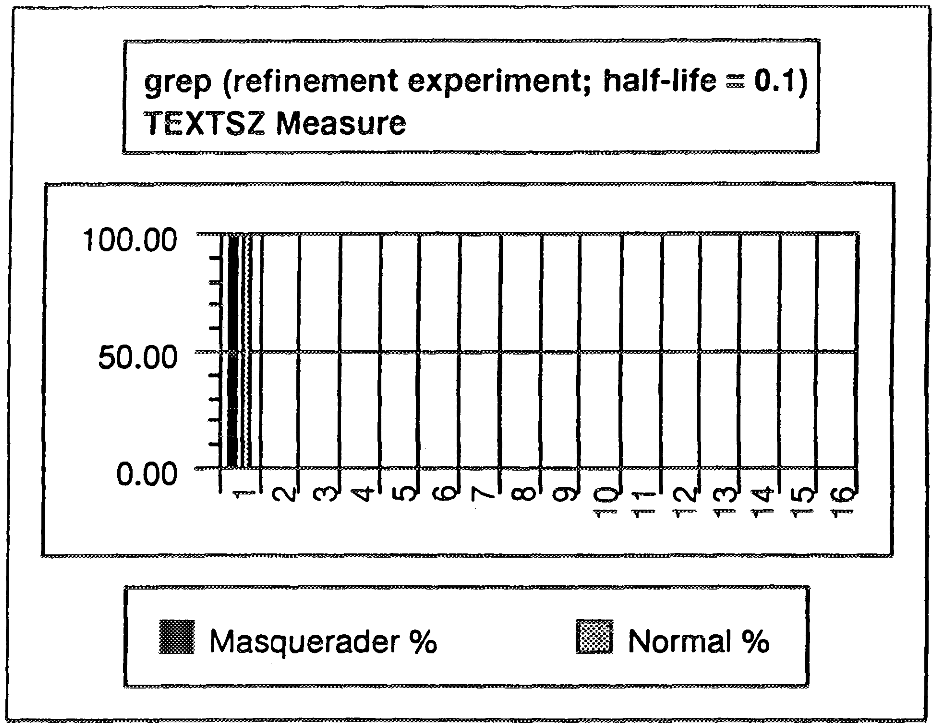




Historical grep profile data are shown in gray for the bin probabilities of the PRCTIME and SIGNAL measures. The observed bin values for the four masquerader records are shown in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the PRCTIME and SIGNAL measures will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records.

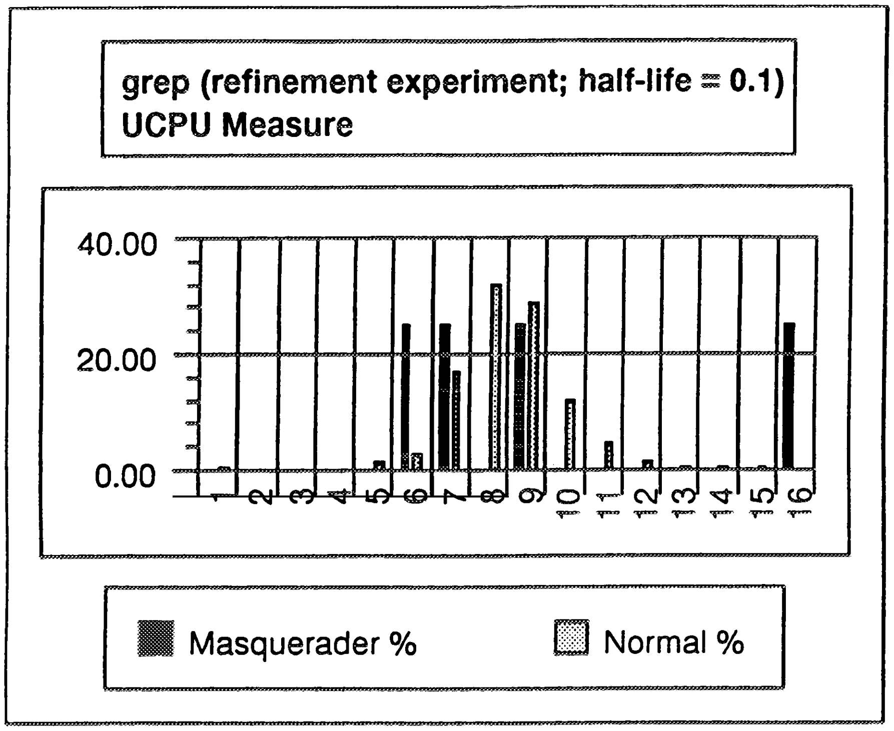
Figure 3.8: Long-term Profiles Category Bins for Application Grep (Measures PRCTIME and SIGNAL)





Historical grep profile data are shown in gray for the bin probabilities of the SCPU and TEXTSZ measures, with the observed bin values for the four masquerader records in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the SCPU and TEXTSZ measures will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records.

Figure 3.9: Long-term Profiles Category Bins for Application Grep (Measures SCPU and TEXTSZ)



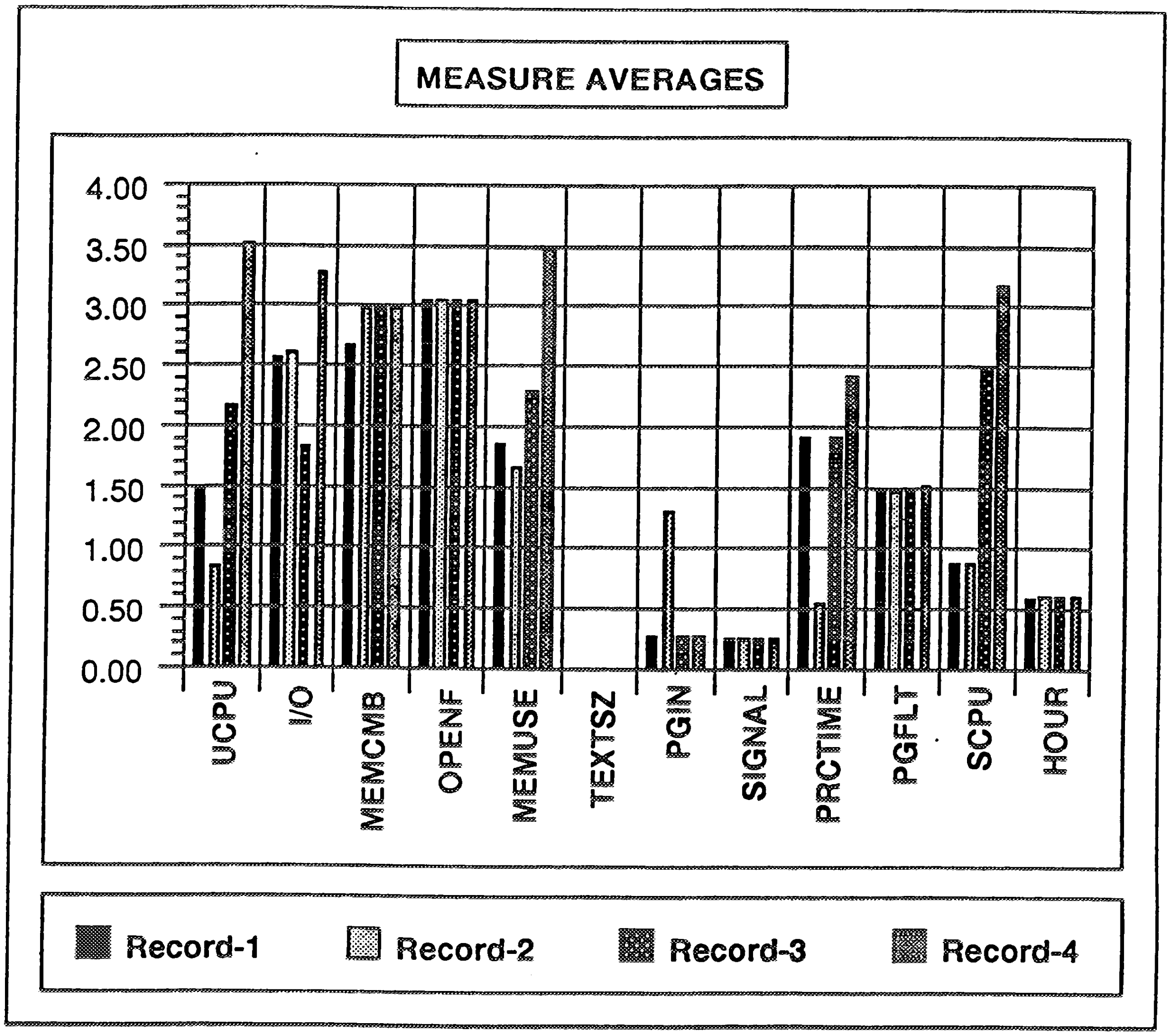
Historical grep profile data are shown in gray for the bin probabilities of the UCPU measure. The observed bin values for the four masquerader records are shown in black. The gray bars represent the historically observed probabilities for the application grep that the audit data's observed value for the UCPU measure will fall into a particular bin. The black bars show the bin locations by percentage of the four masquerader data records.

