Annual Report, A010 • March 7, 1994

The NIDES Statistical Component Description and Justification

Harold S. Javitz
Statistics Program

Alfonso Valdes
Applied Electromagnetics and Optics Laboratory

SRI Project 3131
Contract N00039-92-C-0015

Prepared for:

Department of the Navy
Space and Naval Warfare Systems Command
2451 Crystal Drive
Arlington, VA 22245-5200
Attn: SPAWAR 02-22B LCDR Greg Breen
Attn: SPAWAR OOIE2, Robert D. Patton
Attn: SPAWAR PD51E

NRaD, Code 412
NRL, Code 5540
NSA, R23

Preface

SRI International has prepared this document as a full and complete disclosure of the Next-Generation Intrusion-Detection Expert System (NIDES) [3] statistical algorithm, including how it works, what decisions influenced the form of the algorithm, and the rationale behind those decisions. We have divided this document into four sections:

- Chapter 1 is a description of the NIDES statistical algorithm. It describes what the algorithm is and how it functions.
- Chapter 2 is a broad justification for the NIDES statistical algorithm. This section also includes a comparison to other statistical approaches to intrusion detection.
- Chapter 3 is a set of statistical criteria that can be used to evaluate the appropriateness of any statistical approach to intrusion detection. Although we did not formally use these criteria in the development of the NIDES statistical algorithm, they nevertheless had an important influence on the development of the algorithm. They may also be used to evaluate the suggestions of other statistical algorithm developers.
- Chapter 4 is a set of specific questions and answers that can be posed about the NIDES statistical algorithm. In this section we explore in more depth the specific choices we made in developing the NIDES statistical algorithm. We have found it convenient to use the question-and-answer format to address the relationship of the NIDES statistical algorithm to the work of Helman et al.

Contents

Chapter 1

Description of the NIDES Statistical Component

1.1 Overview of Statistical Component

The SRI NIDES statistical component observes behavior on a monitored computer system and adaptively learns what is normal for individual subjects: users, groups, remote hosts and the overall system. Observed behavior is flagged as a potential intrusion if it deviates significantly from expected behavior. The NIDES statistical component maintains a statistical subject knowledge base consisting of profiles. A profile is a description of a subject's normal (i.e., expected) behavior with respect to a set of intrusion-detection measures. Profiles are designed to require a minimum amount of storage for historical data and yet record sufficient information that can readily be decoded and interpreted during anomaly detection. Rather than storing all historical audit data, the profiles keep only statistics such as frequencies, means, and covariances.

The deductive process used by NIDES in determining whether behavior is anomalous is based on statistics, controlled by dynamically adjustable parameters, many of which are specific to each subject. Audited activity is described by a vector of intrusion-detection measures (or variables). Measures can be turned "on" or "off" (i.e., included in the statistical tests), depending on whether they are deemed to be useful for the monitored system. As each audit record arrives, the relevant profiles are retrieved from the knowledge base and compared with the vector of intrusion-detection measures. If the point in N-space defined by the vector of intrusion-detection measures is sufficiently far from the point defined by the expected values stored in the profiles, then the record is considered anomalous. Thus, NIDES evaluates the total usage pattern, not just how the subject behaves with respect to each measure considered singly.

The statistical knowledge base is updated daily, using the most recent day's observed behavior of the subjects. Before the new audit data are incorporated into the profiles, the frequency tables in each profile are aged by multiplying them by an exponential decay factor. Although this factor can be set by the security officer, we believe that a value that reduces the contribution of knowledge by a factor of 2 for every 30 days is appropriate (this is the long-term profile half-life). This method of aging has the effect of creating a moving time window for the profile data, so that the expected behavior is influenced most strongly by the most recently observed behavior. Thus, NIDES adaptively learns subjects'behavior patterns; as subjects alter their behavior, their corresponding profiles change.

1.2 The NIDES Score Value

For each audit record generated by a user, NIDES generates a single test statistic value (the NIDES score value, denoted T^2*)* that summarizes the degree of abnormality in the user's behavior in the near past. (The concept of near past is defined later.) Consequently, if the user generates 1000 audit records in a day, there will be 1000 assessments of the abnormality of the user's behavior. Because each assessment is based on the user's behavior in the near past, these assessments are not independent.