Figure 3.10: Long-term Profiles Category Bins for Application Grep (Measure UCPU)

	True-positive Results		
Subject	Concept	Verification	Refinement
	HL=1	HL=0.1	HL=0.1
as	**** *	*+++ *	++-+ *
cat	- - +* *	-*** *	- -** *
compile	- - - - -	- -** *	- -+* *
cp	- - - - -	-+** *	- - -* *
csh	+++* +	*+** *	+++* *
diff	- - +* +	n/a	n/a
discuss	**** +	**** *	**++ *
emacs	- ++ - -	*+++ +	**++ +
finger	- - -+ *	- - - - *	- - +* *
fmt	- - +* *	+*** *	+*** *
gawk	+-++ *	+++- *	++++ *
getfullnm	**** *	**** *	**** *
ghostview	n/a	**** *	**++ *
grep	- - - - +	- - - - +	- - - + *
latex	**** *	n/a	**** *
less	- - -+ *	**** *	+++* *
ls	- - - - +	- - -+ *	- - -* *
mail	n/a	++++ *	++++ *
make	- - - - -	+- -+ +	+- -+ *
man	n/a	-+** *	- -+* *
more	+++* +	**** *	-++* *
mymoreproc	++** *	**** *	+*** *
pwd	**** *	- -** *	**** *
rm	- - -+ -	- -** *	- -+* *
sort	- - -+ *	- - - - *	- -++ *
stty	-++* *	-*** *	++** *
vi	- - - - *	*- - - *	- - - - *
Total *	41	69	62
Total +	28	22	39
Total -	51	31	29
Total Detections	69	91	101

This table summarizes the results of the true-positive tests performed during the concept, verification, and refinement experiments for a short-term profile half-life of 1 for the concept experiment and 0.1 for the verification and refinement experiments. Each half-life column contains one character indicating the results of the masquerading record going through the application's profile. We have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection. At the bottom of the table we include counts of the number of "red" threshold detections (*), "yellow" threshold detections (+), non-detections (-) and the total detections (red plus yellow) for each experiment.

Table 3.17: True-positive Comparison for Three Experiments



This figure shows average S scores for the four masquerader records against all subjects. S values that are high on this chart indicate measures in which the masquerader data is most different from the normal subject data. MEMCMB and OPENF are consistently high. For several other measures, some but not all of the masquerader data have high average S scores (for example, SCPU is moderate for record 3 and high for record 4), indicating some variability within the pool of masquerader records.

Figure 3.11: Average Measure S-values over all Three experiments

3.7 Cross-profiling Experiment

Cross profiling is the term we use when running one subject's audit data through another subject's long-term profile. The purpose of cross profiling is two-fold:

- Cross profiling allows us to determine how unique an application's profile is and how successful other applications might be in trying to masquerade as the "host" application.
- By examining the true-positive rate from cross profiling we can also obtain extensive information on the similarities or differences of profiles. This can be used as an aid for determining possible groupings of subjects.

3.7.1 Configuration

For the cross-profile experiment, we used the same configuration as in the refinement experiment (12 measures) and the same subjects as those in the true-positive test phase (26 subjects). Because of time constraints, we selected only a short-term profile half-life of 0.1 because this choice seemed most consistent with our desire to identify individual audit records that were masqueraders. We used the same trained profiles and data set that were used for the false-positive tests during the refinement experiment.

We recorded the true-positive percentages (anomalous record rates) for the yellow threshold; these are shown in Tables 3.18 and 3.19.

3.7.2 Results of Cross-Profiling

Table 3.18 and 3.19 show the true-positive percentages for all applications combined during the cross-profiling experiment. Within each cell in these tables are two numbers. The first entry represents the detection rate for the application listed on the column heading against the profile of the subject in the row heading. For example, on the row headed `compile` under the column for `as`, the first number, 48.5% is the rate of detection of as data when run through the profile of `compile`. The second entry in the cell reverses the relationship of the two applications. For example, the number of 67.3% in the `compile` row under the as column is the detection rate of `compile` data when run through the profile of `as`. Comparing the detection rates between applications on these charts shows several possible relationships. In some cases, an application could pass through the profile of another application (a low detection rate), but the other application could not easily pass through the original application. Examples of such asymmetric detection rates include `gawk` and `cp` with rates of 10.7% and 100%, `pwd` and `compile` with rates of 98.1% and 0.1%, and `sort` and `less` with rates of 95.7% and 17.1%. Still others formed pairs or small groups in which applications could mutually pass through each other's profiles. Examples include `csh` and `compile` (4.6% and 2.0%), `ls` and `compile` (3.4% and 2.7%), and `ls` and finger (2.2% and 4.4%). A third case has applications with consistently high rates, indicating that neither application can pass through the others' profile without raising suspicion.

Table 3.20 summarizes the true-positive percentages shown in Tables 3.18 and 3.19. Table 3.20 shows the minimum, maximum, and average true-positive rates with the yellow threshold (1%) for each application. The chart also shows the percentage of applications that were in each of the following ranges of rates: 0 to 25%, 26 to 50%, 51 to 75% and 76 to 100%. For example, for the application as, 28% of the applications had a detection rate between 0 and 25% when run against as's profile, 16% had a detection rate between 26 and 50%, 24% had a detection rate between 51 and 75% and 32% had a detection rate between 76 and 100%.

Subjects with high average detection rates, such as `getfullnm` and `latex,` are very sensitive to potential masquerader data. Others, with low average rates, such as `vi` and `grep,` are more tolerant and would be candidates for a masquerading attempt. Observe that the thresholds in Table 3.13 for `getfullnm` and `latex` are somewhat low, while those for `vi` and `grep` are on the high side. This may explain the low detection rate for masquerader data using `vi` as a host profile and confirms our low detection rates for `vi` under our true-positive tests in the first three experiments.

3.7.3 Cross-profiling Experiment Figures and Tables

The following pages contain figures and tables showing the results of the cross-profiling experiments.

	as	cat	compile	cp	csh	discuss	emacs	finger	font	gawk	getfullnm	ghostview
as	0.4											
cat	98.4 92.7	2.8										
compile	48.5 67.3	8.3 26.5	0									
cp	33.0 83.8	0 11.8	1.4 9.6	0								
csh	38.3 8.8	57.3 95.7	4.6 2.0	69.2 18.8	0.6							
discuss	15.6 0.1	98.4 98.0	88.9 20.0	93.6 31.5	62.9 27.7	0.4						
emacs	83.1 52.1	95.0 99.6	93.1 98.3	83.1 98.8	92.6 99.2	80.5 94.3	1.9					
finger	99.7 54.6	43.1 39.9	17.0 2.3	44.4 15.0	98.7 22.3	93.2 57.8	99.9 90.7	0.7				
font	100.0 99.7	41.2 0	100.0 19.1	57.9 0	100.0 86.6	100.0 99.4	100.0 96.5	99.9 55.8	1.3			
gawk	38.6 1.5	99.7 85.6	67.9 10.4	100.0 10.7	17.47 8.2	14.8 7.0	99.5 87.9	75.2 86.1	100.0 100.0	2.6		
getfullnm	99.9 95.5	92.4 0	99.9 1.5	98.9 0	99.9 0	99.9 96.8	99.9 83.0	99.8 76.1	99.9 100.0	99.8 96.3	2.6	
ghostview	38.2 19.0	98.6 92.1	98.4 98.1	92.3 94.8	97.9 96.3	66.7 74.7	13.9 10.5	96.5 98.1	99.1 100.0	81.6 98.1	95.0 99.5	0
grep	11.1 14.4	0.4 78.6	0 5.5	3.6 13.1	1.3 23.8	4.6 25.7	90.9 98.1	0 19.2	0.5 99.9	0.7 44.2	0 98.9	0.7 93.0
latex	99.1 53.4	98.9 97.6	98.6 100.0	95.8 99.7	98.3 99.3	98.9 98.8	99.8 92.3	97.8 99.5	99.2 100.0	97.8 100.0	95.7 99.9	94.6 5.9
less	59.2 32.6	49.7 33.5	2.1 0.5	61.5 6.7	5.0 6.4	48.9 77.0	99.7 98.8	25.1 33.3	85.6 100.0	30.1 36.4	5.9 99.9	99.1 99.0
ls	70.4 72.3	4.9 18.5	3.4 2.7	9.6 2.9	56.4 15.2	75.6 85.6	98.3 99.0	2.2 4.4	0.7 99.5	36.0 85.1	0.8 97.2	66.6 98.0
mail	24.5 11.2	98.1 89.1	58.5 40.1	92.2 69.5	2.1 46.5	1.2 9.3	99.8 58.1	61.0 86.5	99.3 100.0	3.9 60.7	96.1 99.8	77.9 47.0
make	23.4 1.0	85.1 98.2	25.4 18.6	72.9 39.5	5.7 31.9	1.7 7.2	66.9 80.1	46.5 99.5	97.9 100.0	4.3 32.3	72.0 99.9	62.3 74.4
man	99.9 70.4	56.3 74.3	87.2 10.6	57.3 15.0	99.2 48.3	99.6 92.3	99.9 92.8	18.7 61.6	89.9 99.2	88.4 94.0	37.6 99.9	99.1 97.9
more	56.5 42.0	35.0 32.7	2.2 1.1	27.3 4.2	3.3 6.2	41.4 71.9	97.6 98.8	15.3 38.7	58.1 100.0	20.1 55.9	2.1 99.7	64.8 99.2
mymoreproc	100.0 99.9	22.5 0	100.0 10.7	27.7 0	100.0 58.7	100.0 99.7	100.0 98.4	100.0 41.8	3.3 34.8	100.0 100.0	50.8 99.9	100.0 99.6
pwd	99.2 95.9	77.2 0.1	98.1 0.1	87.3 0	98.3 0.5	98.3 99.1	99.8 95.5	96.9 13.7	85.3 67.3	97.4 96.1	29.6 71.5	94.6 98.8
rm	71.3 95.2	0.1 4.9	1.2 17.2	2.4 0.6	66.2 47.3	78.9 99.5	99.0 97.5	27.3 46.5	0 52.3	37.1 99.9	0 99.7	89.3 99.3
sort	97.7 36.3	85.8 56.4	99.2 5.2	87.4 0.4	99.0 31.7	99.3 97.0	99.7 92.5	42.6 20.6	95.2 100.0	97.8 78.9	42.9 99.9	96.8 97.7
stty	99.0 100.0	96.4 0.7	98.2 31.3	94.8 0.1	97.9 54.5	98.6 99.0	99.8 94.0	92.7 3.4	98.6 99.1	97.3 100.0	94.7 99.9	93.4 98.4
vi	8.0 41.1	0.7 70.8	0 56.3	4.3 45.9	2.6 88.9	0.8 70.2	37.0 64.1	0.3 99.9	0 100.0	0.2 87.8	0.3 99.9	28.6 68.9