Large values for T^2 are indicative of abnormal behavior, and values close to zero are indicative of normal behavior (e.g., behavior consistent with previously observed behavior). Thus, the security officer should be more concerned about larger values of T^2 than with smaller values.

Using historical information about T^2, cutoff values can be calculated corresponding to various alert levels (with higher alert levels associated with higher T^2 values). Each alert level is associated with a corresponding false positive rate (the probability that a normal user's activities will falsely be declared to be anomalous). The false positive rate can be expressed in two general ways: (1) as the proportion of audit records generated by the normal user that will exceed the threshold for declaring an audit record to be anomalous, or (2) as the probability that a normal user will be declared anomalous sometime during an average day. Currently, we have implemented only the first definition. In future versions of NIDES, we may implement the second definition. When multiplied by the number of system users, the latter definition corresponds loosely to the amount of effort that will be expended by the security officer in tracking down false leads. We have expressed the true positive rate (i.e., the probability that abnormal activity will be declared to be anomalous) as the proportion of "guest" user audit records that exceed the detection threshold (i.e., the proportion of one user's normal audit records that are declared anomalous when "played" through another user's profile, as if the first user had logged on as an uninvited "guest" in the second user's account). If a suite of intrusion scenarios is developed, the definition of true positive rates can be changed to the percentage of the intrusion scenarios that are detected. The security officer decides what actions should correspond to various alert levels, based partially upon the number of false leads that he or she can pursue and partially upon the system security needs. For example, the security officer might choose to completely ignore the lowest alert levels, and be notified only at alert levels corresponding to an average of three false alerts per day.

Because the T^2 statistic summarizes behavior over the near past, and sequential values of T^2 are dependent, the T^2 values will slowly trend upward or downward.

To avoid inundating the security officer with notification of continued alerts we notify the security officer only when a change occurs in the alert level. We also "clear" the alert whenever the T^2 value becomes sufficiently low[1].

1.3 How T^2 is Formed from Individual Measures

The T^2 statistic is itself a summary judgment of the abnormality of many measures taken in aggregate. Suppose that there are n such constituent measures, and let us denote these individual measures by S_i, $1 \leq i \leq n$. Each S_i is a measure of the degree of abnormality of behavior with regard to a specific feature (such as CPU usage or file accesses). In the current version of NIDES, the T^2 statistic has been set equal to the sum of the squares of the S_i:

$$T^2 = (S_1{}^2 + S_2{}^2 + \cdots + S_n^2)\ /n$$

Because the T^2 statistic is a simple average of the n squares of the S_i, T^2 does not explicitly address the correlations among the S_i. We believe that there is additional useful information contained in the correlations among the S_i. Subsequent versions of NIDES could explore ways of introducing this covariation by defining a statistic L^2 as follows:

$$L^2 = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j>1} h(S_i, S_j, C_{ij})$$

Here, $h(S_i, S_j, C_{ij})$ is a well-behaved function of S_i, S_j, and their correlation C_{ij} that takes large values when S_i and S_j are not behaving in accordance with their historical correlations. An audit record would be declared to be abnormal when either T^2 or L^2 exceeded an appropriate threshold.

1.4 Types of Individual Measures

The individual S measures each represent some aspect of behavior. For example, an S measure might represent file accesses, CPU time used, or terminals used to log on. Two S measures might also represent only slightly different ways of examining

[1] The false positive and true positive rates should be calculated as if the modification of the alerting mechanism (to avoid inundating the security officer) had not been implemented. For example, the true positive rate is the proportion of intruder audit records detected as exceeding the threshold, whether or not alerts after the first were suppressed.

the same aspect of behavior. For example, both S_i and S_j might represent slightly different ways of examining file access.

We have found it useful to classify the different types of individual measures in the NIDES statistical system into the following four classes:

- *Intensity measures* - These three measures track the number of audit records that occur in different time intervals, on the order of 1 minute to 1 hour. These measures can detect bursts of activity or prolonged activity that is abnormal, primarily based on the volume of audit data generated.