This table shows the true-positive percentages for all applications that were used for the cross-profiling experiment. To read this chart, let subject A be the host represented by the row, and subject B be the subject represented in the column. Each cell contains two values: the top value is the detection rate of B's data being run through A's profile, and the bottom number is the detection rate of A's data being run through B's profile.

Table 3.18: Cross Profile Chart (26 subjects, 1% threshold level)

	grep	latex	less	ls	mail	make	man	more	mymoreproc	pwd	rm	sort	stty	vi
grep	0.7													
latex	96.7 98.7	1.5												
less	14.0 0.1	99.7 99.8	0.6											
ls	9.3 0.1	87.9 99.8	18.7 14.2	0.9										
mail	17.1 12.0	98.6 97.7	29.9 64.3	82.7 69.0	1.5									
make	16.8 5.3	96.4 98.9	13.6 51.6	62.2 83.3	8.8 6.7	1.7								
man	7.9 0	99.7 98.3	19.4 35.4	5.9 7.6	75.3 1.5	99.5 46.1	1.1							
more	14.3 0.1	86.9 99.8	2.5 2.7	7.4 15.7	52.2 45.3	44.6 37.4	29.0 30.1	0.7						
mymoreproc	99.5 1.6	100.0 99.7	100.0 44.1	99.9 13.5	100.0 99.7	100.0 94.7	100.0 46.2	100.0 19.5	1.5					
pwd	98.9 0	98.4 99.0	78.4 1.0	97.6 0.1	97.7 94.1	98.9 95.2	97.9 16.3	92.6 0.2	78.3 24.5	0.9				
rm	31.3 1.6	97.0 99.5	14.2 97.0	4.4 6.9	73.8 98.9	80.4 92.6	21.4 86.3	8.0 59.4	0 33.3	0 97.5	0.4			
sort	86.5 0.1	99.1 98.3	95.7 17.1	22.4 2.1	98.7 98.3	97.1 24.9	94.0 43.8	87.9 5.3	90.6 100.0	68.7 98.3	97.3 5.6	0.9		
stty	98.7 0	98.0 98.6	99.7 99.4	91.5 0	97.2 98.7	98.6 96.8	96.5 19.4	99.8 50.2	99.6 98.7	98.7 98.6	49.6 0	97.0 6.8	0.8	
vi	3.1 21.7	54.6 99.7	0.4 88.5	0.2 82.2	0.2 80.3	5.6 32.2	4.5 99.9	0.3 81.4	0 100.0	0 99.7	1.1 61.5	0.1 86.2	0.1 99.7	1.8

This table shows the true-positive percentages for all applications that were used for the cross-profiling experime
To read this chart, let subject A be the host represented by the row, and subject B be the subject represented in the column. Each cell contains two values: the top value is the detection rate of B's data being run through A's profile, and the bottom number is the detection rate of A's data being run through B's profile.

Table 3.19: Cross Profile Chart-Continued (26 subjects, 1% threshold level)

	True-Positive Percentages			Applications Distribution			
	(%)	(%)	(%)	(%)	(%)	(%)	(%)
Application	Minimum	Maximum	Average	0-25%	26-50%	51-75%	76-100%
as	0.10	100	53.62	28	16	24	32
cat	0.00	98.84	52.13	32	16	12	40
compile	0.90	98.33	24.71	72	12	4	12
cp	0.00	98.77	24.07	68	16	4	28
csh	0.00	99.17	42.76	36	28	16	20
discuss	7.04	99.49	72.87	16	4	20	60
emacs	10.53	98.80	86.23	4	0	8	88
finger	3.37	99.91	59.23	24	24	8	44
fmt	34.82	100.00	90.02	0	8	12	80
gawk	14.76	100.00	75.15	8	16	12	64
getfullnm	71.54	99.98	98.24	0	0	4	96
ghostview	5.92	99.31	82.21	8	8	12	12
grep	0.00	90.94	13.38	88	0	0	12
latex	94.62	99.81	98.53	0	0	0	100
less	1.02	99.69	48.04	36	20	16	28
ls	0.00	87.87	32.84	60	1	16	20
mail	1.15	99.68	64.27	24	12	8	56
make	1.70	96.83	51.27	36	16	20	28
man	5.88	99.96	63.32	24	16	12	48
more	0.16	86.85	35.06	44	24	20	12
mymoreproc	3.29	100.00	82.40	12	8	4	76
pwd	29.64	99.74	91.82	0	4	0	96
rm	0.00	97.01	34.81	52	12	16	20
sort	6.84	99.32	82.96	8	8	4	80
stty	49.59	99.76	95.43	0	4	0	76
vi	0.00	36.97	6.19	88	8	4	0

This table shows the true positive detection percentages obtained by running each subjects' data through the profiles for all other subjects ("cross-profiling"). The first three columns are the minimum, maximum, and average detection percents. For example, for the application as, at least one "guest" subject had only 0.1% of its records declared anomalous against as, and at least one subject had all its records so declared. Across all guest subjects, the average detection rate of the as profile was 53.62%. The last four columns give a breakdown of what percent of guest subjects were detected at various levels. For example, referring to as, we see that 28% (7 subjects) had a detected rate between 0 and 25%, that is 0 to 25% of the subjects' data when run through as' profile was flagged as anomalous, 16% (4 subjects) were detected between 26 and 50% of the time, 24% (6 subjects) were at a 51 and 75% rate, and 32% (8 subjects) were over a 75% rate.

Table 3.20: True-positive Percentages for Cross-Profiling Experiment

3.8 Grouping Experiment

Using the results from the cross-profiling experiment, we performed a grouping experiment to determine whether the development of profiles of application groups would enhance our detection abilities.

We used two approaches to examine to what degree applications can be classified into groups: grouping using the NIDES scoring mechanism, and grouping by long-term profile overlap.

The first approach (grouping by NIDES scoring mechanism) gives a sense of how similar application groups look to NIDES. We used profiles constructed by using the 0.1 short-term profile half-life and examined all pairwise false-positive rates at the 1% ("yellow" threshold) cutoff (see Section 3.7). For the 26 applications, there are 650 pairs (26 times 25) of "A vs. B" and "B vs. A" (note that these are not symmetric).

The second approach (grouping by long-term profile overlap) helped us assess the usefulness of a default profile for a new application based on some hypothesized functional or other grouping.