- *Audit record distribution measure* - This single measure tracks all the types of activity that has been generated in the recent past, with the last few hundred audit records having the most influence on the distribution of activity types. For example, we might find that the last 100 audit records contained 25 audit records indicating that files were accessed, 50 audit records indicating that CPU time was incremented, 30 audit records indicating that I/O activity occurred, and 10 audit records indicating activity from a remote host. These data are compared to a profile of previous activity (generated over the last few months) to determine whether or not the distribution of activity types generated in the recent past (i.e., the last few hundred audit records) is unusual.

- *Categorical measures* - These are transaction-specific measures for which the outcomes are categories. For example, categorical measures could include the names of files accessed, the ID of the terminals used for logon, and the names of the remote hosts used. For the categorical measure of the names of files accessed, the individual categories within that measure are the file names themselves. The names of the files that were used in the last 100 to 200 audit records containing file names are compared to a historical profile of file names used to determine if recent usage is abnormal.

- *Counting measures* - These are measures for which the outcomes are counts. For example, counting measures might include CPU time (which counts the number of seconds of CPU used, with accuracy to 0.001 second) or the amount of I/O. Behavior over the last 100 to 200 audit records is compared to a historical profile of behavior to determine if recent usage is abnormal.

These different classes of measures serve different hierarchical purposes. The intensity measures assess the extent to which the volume of audit records generated is normal. The audit record distribution measure assesses, over the last few hundred audit records generated, the extent to which the types of measures being affected are normal. The categorical and counting measures assess within a type of audit record (e.g., an audit record that involves accessing a file, or that involves incrementing CPU time), the extent to which the behavior is normal over the past few hundred audit records.

1.5 Algorithm for Computing *S* from *Q* for the Intensity Measures

Each S measure is derived from a corresponding statistic that we will call Q. In fact, each S measure is a "normalizing" transformation of the Q statistic so that the degree of abnormality for different types of features (such as CPU usage and the names of files accessed) can be added on a comparable basis. Two different methods for transforming the Q statistics into S values are used. One method is used for computing the values of S corresponding to the three intensity measures; a second method is used for computing the values of S corresponding to all the other measures.

For the intensity measures, the value of Q corresponding to the current audit record represents the number of audit records that have arrived in the recent past. Here, "recent" past corresponds to the last few minutes for the Q statistic with a half-life of 1 minute and to the last few hours for the Q statistic with a half-life of 1 hour. In addition to knowing the current value for Q, NIDES maintains a historical profile of all previous values for Q. Thus, the current value of Q can be compared to this historical profile to determine whether the current value is anomalous.

The transformation of Q to S for the intensity measures requires knowledge of the historical distribution of Q. For example, we might find the following historical information for the intensity measures Q with a half-life of 1 minute:

- 1% of the Q values are in the interval 0 to 10 audit records
- 7% are in the interval 10 to 20
- 35% are in the interval 20 to 40
- 18% are in the interval 40 to 80
- 28% are in the interval 80 to 160
- 11% are in the interval 160 to 320

The S statistic would be a large positive value whenever the Q statistic was in the interval 0 to 10 (because this is a relatively unusual value for Q) or whenever Q was larger than 320 (because this value has not historically occurred). The S statistic would be close to zero whenever Q was in the interval 20 to 40, because these are relatively frequently seen values for Q. The selection of appropriate intervals for categorizing Q is important to the functioning of the algorithm. We are currently using 32 intervals for each Q measure, with interval spacing being either linear or geometric. The last interval does not have an upper bound, so that all values of Q belong to some interval.

Small values of Q are indicative of a recent past that is similar to historical behavior, and large values of Q are indicative of a recent past that is not similar to historical behavior. This induces a modification in the transformation of Q to S, so that S is small whenever Q is small, and S is large whenever Q is large. Hence, S can be viewed as a type of rescaling of the magnitude of Q.

The algorithm for converting individual Q values to S values for the intensity measures (but not for other measures) is as follows:

1. Let P_m denote the relative frequency with which Q belongs to the m^{th} interval. In our example, the first interval is 0 to 10 and the corresponding P value (say P_0) equals 1%. There are 32 values for P_m, with $0 \leq m \leq 31$.