3.8.1 Grouping by NIDES Scoring Mechanism

With respect to identifying potential subject groupings from the cross-profiling experiment, there were some interesting observations. For some application pairs, both true-positive rates were very high (over 90%). For many other pairs, the true-positive rates were very different. For example, `rsh` activity masquerading as `getfullum` had a 99.970 true-positive rate, while `getfullnm` masquerading as `rsh` had a true-positive rate of 0. This indicates that the profile for `rsh` (either in categories or in its variance about those categories) in some sense encompasses that for `getfullnm` (i.e., the values typically seen for `getfullnm` are a subset of the values typically seen for `rsh`, but not vice-versa.) `Getfullnm` is one of the applications with the highest overall detection sensitivity.

Similar applications are those for which both true-positive rates are low. We selected all application pairs for which both true-positive rates were under 20% and came up with the list of application pairings shown in Table 3.21. Only 33 of the possible 650 application pairs passed even this fairly generous criterion, indicating that even these groupings might be statistical quirks. It is apparent that applications that warrant grouping are rather rare. To-further explore which applications we might group we defined a distance measure

$$100 * (1 - (1 - p(AvsB))(1 - p(BvsA)))$$

where p(A vs. B) and p(B vs. A) are the true-positive rates of two mutually masquerading applications, A masquerading as B, and B masquerading as A (see Table 3.19).

We observed that the most similar application pairs are not functionally related (only less and more have a distance measure of less than 5 and are functionally related). Only one editor `(emacs)` made the list, exhibiting a weak similarity to `ghostview.` We conclude from this that the NIDES scoring mechanism (with this measure set) concludes subjects are similar probably based on aggregate resource usage or large variance about this usage (for example, use of a large number of files or a highly variant number of files from instance to instance) rather than on shared functionality. Of course, NIDES knows nothing of the applications' functionality.

3.8.2 Grouping by Long-Term Profile Overlap

An alternative approach to grouping considers the similarity of the long-term profiles for two applications. On a per-measure basis, we examined the combined category set for each application pair, and took the sum across bins of the smaller of the two category probabilities. If, for example, category *X* has probabilities 0.3 for *A* and 0.1 for *B,* then category *X* contributes 0.1 to the sum. For a category that has nonzero probability in one profile and zero in the other, the contribution is zero. It is thus apparent that a measure for which perfect overlap exists has a similarity of 1, whereas totally different profiles have a similarity of 0. The per-measure similarities are then averaged across measures, resulting in an overall similarity between 0 and 1 for the profiles of any two applications. Note that, unlike the NIDES scoring mechanism, this grouping measure is symmetric, so that only half the earlier number of application pairs need be considered.

3.8.3 Group Selection

Based on the combination of these two approaches, we selected two subject groups. In addition, we decided to form another grouping of applications recommended to us by TIS, based on functionality of application. Note that for the third group, we were only able to include subjects for which we had trained profiles. TIS' list had more subjects, but insufficient audit data was available to include these. The three groups were:

- **GroupA -** `compile`, `less`, and `more`. This grouping is derived from a combination of the NIDES scoring mechanism and a profile overlap analysis.
- **GroupB -** `cat`, `cp`, and `rrn`. This grouping is derived from a combination of the NIDES scoring mechanism and a profile overlap analysis.
- **GroupC -** `cat`, `fmt`, `less`, `man`, `more`, and `sort`. This is the suggested functional grouping from TIS.

We refer to GroupA and GroupB as the NIDES-based groupings because they were obtained as the best results of mutually low within-group detection rates using the NIDES scoring mechanism and a high degree of profile overlap. GroupC is obtained from a hypothesized functional relation (here, file manipulation) irrespective of how similar the members look to NIDES.

Table 3.22 shows the distance measures for the NIDES-based groupings. Values near zero indicate a high degree of similarity; note that the within-group similarities are much higher than the between-group similarities. The diagonals are the distance of an application from itself, essentially twice the false-positive rate. Table 3.23 presents the same distances for the functional grouping. Note that these distances are generally much greater than for the NIDES-based groupings. Tables 3.24 and 3.25 present the profile overlap statistics for the same groupings; in these tables, a value near 100 indicates a high degree of overlap. Again we observe significantly greater overlap for the NIDES-based groups.

3.8.4 Configuration

For the grouping experiment configuration, we used the same 12 measures as in the refinement experiment, and used the short-term profile half-life of 0.1.

We did a short version of the cross-profiling tests by running individual subject data (the same 26 subjects from the cross-profiling experiment) through the three group profiles. We also conducted true-positive testing using the resource-intensive application data.

3.8.5 Results

Table 3.26 summarizes the score thresholds for both the yellow and red thresholds for the three groups that were profiled for this experiment.

Table 3.27 shows the false-positive rate of individual group members' data running through its group profile. The charts are similar to the false-positive tables in the previous experiments.

For purposes of the grouping experiment, we define a "false-positive" as a detection for a subject run against a group profile of which it is a member, and a "true-positive" as a detection for a subject run against a group profile of which it is *not* a member.

The group profiles are additionally dominated by members who contribute the bulk of the training data (for example, more contributes about five times as much data to the GroupC profile as does man - see Table 3.27). This can lead to high false-positive rates for individual group members while still maintaining an approximately correct rate for the group as a whole (see the GroupC false-positive rates for man shown in Table 3.27).

Table 3.28 summarizes the true-positive rates for the group profiles. Generally, we expect a group profile to be somewhat less sensitive in detection of anomalous data. This was weakly true for the NIDES-based groupings, although it is important to point out that profiles for members of the NIDES-based groupings were more generic and forgiving as compared to other profiles. The functional grouping was seen to be less sensitive (as compared to the sensitivity of its component members) against the resource-intensive data and significantly less sensitive when examining a true-positive rate based on cross-profiling.

Table 3.29 shows the anomaly rates for all experiment subjects running through each of the group profiles. Due to time constraints, we used only the 0.1 short-term profile half-life for this experiment. Both the yellow and red anomaly rates were recorded. Note that this table has the rows as guest data, and the columns are the host profiles. In particular, note that applications with similar functionality (as indicated by the TIS grouping) do not necessarily appear similar as indicated by the NIDES-based grouping.

3.8.6 Grouping Experiment Figures and Tables

The following pages contain figures and tables showing the results of the grouping experiments.

Application-x	Application-y	%(xy)	%(yx)	Distance(%)
				Distance = 100*(1- (1-pxy)(1-pyx))
less	compile	2.1	0.5	2.6
rm	cp	2.4	0.6	3.0
more	compile	2.2	1.1	3.3
rm	cat	0.1	4.9	5.0
more	less	2.5	2.7	5.1
grep	compile	0.0	5.5	5.5
ls	compile	3.4	2.7	6.0
ls	finger	2.2	4.4	6.5
csh	compile	4.6	2.0	6.5
man	grep	7.9	0.0	7.9
make	discuss	1.7	7.2	8.8
more	rsh	3.3	6.2	9.3
ls	grep	9.3	0.1	9.4
mail	discuss	1.2	9.3	10.4
cp	compile	1.4	9.6	10.9
rm	ls	4.4	6.9	11.0
less	rsh	5.0	6.4	11.1
cp	cat	0.0	11.8	11.8
ls	cp	9.6	2.9	12.2
man	ls	5.9	7.6	13.1
less	grep	14.0	0.1	14.1
more	grep	14.3	0.1	14.4
make	mail	8.8	6.7	14.9
discuss	as	15.6	0.1	15.7
grep	cp	3.6	13.1	16.2
rm	compile	1.2	17.2	18.2
grep	finger	0.0	19.2	19.2
gawk	discuss	14.8	7.0	20.8
more	ls	7.4	15.7	21.9
ls	cat	4.9	18.5	22.5
ghostview	emacs	13.9	10.5	22.9
grep	as	11.1	14.4	23.9
ls	less	18.7	14.2	30.2

This table shows the true positive percents for application pairs with mutual detection rates below 20%. The first column is arbitrarily labeled "Application x" and the second "Application y", The third column is the detection (true positive) rate for "Application x" through the profile of "Application y", while the second is the converse detection rate. The final column is the distance given by the above formula, which is a symmetric measure of how far apart the two subjects appear to the NIDES scoring mechanism. Observe that for sufficiently low mutual detection rates, the distance is essentially the sum of the detection rates.