2. For the m^{th} interval, let $TPROB_m$ denote the sum of P_m and all other P values that are smaller than or equal to P_m in magnitude. In our previous example, $TPROB$ for the interval of $40 \leq Q \leq 80$ equal to 18% + 11% + 7% + 1% = 37%.

3. For the m^{th} interval, let s_m be the value such that the probability that a normally distributed variable with mean 0 and variance 1 is larger than s_m in absolute value equals $TPROB_m$. The value of s_m satisfies the equation

$$P(|N(0,1)| \geq s_m) = TPROB_m$$

or

$$s_m = \Phi^{-1}(1 - (TPROB_m/2))$$

where Φ is the cumulative distribution function of a N(0,l) variable. For example, if $TPROB_m$ is 5%, then we set s_m equal to 1.96, and if $TPROB_m$ is equal to l00%, then we set s_m equal to 0. We do not allow s_m to be larger than 4.0.

4. Suppose that after processing an audit record we find that the Q value is in the m^{th} interval. Then S is set equal to s_m, the s value corresponding to $TPROB_m$.

1.6 Algorithm for Computing *S* from *Q* for All Other Measures

For all measures other than the intensity measures, Q compares short-term behavior to long-term behavior. For example, for the command usage measure, Q measures the extent to which the most recent few hundred commands issued are consistent with long-term command usage.

For both the intensity measures and the other measures, we calculate a long-term profile for Q using 32 intervals. For example, for a non-intensity measure such as names of commands used we might find a probability distribution for Q similar to the one displayed earlier for an intensity Q, except that the range of values into which Q could be classified would not be in units of audit records. Rather, the range of values would be expressed in terms of the degree of similarity between the short-term profile of command usage and the long-term profile of command usage, with larger numbers representing less similarity.

Because of the difference in the way that Q is defined for intensity measures and other measures, the transformation of Q to S is slightly different for non-intensity measures. For non-intensity measures, we let $TPROB_m = P_m + P_{m+1} + \cdots + P_{31}$. In our previous example, if Q were a non-intensity measure, the *TPROB* value of the interval $40 \leq Q \leq 80$ would be equal to 18% + 28% + 11% = 49%. Thus, in these cases, S is a simple mapping of the percentiles of the distribution of Q onto the percentiles of a half-normal distribution.

In practice these algorithms are easy to implement, with the Q tail probability calculations done only once - at update time (usually close to midnight). (The s_i values, requiring only a table look up, are done in real time.) Each interval for Q is associated with a single s value, and when Q is in that interval, S takes the corresponding s value.

1.7 Computing the Frequency Distribution for *Q*

The historical frequency distribution for Q is required for Q to be transformed into S. The calculation procedures described here are used for all types of measures.

The first step in calculating the historical probability distribution for Q is to define bins into which Q can be classified. We always use 32 bins (numbered 0 to 31) for a measure Q. Let Q_{max} be the maximum value that we ever expect to see for Q. This maximum value depends on the particular types of measures being considered. Default values are provided in NIDES for these maximum values and they should be reset by the security officer if Q is in the highest bin more than 1% of the time. The cut points for the 32 bins are defined on either a linear or geometric scale. For example, when a geometric scale is used, bin 0 extends from 0 to ${Q_{max}}^{1/32}$, bin 1 extends from ${Q_{max}}^{1/32}$ to ${Q_{max}}^{2/32}$, bin 2 extends from ${Q_{max}}^{2/32}$ to ${Q_{max}}^{3/32}$, and bin 31 extends from ${Q_{max}}^{31/32}$ to infinity.

As before, let P_m denote the relative frequency with which Q is in the m^{th} interval (i.e., bin). Each Q statistic is evaluated after each audit record is generated (whether or not the value of Q has changed), and therefore P_m is the percentage of all audit records for which Q is in the m^{th} interval.