Table 3.21: Application Pairs for which Both True-positive Rates are ≤ 20 %

	GroupA			GroupB		
Application	compile	less	more	rm	cp	cat
compile	0.0	2.6	3.3	18.2	10.9	32.6
less	2.6	1.2	5.1	97.4	64.1	66.6
more	3.3	5.1	1.4	62.6	30.4	56.3
rm	18.2	97.4	62.6	0.8	3.0	5.0
cp	10.9	64.1	30.4	3.0	0.0	11.8
cat	32.6	66.6	56.3	5.0	11.8	5.6

The distance measure from Table 3.21 is shown here for applications comprising the NIDES-based groupings described in section 3.8. The symmetry of the distance measure makes the above table symmetric. The subjects **compile, less, more** make up GroupA, and the subjects **rm, cp, cat** make up GroupB. The two diagonal 3 by 3 blocks represent the within-group distances for each group; observe that these are fairly low, indicating that NIDES finds the component subjects very similar. The off-diagonal 3 by 3 blocks give an idea of the between-group distance; observe that these distances are substantially higher. The diagonals give false-positive rates, which for the purposes of this table may be considered to be a subject's distance from its own profile.

Table 3.22: Distance Based on Scoring Mechanism - NIDES-Based Grouping

Application	cat	fmt	less	man	more	sort
cat	5.6	n/a	67.6	88.8	56.3	93.8
fmt	n/a	n/a	n/a	n/a	n/a	n/a
less	67.6	n/a	1.2	47.9	5.1	96.4
man	88.8	n/a	47.9	2.2	50.4	96.6
more	56.3	n/a	5.1	50.4	1.4	88.5
sort	93.8	n/a	96.4	96.6	88.5	1.8

The distance measure from Table 3.21 is shown here for applications comprising the functional-based grouping described in Section 3.8. The symmetry of the distance measure makes the table symmetric. This table gives the same information as Table 3.22 but for GroupC, which is based on functional grouping irrespective of observed behavioral similarity. The diagonals give false positive rates, which for the purposes of this table may be considered to be a subject's distance from its own profile. Observe that the within-group distances are much higher than the within-group distances shown in Table 3.22.
Note: **fmt** score data was not available because **fmt** was excluded from the cross-profiling experiment.

Table 3.23: Distance Based on Scoring Mechanism - Functional Grouping

	GroupA			GroupB		
Application	compile	less	more	rm	cp	cat
compile	100.0	74.4	68.9	49.4	47.2	48.1
less	74.4	100	79.1	54.1	55.1	58.6
more	68.9	79.1	100	57.5	60.1	59.4
rm	49.4	54.1	57.5	100	86.0	83.6
cp	47.2	55.1	60.1	86.0	100	84.0
cat	48.1	58.6	59.4	83.6	84.0	100

This table shows the profile overlap statistics for members in GroupA and GroupB, the NIDES-based groupings. Each cell contains the percentage of profile overlap of each group member with another group member (note that the tables are symmetric, and that a member compared to itself is always 100%).

Table 3.24: Profile Overlap - NIDES-Based Grouping

Application	cat	fmt	less	man	more	sort
cat	100	83.6	58.6	49.1	59.4	65.8
fmt	83.6	100	49.3	39.0	49.0	59.9
less	58.6	49.3	100	62.3	79.1	77.6
man	49.1	39.0	62.3	100	67.3	56.6
more	59.4	49.0	79.1	67.3	100	64.9
sort	65.8	59.9	77.6	56.6	64.9	100

This table shows the profile overlap statistics for members in GroupC, the functional-based grouping. Each cell contains the percentage of profile overlap of each group member with another group member (note that the tables are symmetric, and that a member compared to itself is always 100%).

Table 3.25: Profile Overlap - Functional-Based Grouping

		Half-life=2		Half-life=1		Half-life=0.1	
Subject	# of Records	Yellow	Red	Yellow	Red	Yellow	Red
GroupA	12044	3.6	5.3	3.7	5.5	3.5	5.6
GroupB	3515	4.0	6.7	3.9	6.3	3.9	6.3
GroupC	15196	3.3	4.9	3.4	5.5	3.4	5.8

This table shows the score thresholds for each application computed during the group profile-training process. Each application group is listed with the number of records used for training. For each group, we have recorded the yellow and red threshold score values calculated for each of the short-term profile half-lives used for the grouping experiment (2, 1 and 0.1). The "yellow" column represents the score value at which records will be flagged at a warning level, and the "red" column indicates the score value at which records will be flagged at a critical level.

Table 3.26: Score Thresholds for Group Experiment

		# of Recs	# of Recs	Half-life=2		Ha lflife=1		Half-life=0.1	
Subject	Group	Training	Testing	Yellow	Red	Yellow	Red	Yellow	Red
compile	GroupA	838	172	0.7	0.0	0.7	0.0	1.1	0.0
less	GroupA	4709	700	0.5	0.1	0.7	0.3	0.7	0.1
more	GroupA	6497	1518	0.6	0.0	0.8	0.1	0.7	0.1
cat	GroupB	1058	137	1.1	0.0	1.7	0.0	0.9	0.0
cp	GroupB	273	105	2.7	0.0	4.1	0.0	3.5	0.6
rm	GroupB	2184	355	0.8	0.0	0.5	0.3	1.0	0.0
cat	GroupC	1058	137	1.8	0.4	1.4	0.4	0.3	0.1
fmt	GroupC	1522	297	0.2	0.0	0.2	0.0	0.1	0.0
less	GroupC	4709	700	0.7	0.2	1.0	0.2	0.8	0.1
man	GroupC	708	230	31.4	3.0	20.4	0.0	4.0	0.0
more	GroupC	6497	1518	0.9	0.1	1.1	0.1	0.7	0.2
sort	GroupC	702	189	3.4	0.6	2.2	0.3	0.4	0.1

This table shows the false-positive rates of an individual application running through its own group profile. As in previous false-positive tables, this table shows both the yellow and red threshold rates for three short-term profile half-lives (2, 1 and 0.1). We also indicate the number of records used for training the group profiles and for performing the false-positive testing for each application (the sum of the records used for training each group should be equal to the total number of records used for the group profile training listed in Table 3.26).

Table 3.27: False-positive Rates for the Group Experiment

	Maquerader 1			Masquerader 2		
Subject	HL=2	HL=1	HL=0.1	HL=2	HL=1	HL=0.1
GroupA	- -+*	- -+*	-++*	*	*	*
GroupB	- -++	- -+*	- -+*	*	*	*
GroupC	- - -*	- -+*	- - -+	*	*	*

This table summarizes the true-positive rates for the group profiling experiment using the two masquerader data sets. For each group, we summarize the results *of* the test for each short-term profile half-life used (2, 1, and 0.1). Each half-life column contains five characters indicating the results of each of the five masquerading records data going through the group's profile. For each half-life, we have recorded an asterisk (*) for detection above the critical (red) threshold, a plus (+) for detection above the warning (yellow) threshold, and a dash (-) for no detection.

Table 3.28: True-positive Chart for Grouping Experiments

	Host Profile					
	GroupA		GroupB		GroupC	
	Yellow	Red	Yellow	Red	Yellow	Red
Guest Data	%	%	%	%	%	%
as	54.8	32.2	69.1	14.4	46.6	35.8
cat (B,C)	36.9	1.2	0.9	0.0	0.3	0.1
compile (A)	1.1	0.0	1.3	0.0	0.7	0.0
cp (B)	27.5	4.4	3.5	0.6	4.8	0.7
csh	4.6	1.1	13.3	0.9	5.1	0.6
discuss	39.7	11.8	70.2	17.3	49.6	11.3
emacs	97.8	93.7	98.7	96.8	96.9	84.5
finger	11.1	0.0	25.5	0.0	2.4	0.0
fmt (C)	53.9	0.2	0.0	0.0	0.1	0.0
gawk	18.3	2.6	34.2	0.7	14.1	2.1
getfullnm	4.4	0.0	0.0	0.0	0.2	0.0
ghostview	64.4	62.9	82.7	74.4	59.0	52.0
grep	13.9	3.9	27.7	3.1	7.8	2.6
latex	86.6	86.6	94.6	94.6	84.0	83.9
less (A,C)	0.7	0.1	9.9	0.2	0.8	0.1
ls	7.7	0.5	3.8	0.1	2.4	0.1
mail	51.2	29.5	66.4	47.4	52.1	26.6
make	37.5	7.8	71.8	16.6	33.6	5.8
man (C)	27.2	7.8	9.4	0.1	4.0	0.0
more (A,C)	0.7	0.1	5.3	0.3	0.7	0.2
mymoreproc	19.2	0.1	0.0	0.0	0.1	0.0
pwd	0.5	0.0	0.0	0.0	0.1	0.0
rm (B)	66.9	4.7	1.0	0.0	1.8	0.2
sort (C)	9.6	0.1	6.1	0.3	0.4	0.1
stty	64.5	2.7	0.0	0.0	0.3	0.0
vi	80.2	33.0	58.6	27.2	38.5	15.8

This table shows the anomaly rates resulting from running cross-profile tests with groups. To read this table, we refer to the *host* as the group whose profile was used and *guest* as the application whose data was used to run against the host's profile. For example, the detection rate for running cp's audit data through GroupB's profile is 3.5% and 0.6% respectively for the yellow and red threshold levels. Next to the guest application names we also note whether the guest was a member of a group or groups, and the group(s) it was a member of.