The formula for calculating P_m on the k^{th} day after initiating NIDES monitoring of a user is:

$$P_{m,k} = (1/N_k) \sum_{j=1}^{k} (W_{m,j} 2^{-b(k-j)})$$

where

k = the number of days that have occurred since the user was first monitored

b = the decay rate for P_m that determines the half-life of the data used to estimate P_m; we currently recommend a 30-day half-life, corresponding to a b value of $-log_2(0.5)/30 = 0.0333$

$W_{m,j}$ = the number of audit records on the j^{th} day for which Q was in the m^{th} interval

N_k = the exponentially weighted total number of audit records that have occurred since the user was first monitored

The formula for N_k is:

$$N_k = \sum_{j=i}^{k} W_j 2^{-b(k-j)}$$

where

W_j = the number of audit records occurring on the j^{th} day

The formula for $P_{m,k}$ appears to involve keeping a long sum (e.g., since monitoring first began), but the computations are simplified by using the following recursion formulas:

$$P_{m,k} = (2^{-b} P_{m,k-1} N_{k-1} + W_{m,k}) / N_k$$

$$N_k = 2^{-b} N_{k-1} + W_m$$

In NIDES, we update $P_{m,k}$ and N_k once per day and keep running totals for $W_{m,k}$ and W_k during the day.

1.8 Computing the *Q* Statistic for the Intensity Measures

When a user is first audited, that user has no history. Consequently, we must choose some convenient value to begin the Q statistic history. For example, we might initially let each Q measure be zero, or some value close to the mean value for other similar users.

Each Q statistic for intensities is updated each time a new audit record is generated. Let us now consider how to update Q. Let Q_n be the value for Q after the n^{th} audit record, and Q_{n+1} be the value for Q after the *(n +1) st* audit record. The formula for updating Q is:

$$Q_{n+1} = 1 + 2^{-rt}Q_n$$

where

- · The variable t represents the time (say in minutes or fractions thereof) that has elapsed between the n^{th} and $(n + 1)st$ audit records.
- · The decay rate r determines the half-life of the measure Q. Large values of r imply that the value of Q will be primarily influenced by the most recent audit records. Small values of the decay rate r imply that Q will be more heavily influenced by audit records in the more distant past. For example, a half-life of 10 minutes corresponds to an r value of $0.10 = log_2(0.5)/10.0$. The security officer may set the half-life of the intensity measures at any values that he or she feels are appropriate. We are currently using three intensity measures with half-lives of 1, 10, and 60 minutes, respectively.

Q is the sum of audit record activity over the entire past usage, exponentially weighted so that the more current usage has a greater impact on the sum. Q is more a measure of near past behavior than of distant past behavior, even though behavior in the distant past also has some influence on Q. The Q statistic has the important property that it is not necessary to keep extensive information about the past to update Q.

We note that the intensity measures use clock time as the unit by which age is calculated. This is important because the intent of this measure is to assess the extent to which bursts of activity are normal. All the other measures in NIDES determine "age" using audit record counts. For example, an audit record may be the most recent record (that affects that measure), the second most recent, the third most recent, and so forth. This assures profile continuity over nights and weekends - periods when there is typically little activity. Thus, for the non-intensity measures we address the issue of whether over the past few hundred audit records that affected the measure, behavior was normal in comparison with historical standards, regardless of when that activity took place. (Although at first glance this means that abnormal "short term" behavior might include behavior from days, weeks, or even months in the past, this is not a concern, since the portion of the activity in the short-term profile that has taken place earlier than the last profile updating will have already been incorporated into the historical profile, and will tend to be assessed as normal).

1.9 Computing the *Q* Statistic for the Audit Record Distribution Measure

Each audit record that is generated indicates one or more types of activity that have occurred for a user. For example, a single audit record may indicate that a file has been accessed, that I/O has occurred, and that these activities occurred from a remote host. The Q statistic for the audit record distribution measure is used to evaluate the degree to which the type of activity in the recent past agree with the distribution in a longer-term profile.

The calculation of the audit record distribution measure begins with the specification of the types of activities that will be examined. We currently recommend that each categorical and counting measure (with some exceptions as noted below) constitute an activity type. For example, if name of file accessed is a categorical measure, then the corresponding activity would be that any named file was accessed. Similarly, if amount of I/O used is a counting measure, then the corresponding activity would be that any I/O was used. That is, if an audit record would cause a categorical or counting measure to be recalculated, then a corresponding activity type should be defined. The exception would be a categorical or counting measure that would be affected by any audit record. For example, if hour of audit record generation is a categorical measure, then every audit record causes this categorical measure to be updated and no purpose is served in defining a corresponding activity type. We note that in general a single audit record can impact multiple measures. (That is, a single audit record can “touch” multiple measures.)

It may be useful to define activity types in addition to those based on the categorical and counting measures used. For example, the security officer may want to evaluate the percentage of audit records generated that indicate that the user is logged onto a remote host. The security officer who is not interested in which remote host is being used may accomplish this goal in one of two ways. The first way is to establish a categorical measure with two categories - "remote host indicated” and “remote host not indicated”. If this approach is used, then there should be no corresponding activity type defined because every audit record is relevant to the calculation of the categorical measure. The second way is to define an activity type of “remote host indicated” to be used by the audit record distribution measure and not to define a categorical measure. These two approaches yield similar (although not totally equivalent) results, but the second approach is computationally less intensive.

Suppose that we have established M activity types. For each activity we must calculate a long-term historical relative frequency of occurrence, denoted f_m, for that activity type. For example, suppose that over the last 3 months, 7% of all audit records have involved file accesses. Then f_m for the file access activity type would be 0.07. Note that each f_m is between 0 and 1.0 inclusive. The sum of the f_m may be greater than 1.0 because a single audit record may indicate that multiple activity types have occurred.

The algorithm used to compute f_m is essentially the same as that used to calculate P_m and uses the same decay rate. That is, we may write that the value of f_m on the k^{th} day is equal to

$$f_{m,k} = (1/N_k) \sum_{j=1}^{k} (W_{m,j} 2^{-b(k-j)})$$

where N_k, and b are defined as before and $W_{m,j}$ is the number of audit records on the j^{th} day that indicate that the m^{th} activity type has occurred.

In NIDES, the Q statistic compares the short-term distribution of the types of audit records that have been generated with the long-term distribution of types. In the simplest situation, Q_n (the value of the Q statistic when the n^{th} audit record is processed) is defined as follows:

$$Q_n = \sum_{m=1}^{M} [(g_{m,n} - f_m)^2 / V_m]$$

where

$g_{m,n}$ = the relative frequency with which the m^{th} activity type has occurred in the recent past (which ends at the n^{th} audit record)
V_m = the approximate variance of the $g_{m,n}$

If we view $g_{m,n}$ as the short-term profile for the audit record distribution and we view f_m as the long-term profile for audit record distribution, then Qn measures the degree of dissimilarity between the short-term behavior and long term-behavior. That is, Q_n is larger whenever the distribution of activity types in the recent past differs substantially from the historical distribution of activity types, where "substantially" is measured in terms of the statistical variability introduced because the near past contains relatively small (effective) sample size. The value of $g_{m,n}$ is given by the formula

$$g_{m,n} = (1/N_r) \sum_{j=1}^{n} [I(j,m) 2^{-r(n-j)}]$$

or by the recursion formula

$$g_{m,n} = 2^{-r} g_{m,n-1} + [I(n,m)/N_r]$$

where

j = an index denoting audit record sequence
$I(j, m)$ = 1.0 if the j^{th} audit record indicates activity type m has occurred and 0.0 otherwise

r = the decay rate for Q that determines the half-life for the Q measure; we have set the half-life to approximately 100 audit records, corresponding to an r value of $-\log$ = (0.5)/l00 = 0.01.

N_r = the sample size for the Q statistic, which is given by the formula

$$N_r = \sum_{j=1}^{n} 2^{-r(n-j)}$$

and which rapidly approaches an asymptotic value of l/(1 - 2^{-r})

The value of V_m is given by the formula

$$V_m = f_m(1 - f_m)/N_r$$

except that V_m is not allowed to be smaller than $0.01/N_r$.

1.10 Computing the *Q* Statistic for Categorical Measures

Categorical measures are those that involve the names of particular resources being used (such as the names of files being accessed, or the location from which logons are attempted) or involve other categorical characteristics of audit records, such as the hour of the day on which the audit record was generated.