Table 3.29: Anomaly Rates for Group Cross profiling - Half-life = 0.1

Chapter 4

Conclusions

Overall, the NIDES statistical component was successful in its efforts to detect masquerading applications, whether these were special resource-intensive applications or standard applications run through historical profiles other than their own. The primary conclusions drawn from the experiments undertaken as part of the NIDES Safeguard project are as follows:

- The set of measures available for differentiating applications appears generally sufficient for this task. Identifying the relevant set of 12 measures permitted a high degree of differentiation of applications with acceptable false-positive rates.

- NIDES can be and has been modified for the task of detecting when an audit stream has been infected with data from a masquerading application. In addition NIDES can be and has been modified so that it can detect individual audit records from masquerading applications. The modifications required for Safeguard were minor, having to do mainly with profile aging rates. The motivation for the more rapid profile aging used here versus that used for user monitoring is that the Safeguard data captures in a single audit record the total information set for the execution of an application;information that is typically spread over hundreds of audit records of user-level Unix audit data. In addition, the shorter profile length is more appropriate for identifying single abnormal audit records.

- Masquerading applications (from which we had five audit records) can usually be detected by NIDES, with 101 detections in 130 opportunities. With respect to the four audit records generated by one of these resource-intensive applications, we were able to detect one or more of these individual records at the red threshold for 19 of 26 host applications. For three additional hosts there were three or more detections at the yellow threshold. For only two hosts were there no detections or only a single detection at the yellow threshold. Results with respect to the single audit record from the second resource-intensive application were even better. We were able to detect this individual record at the red threshold for 25 out of 26 host applications, and at the yellow threshold for the remaining host application. If these audit records from the sample masquerading applications are representative of typical masqueraders, we believe that NIDES will be generally successful in detecting them.

- Not all pairs of applications can be differentiated. Some applications are very "generic" with respect to the available measures and can easily masquerade as other applications. A few other applications with highly variable behaviors are particularly "forgiving" hosts. These latter applications are generally more vulnerable to masquerader data successfully slipping through.

- NIDES generally has a reasonably high probability of detecting individual audit records that are masquerading as other applications. There are exceptions that appear to be partially the result of highly variable behavior by the "host" application and partially the result of similar behavior by the masquerading and "host" applications.

- For some applications, it would be desirable to have training data that stretches over additional months of calendar time, because these applications demonstrate "shifts" in behavior over different months.

- Using the long-term profiles, it was possible to assess whether different applications could be grouped. Very few applications appeared to cluster together, and the derived clusters were quite small. In the few cases where we identified reasonable clusters, all or most of the members of the cluster were "forgiving" hosts.

- Clustering on the basis of functionality (i.e., groupings suggested by TIS on the basis of the function of the software) did not yield cohesive groups. The historical profile developed by NIDES was more forgiving than would be expected based on the detection sensitivity of the individual members. Because of this, we believe that clustering or grouping of applications is generally not desirable. Most applications are sufficiently different from all other applications that clustering only reduces the ability of NIDES to detect masqueraders.

In spite of the overall high detection rates of resource-intensive application data, we caution that a base of five masquerading audit records is too small to lead to definitive conclusions, and there has never been an assertion that these five audit records are representative of all masqueraders. Therefore, we also performed cross-profiling experiments, wherein each of the 26 applications for which we could build profiles acted as a masquerader for each of the other 25 applications. The proportion of audit records detected during cross-profiling is a lower bound on our ability to detect "true" masqueraders, under the assumption that masqueraders are very different from the applications that were our "normal" subjects. These results were very encouraging. For 16 of the 26 host applications, we were able to detect at the red threshold 50% or more of the masquerading audit records. For only two applications were the average percentage of masqueraders identified at the red threshold below 15%. However, detection capability was not uniform across masqueraders, and most applications could not be differentiated from at least one masquerading "normal" application. (In these cases, the masquerading application generated audit records that were within the range of behavior observed for audit records generated by the host application.) Whether the inability to find all potential masquerading applications is a deficiency depends on whether the applications that cannot be adequately differentiated might be representative of the type of masqueraders that pose a security threat.

4.1 Experiment Deficiencies

Our ability to tune NIDES to the Safeguard environment, and our ability to test the detection capability of NIDES, would have been greatly enhanced if the following were available:

- *Additional data from normal applications.* Very few of the applications that we received had sufficient numbers of audit records to build accurate long-term profiles. We were able to profile only 30 out of 288 applications represented in the data. Additional data would allow us to test a wider variety of applications.

- *Additional chronological time for applications.* Some applications demonstrated changes in behavior over time. A longer training period would have allowed that behavior to be reflected in the normal profiles, thus reducing the false-positive anomaly rate.

- *Additional masquerading applications.* We were provided with only two examples of masquerading applications. These probably do not span the range of applications that might be masqueraders.

- *Additional audit records from masquerading applications.* If the "normal" applications are any indication, there can be substantial variation from audit record to audit record within an application. There is no reason to believe that variability is any less substantial within masquerading applications. We would encourage the development of 50 audit records for each masquerading application.

- *Improved scenario development.* In testing NIDES we have taken the extreme position that it is necessary to identify each and every instance of a masquerading application and that, if a stream of audit records contained a single masquerading audit record, it would be important to identify that record. More reasonable scenarios might postulate the presence of multiple masquerading audit records and require that the stream be identified as "infected". We believe NIDES could be configured to find infected audit record streams even more reliably than attempts to identify individual and isolated audit records.

Glossary

Aging factor The factor by which past data are multiplied so as to fade their value at a desired rate. For a half-life of k audit records, for example, the factor is set at the kth root of 1/2, so that after k steps the data are faded to one-half of their original contribution. Storing profiles as aged cumulative totals permits relatively compact profile structures and allows the system to adapt to changes in subject behavior. NIDES has a short-term aging factor applied to each audit record and a long-term factor applied to daily totals at update time.

Bin Table entry to which an observed value is assigned. For categorical measures, such as HOUR, there is a one-to-one correspondence between bins and observed category values. For continuous measures, the bins correspond to value ranges.

Categorical measure A measure that assumes values in discrete categories. For some such measures, such as HOUR, the values are known beforehand (the hours 0, 1,2, ... 23). For others, new categories are allocated by NIDES as they are encountered.

Category An observed value (such as error type or hour of use on a 24-hour clock) for categorical measures, or a value range for a continuous measure such as SCPU.

Continuous measure A measure that takes continuous values, such as SCPU in time units.

Cross profiling An experiment in which data for each subject are tested against the trained profile for each other subject. Long-term profile update **is** disabled for such experiments.

Detection A declaration by NIDES that a stream of audit data contains anomalous activity. They can be at a "yellow" (caution) or "red" (critical) threshold.

Detection rate The percent of audit records in a given audit data stream that trigger detections.

False positive A detection for a subject against its own profile.

Effective n The effective length of the short term profile, which equals the series sum-of all powers of the aging factor (or approximately 1.5 times the short-term half-life, but never less than one).

Half-life The number of audit records (in the case of the short-term profile) or the number of days (in the case of the long-term profile) by which time the contribution of a data item to the present cumulative totals is reduced by one half. The short-term profile half-life for Safeguard is set at a small number of audit records (a half-life of 1 or 0.1, the latter corresponding to a short-term profile of a single record).

Historical effective n The effective count of audit records contributing to the long-term profile. It consists of the sum of all daily totals, each weighted by the appropriate power of the long- term aging factor.

Historical profile See long-term profile.

Long-term profile For each subject and measure, the observed categories and the observed long-term probabilities for each category, the historical effective n, and the empirical Q distributions. For the subject there is also an empirical score (T^2) distribution, which is aggregated across all measures. At the end of each day, this profile is aged by the long-term aging factor and combined with the new daily totals.

Minimum effective n The minimum count of records in the long-term profile that must be accumulated before the scoring mechanism is considered reliable. It is measure specific.

Profile aging rate, half-life and length The concepts of aging rate, half-life, and effective length of a profile are closely related. The aging rate is a multiplicative factor less than or equal to unity, by which the existing information in a profile is aged. The smaller the rate, the more rapidly this information is 'forgotten'. The half-life is the number of audit records that need to transpire before the contribution of a given data item is decayed (downweighted) by one half. For example, if the aging rate is 0.8, the third most recent audit record has a weight of 0.8*0.8*0.8 or 0.512, so we would say that an aging rate of 0.8 corresponds to a short-term profile half-life of three audit records. The effective length of a profile is given by the series sum of all powers of the aging factor, which will converge if this factor is strictly less than 1. For example, an aging rate of 0.8 corresponds to an effective profile length of 5 audit records. As a rule of thumb, the effective length of the profile (in audit records) is approximately 1.5 times the half-life.

Profile realization The short-term profile realization **is** a value that indicates the amount of activity that has contributed to the information in the short-term profile. The larger the number of profile realizations the more observations of subject activity. The number of profile realizations is based on the number of audit records that have contributed to the short-term profile related to the short-term profile length. For example, with a short-term profile half-life of 50 records (which corresponds to an effective short-term profile length of about 75), a training data set of 300 records contains only four independent realizations of the short-term profile.

Profile training The general procedure of updating profiles, adding and dropping categories, and adjusting the empirical distributions for Q and T^2. It proceeds in three stages. In the first, category probabilities are obtained from a number of days of raw data. In the second, the Q distribution is estimated over an additional number of days. Finally, the T^2 distribution is estimated, after which time NIDES is ready to score audit records. In a production environment, profile training continues indefinitely. For experimentation with known masquerader data, profile updating and training are disabled.

Profile update The merging of the historical profile with new information at the end of each day. Long-term probabilities are converted to effective counts (by multiplying by the historical effective n). The new daily counts are summed in, and the results converted back to probabilities. Categories that have too low a probability are folded into a "rare" category, which can change daily.

Q-score A chi-square-like square difference statistic based on the difference between the short- and long-term profiles for each measure.

QMax A scale value used to assign the Q-score into bins to obtain its empirical distribution.

S-value A unitless quantity obtained by inverting the observed Q-score using the Q empirical distribution and a half-normal transform. This results in all measure scores being comparably distributed.

Scalar A value used to scale observed (raw) values to assign them to category (range) bins.

Short-term profile For each subject and measure, the number of counts recently observed for each category in the long-term profile with special handling for new categories. Due to the aging procedure, these counts are generally fractional.

True positive A detection for a subject (possibly a masquerader) against another subject's profile.

Threshold The NIDES-estimated value for T^2 at which a detection is declared. It is set to achieve no greater than some user-specified percent (usually 1% for yellow, 0.1% for red) of false positives.

T^2 The overall NIDES score on which anomalies are declared, aggregated across all measures.

Bibliography

[l] R. Jagannathan, T. F. Lunt, F. Gilham, A. Tamaru, C. Jalali, P. Neumann, D. Anderson, T. D. Garvey, and J. Lowrance. Requirements specification: Next generation intrusion-detection expert system(NIDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, September 1992.

[2] R. Jagannathan, A. Tamaru, F. Gilham, D. Anderson, C. Jalali and C. Dodd, H. S. Javitz, A. Valdes, T. F. Lunt, and P. G. Neumann. Next generation intrusion-detection expert system (NIDES) software design specifications. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, March 1993.

[3] H. S. Javitz and A. Valdes. The SRI statistical anomaly detector. In Proceedings of the 1991 *IEEE Symposium on Research in Security and Privacy,* May 1991.

[4] H. S. Javitz, A. Valdes, T. F. Lunt, A. Tamaru, M. Tyson, and J. Lowrance. Next generation intrusion-detection expert system (NIDES): Statistical algorithms rationale and rationale for proposed resolver. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, March 1993.

[5] T. F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, H. S. Javitz, A. Valdes, P. G. Neumann, and T. D. Garvey. A real-time intrusion-detection expert system (IDES), final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

Computer Science Laboratory
SRI INTERNATIONAL

Technical Reports - 1995

SRI-CSL-95-01, March 1995
Formal Methods and their Role in the Certification of Critical Systems, John Rushby

SRI-CSL-95-02, January 1995
Adaptive Fault-Resistant Systems, Jack Goldberg, Li Gong, Ira Greenberg, Raymond Clark, E. Douglas Jensen, Kane Kim, and Douglas M. Wells

SRI-CSL-95-03, January 1995
Mechanized Formal Verification - Seven Papers, David Cyrluk, Patrick Lincoln, Steven P. Miller, Paliath Narendran, Sam Owre, Sreeranga Rajan, John Rushby, Natarajan Shankar, Jens Ulrik Skakkebaek, Mandayam Srivas, and Friedrich von Henke

SRI-CSL-95-04, July 1995
Formal Verification of a Commercial Microprocessor, Steven P. Miller and Mandayam Srivas.

SRI-CSL-95-05, July 1995
Execution Driven Distributed Simulation of Parallel Architectures, Livio Ricciulli, Patrick Lincoln, and José Meseguer.

SRI-CSL-95-06, May 1995
Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES), Debra Anderson, Teresa F. Lunt, Harold Javitz, Ann Tamaru, and Alfonso Valdes.

SRI-CSL-95-07, May 1995
Next Generation Intrusion Detection Expert System (NIDES): A Summary. Debra Anderson, Thane Frivold, and Alfonso Valdes.

SRI-CSL-95-08, June 1995
Correct Schema Transformations, Xiaolei Qian.

SRI-CSL-95-09, June 1995
Query Folding, Xiaolei Qian.

SRI-CSL-95-11, June 1995
Architecture and Techniques for Fast Computer Tomography, Iskender Agi.

SRI-CSL-95-18, November 1995
View-based Access Control with High Assurance, Xiaolei Qian.

Technical Reports - 1987 - 1994

SRI-CSL-94-01, January 1994
General Logics and Logical Frameworks, Narciso Martí-Oliet and Jose Meseguer.

SRI-CSL-94-02, April 1994
Semantic Interoperation: A Query Mediation Approach, Xiaolei Qian and Teresa Lunt.

SRI-CSL-94-03, March 1994
Elements of Trusted Multicasting, Li Gong and Nachum Shacham.

SRI-CSL-94-04R, March 1994, Revised October 1994
Adaptive Fault Tolerance: Two Papers Jack Goldberg, Li Gong, Ira Greenberg, and Thomas Lawrence.

SRI-CSL-94-05, April1994
A Formal Approach to Correct Refinement of Software Architectures, Mark Moriconi, Xiaolei Qian, and R.A. Riemenschneider.

SRI-CSL-94-06, July 1994
GLU Programmer's Guide Version 0.9, R. Jagannathan and Chris Dodd

SRI-CSL-94-07, April 1994
Action and Change in Rewriting Logic, Narciso Martí-Oliet and Jo& Meseguer.

SRI-CSL-94-08, May 1994
Authentication, Key Distribution, and Secure Broadcast in Computer Networks Using No Encryption or Decryption, Li Gong
Also included: Using One-Way Functions for Authentication, Li Gong
Secure, Keyed, and Collisionful Hash Functions, Thomas A. Berson, Li Gong, and T. Mark A. Lomas.

SRI-CSL-94-10, October 1994
Transformations in High Level Synthesis: Formal Specification and Efficient Mechanical Verification, P. Sreeranga Rajan.

SRI-CSL-94-11, May 1994
Specification, Transformation, and Programming of Concurrent Systems in Rewriting Logic, Patrick Lincoln, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-94-12,Aug. 1994
A Policy Framework for Multilevel Relational Databases, Xiaolei Qian and Teresa F. Lunt.

SRI-CSL-94.14,Oct. 1994
Fail-Stop Protocols: An Approach to Designing Secure Protocols, Li Gong.

SRI-CSL-94-15, Oct. 1994
Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations, Li Gong.

SRI-CSL-93-01, March 1993
Critical System Properties: Survey and Taxonomy, John Rushby.

SRI-CSL-93-02, March 1993
A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model, Patrick Lincoln and John Rushby.

SRI-CSL-93-03, December 1993
Multidimensional Problem Solving in Lucid, A.A. Faustini and R. Jagannathan

SRI-CSL-93-04, May 1993
Eight Papers on Formal Verification, Patrick Lincoln, Sam Owre, Natarajan Shankar, John Rushby, and Friedrich von Henke.

SRI-CSL-93-05, August 1993
Rewriting Logic as a Logical and Semantic Framework, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-93-06, November 1993
A Model-Theoretic Semantics of the Multilevel Secure Relational Model, Xiaolei Qian.

SRI-CSL-93-07, November 1993
Formal Methods and the Certification of Critical Systems, John Rushby.

SRI-CSL-93-08, December 1993
A Lazy Approach to Compositional Verification, by N. Shankar

SRI-CSL-93-09, December 1993
Abstract Datatypes in PVS, by N. Shankar

SRI-CSL-93-10, December 1993
A Duration Calculus Proof Checker: Using PVS as a Semantic Framework, by J.U. Skakkebaek and N. Shankar

SRI-CSL-93-11, December 1993
Linear Logic and Proof Theory: Three Papers, by Patrick Lincoln, Andre Scedrov, Natarajan Shankar, and Timothy Winkler

SRI-CSL-93-12, December 1993
Microprocessor Verification in PVS: A Methodology and Simple Example, by David Cyrluk

SRI-CSL-93-13, December 1993
The Complexity and Composability of Secure Interoperation, by Li Gong and Xiaolei Qian.