The method used for computing Q for categorical measures is essentially the same as that previously described for computing Q for the audit record distribution measure. In fact, we may view the audit record distribution measure as a categorical measure. The only difference in the definition of Q is that every audit record results in the recalculation of the audit record distribution measure, whereas for all other categorical measures, Q is only updated whenever the audit record contains information relevant to the particular measure. For example, the file name accessed measure would only be updated whenever the audit record concerns a file access. A half-life for the file name accessed measure of 100 audit records would refer to 100 audit records relevant to file names accessed, rather than to the last 100 audit records.

1.11 Computing the *Q* Statistic for Counting Measures

Counting measures are those that involve counts of particular resources used (such as CPU time in milliseconds or I/O counts) or some other numerical feature of audit records (such as the interarrival time in milliseconds of consecutive audit records).

Counting measures are transformed to categorical measures. For example, consider the counting measure of CPU time. Individual audit records arrive that indicate that a non-zero amount of CPU time has been incrementally expended since the last reporting of CPU usage. We might expect that this delta-CPU value would be between 0 and a maximum of 200 seconds. We define 32 geometrically scaled intervals between 0 and 200 milliseconds employing the same procedure as we used for defining intervals for the historical profile for Q (including the convention that the last interval actually extends to infinity). When a delta-CPU value arrives and is classified into interval m, we state that a categorical event m has occurred. Thus, we translate CPU time into a categorical measure where a category is activated whenever an audit record arrives with a delta-CPU value in that category. Thus, the Q measure for CPU time doesn't directly measure total CPU usage in the near past; rather, it measures whether the distribution of delta-CPU values in the near past is similar to the historical distribution of delta-CPU values. Once the counting measure is redefined as a categorical measure as discussed above, the Q statistic is calculated in the same fashion as for any other categorical measure.

1.12 Addition of a "Rare" Category

In previous versions of NIDES, each category of behavior in a particular measure was treated separately when comparing the long- and short-term profiles. For example, for the measure of "files accessed," each different file name in the long-term profile was treated as an individual category. Often, this resulted in hundreds of categories in the long-term profile, the vast majority of which had very small probabilities (much less than 1%).

The presence of so many categories makes it difficult to detect intruders. An intruder browsing the host's files would tend to touch a reasonable number of the files that the user doesn't use very often. Thus, the intruder might demonstrate a short-term profile with small but non-negligible probabilities (say on the order of 2%) for many of these files. On a file-by-file basis, a comparison of two small probabilities would not be statistically significant.

We decided that if we aggregated the rarest categories, then the detection of an intruder who frequently touched rare categories would be easier. We now create a temporary new "rare" category (at profile updating time) that contains all of the user's files with very small probabilities. Categories are temporarily added to the rare category until the addition of another category would cause the cumulative sum of the probabilities that have already been added to the rare category to first exceed a preset threshold for the sum of rare probabilities (for example, 1%). When the short-term profile is compared to the long-term profile, all of the files that are combined into the new rare category are treated as if they were the same file. Thus, an intruder might have a short-term profile in which 30% of the probability was in the rare category. The difference between the actual rare category probability sum in the long-term profile and 30% probability in the short-term profile will tend to be easier to detect than the differences between very small probabilities.

We note that the use of the rare category is not without trade-offs. For example, if the intruder continues to use a single file in the rare category, this will be more difficult to detect than previously. We also note that separate counts for all categories are maintained (whether or not a particular category has temporarily become part of the rare category for a particular day); this allows categories to migrate into and out of the rare category on different days.

1.13 Addition of a “New” Category

One type of behavior that we need to be particularly careful to identify is the creation of, or use of, new categories. The most obvious example would be the use of commands that the host user has never used before. (Because our profiles eventually discard categories with associated probabilities that decay below certain limits, it is more precise to say that these are categories that have not been used recently by the host.)

To increase our ability to detect new behavior within a measure, we have added a “new” category. The first time during a day when a short-term behavior is noticed for which there is no category in the long-term profile, the short-term probability of the new category is incremented. This is compared to the probability in the long-term profile for the new category to determine whether the amount of new behavior is statistically significant.