SRI-CSL-92-01, July 1992
Formal Verification of an Oral Messages Algorithm for Interactive Consistency, by John Rushby.

SRI-CSL-92-02, July 1992
Noninterference, Transitivity, and Channel-Control Security Policies, by John Rushby.

SRI-CSL-92-03, March 1992
Introducing OBJ, by Joseph A. Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud.

SRI-CSL-92-04, March 1992
Survey of Object-Oriented Analysis/Design Methodologies and Future CASE Frameworks, by Donovan Hsieh.

SRI-CSL-92-05, April1992
A Real-Time Intrusion-Detection Expert System (IDES), by Teresa F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, and Peter G. Neumann.

SRI-CSL-92-06, May 1992
Multiparadigm Logic Programming, by José Meseguer.

SRI-CSL-92-07, July 1992
Simulation and Performance Estimation for the Rewrite Rule Machine, by Hitoshi Aida, Joseph A. Goguen, Sany Leinwand, Patrick Lincoln, José Meseguer, Babak Taheri, and Timothy Winkler.

SRI-CSL-92-08, July 1992
A Logical Theory of Concurrent Objects and its Realization in the Maude Language, by José Meseguer .

SRI-CSL-92-09, Sept. 1992
On the Semantics of Place/Transition Petri Nets, by José Meseguer, Ugo Montanari, and Vladimiro Sassone.

SRI-CSL-92-11, Sept. 1992
Using Rewriting Logic to Specify, Program, Integrate, and Reuse Open Concurrent Systems of Cooperating Agents, by José Meseguer, Kokichi Futatsugi, and Timothy Winkler.

SRI-CSL-92-12, November,1992
Mechanized Verification of Real-Time Systems Using PVS, by N. Shankar.

SRI-CSL-92-13, December, 1992
GLu: A Hybrid Language for Parallel Applications Programming, by R. Jagannathan and A.A. Faustini.

SRI-CSL-92-14, December, 1992
Solving the Inheritance Anomaly in Concurrent Object-Oriented Programming, by José Meseguer .

SRI-CSL-92-15, December, 1992
A Logical Semantics for Object-Oriented Databases, by José Meseguer and Xiaolei Qian.

SRI-CSL-91-01, February 1991
Multilevel Security for Knowledge Based Systems, by Thomas D. Garvey and Teresa F. Lunt.

SRI-CSL-91-02, February 1991
An Introduction to Formal Specification and Verification Using EHDM, by John Rushby, Friedrich von Henke, and Sam Owre.

SRI-CSL-91-03, June 1991
Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems, by John Rushby.

SRI-CSL-91-04, June 1991
Mechanical Verification of a Schematic Protocol for Byzantine Fault-Tolerant Clock Synchronization, by N. Shankar.

SRI-CSL-91-05, February 1991
Conditional Rewriting Logic as a Unified Model of Concurrency, by José Meseguer.

SRI-CSL-91-06, March 1991
Final Algebras, Cosemicomputable Algebras, and Degrees of Unsolvability, by Lawrence S. Moss, José Meseguer, and Joseph A. Goguen.

SRI-CSL-91-07, April1991
From Petri Nets to Linear Logic Through Categories: A Survey, by Narciso Martí-Oliet and José Meseguer .

SRI-CSL-91-08, November 1991
Parallel Programming in Maude, by José Meseguer and Timothy Winkler.

SRI-CSL-90-01, February 1990
Duality in Closed and Linear Categories, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-90002R, February 1990, Revised June 1990
Rewriting as a Unified Model of Concurrency, by José Meseguer.

SRI-CSL-90-03R, February 1990, Rev. Dec. 1990
Compiling Concurrent Rewriting onto the Rewrite Rule Machine, by Hitoshi Aida, Joseph Goguen, and José Meseguer.

SRI-CSL-90-04, August, 1989
Secure Knowledge-Based Systems, by Teresa F. Lunt and Jonathan K. Millen

SRI-CSL-90-06, June, 1990
Order-Sorted Algebra Solves the Constructor-Selector, Multiple Representation and Coercion Problems, by José Meseguer and Joseph Goguen.

SRI-CSL-90-07, July,1990
A Logical Theory of Concurrent Objects, by José Meseguer.

SRI-CSL-90-08, August,1990
Decision Problems for Propositional Linear Logic, by Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar.

SRI-CSL-90-09, October, 1990
On Location: Points about Regions, by Judith S. Crow and Peter B. Ladkin.

SRI-CSL-90-10, October 1990
On the Design of Dependable Computer Systems for Critical Applications, by Peter G. Neumann.

SRI-CSL-90-11, November 1990
The GLU Programming Language, by R. Jagannathan and A.A. Faustini

SRI-CSL-90-12, November 1990
Axiomatizing the Algebra of Net Computations and Processes, by Pierpaolo Degano, José Meseguer, and Ugo Montanari.

SRI-CSL-90-13, December 1990
Semantic Specifications for the Rewrite Rule Machine, by Joseph A. Goguen.

SRI-CSL-90-15, December 1990
A Logic to Unify Semantic Network Knowledge Systems with Object-Oriented Database Models, by Donovan Hsieh.

SRI-CSL-90-16, December 1990
Inclusions and Subtypes, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-90-17, December 1990
Architectural Design of the Rewrite Rule Machine Ensemble, by Hitoshi Aida, Sany Leinwand, and José Meseguer.

SRI-CSL-89-1, January 1989
An Intensional Parallel Processing Language for Applications Programming, by E.A. Ashcroft, A.A. Faustini, and R. Jagannathan.

SRI-CSL-89-2, January 1989
Dataflow-based Methodology for Coarse-Grain Multiprocessing on a Network of Workstations, by R. Jagannathan, Alan Downing, William Zaumen, and Rosanna Lee.

SRI-CSL-89-3R, February 1989, Revised August 1991
Formal Verification of the Interactive Convergence Clock Synchronization Algorithm using EHDM, by John Rushby and Friedrich von Henke.

SRI-CSL-8904R, March 1989, Rev. June 1989
From Petri Nets to Linear Logic, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-89-5, March 1989
General Logics, by José Meseguer.

SRI-CSL-89-6, March 1989
The Rewrite Rule Machine Project, by Joseph Goguen, José Meseguer, Sany Leinwand, Timothy Winkler, and Hitoshi Aida.

SRI-CSL-89-8, July 1989
A Categorical Manifesto, by Joseph A. Goguen.

SRI-CSL-89-9, Sept. 1989
A Practical Approach to Semantic Configuration Management, by Mark Moriconi.

SRI-CSL-89-10, July 1989
Order-sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations, by Joseph A. Goguen and José Meseguer.

SRI-CSL-89-11, December 1989
An Algebraic Axiomatization of Linear Logic Models, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-88-1, January 1988
Higher Order Functions Considered Unnecessary for Higher Order Programming, by **Joseph** Goguen.

SRI-CSL-88-2R2, January 1988, Rev. 2, August 1988
What is Unification? A Categorical View of Substitution, Equation and Solution, by Joseph Goguen.

SRI-CSL-88-3R, January 1988, Rev. June 1989
Petri Nets Are Monoids, by José Meseguer and Ugo Montanari.

SRI-CSL-88-4R2, August 1988
OBJ as a Theorem Prover with Applications to Hardware Verification, by Joseph Goguen.

SRI-CSL-88-5, May 1988
A Descriptive and Prescriptive Model for Dataflow Semantics, by R. Jagannathan.

SRI-CSL-88-7R, Sept. 1988
Quality Measures and Assurance for AI Software, by John Rushby.

SRI-CSL-88-10R, Sept. 1988, Rev. Jan. 1989
Cell, Tile, and Ensemble Architecture of the Rewrite Rule Machine, by Joseph A. Goguen, Sany Leinwand, and Timothy Winkler.

SRI-CSL-88-11, Sept. 1988
Software for the Rewrite Rule Machine, by Joseph A. Goguen and José Meseguer.

SRI-CSL-88-13, Oct. 1988
Relating Models of Polymorphism, by José Meseguer.

SRI-CSL-88-14, Nov. 1988
Reasoning about Design Changes, by Mark Moriconi and Gail A. Harrison.

SRI-CSL-87-1, May1987
The Rewrite Rule Machine Project, by Joseph Goguen, Claude Kirchner, Sany Leinwand, José Meseguer, and Timothy C. Winkler.

SRI-CSL-87-2, May 1987
Concurrent Term Rewriting as a Model of Computation, by Joseph Goguen, Claude Kirchner, and José Meseguer.

SRI-CSL-87-3, May 1987
An Abstract Machine for Fast Parallel Matching of Linear Patterns, by Ugo Montanari and Joseph Goguen.

SRI-CSL-87-7, July 1987
Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics! by Joseph Goguen and José Meseguer.