For example, suppose that the probability in the long-term profile for the new category is 0.1%. (This will almost always be a very small probability.) Further suppose that there are no categories in the long-term profile for commands W, X, Y, or Z. Suppose that the short-term profile is effectively based on 200 audit records, and that until now no new short-term behavior has been observed over, say, the last 1000 audit records.) Finally, suppose that 10 new audit records arrive, including two occurrences of X, three of Y, one of Z, and one of W. The short-term probability of the new category would be 4/200 = 2%[2]. In the calculation of Q we include a term for the "new category” with an observed percentage of 2% and an expected percentage of 0.1% (e.g., an observed count of about 4 and an expected count of about 0.2).

[2] The probability for the new category is calculated separately from the probabilities for all other categories. Thus, if there is a measure with many new categories (such as a file access measure) the probabilities for all other categories still sum to 100%.

Chapter 2

Rationale for the Current NIDES Statistical Component

The objective of describing the rationale for the NIDES statistical component is fourfold:

- To justify the statistical algorithms that SRI is currently using in NIDES
- To discuss how the NIDES approach relates to those used in two other intrusion-detection systems (Wisdom and Sense and Haystack)
- To discuss how the NIDES approach relates to four general statistical approaches (pattern recognition, discriminant and classification analysis, Marcovian transition analysis, and Bayesian decision analysis)

2.1 The Current NIDES Statistical Component

2.1.1 Statistical Component Philosophy

Understanding why SRI has chosen to implement certain types of algorithms in the NIDES statistical component requires an explanation of the philosophy underlying the statistical component. A critical element in the development of this philosophy was the knowledge that the NIDES statistical component would be used in conjunction with a rule-based component. It was assumed that the rule-based component would be defined to detect all known methods of compromising system security (or detect indicators that system security is or might be compromised). The statistical component is therefore devoted to uncovering all anomalous behaviors, just in case these anomalous behaviors might imply that system security is being compromised via some unknown or otherwise undetectable method. It may be the only way of detecting an insider attack, which may be composed of actions that in other circumstances would be considered acceptable or which exploit previously unknown system vulnerabilities.

The basic philosophy underlying the statistical component is that it is intended as the detection or protection system of last resort. It is intended to uncover those actions that cannot be prevented (e.g., through physical or software safeguards, such as passwords) or detected by a set of rules embedded in a rule-based component. Thus, whenever we (i.e., the intrusion-detection community) know that a particular set of actions either constitutes a violation of system security or is sufficiently serious to warrant an investigation or immediate preventive steps (such as disconnecting the user), the statistical component assumes that such set of actions is either prevented via physical or software safeguards or encoded as a set of rules in a rule-based component.

Because it is intended as a detection system of last resort, the statistical component cannot rely on our previous knowledge of security violations or methods of compromising security. It must assume that the user might be engaging in a totally new or previously unknown method of compromising security. It cannot therefore rely on the body of knowledge that we have previously gathered. All that previous knowledge is assumed to be embedded in the rule-based component. (However, testing the statistical component against known intrusion, masquerading, or malfeasance scenarios is valuable in that it provides us with some level of assurance that the statistical component is capable of detecting these types of anomalous behavior.)

Because it cannot rely on previous knowledge of methods to compromise security, the statistical component attempts to detect any anomalous behavior, whether or not that behavior has been previously associated with malfeasance. Thus, if the user engages in unusual behavior (where the unusualness of behavior is measured relative to that user's own prior behavior or relative to the behavior of a group of employees who are supposed to be using the computer system in the same way), the statistical component raises a warning flag. The warning flag does not mean that system security is being compromised — only that the user is behaving in an unusual fashion. For example, if a user suddenly starts accessing directories that he or she has not previously accessed, this anomaly would be detected even if the user has full and legitimate access to those directories. On the other hand, if a user *routinely* exploits a flaw in the operating system to change his or her access privileges to "root" and then back again, then this behavior would not be deemed anomalous even though this action violates system security.

2.1.2 Basic NIDES Statistical Approach

The basic statistical approach in NIDES is to compare a user's short-term behavior to the user's long-term behavior. Whenever short-term behavior is sufficiently unlike long-term behavior, a warning flag is raised. As discussed above, this statistical approach requires no *a priori* knowledge about what types of behaviors would lead to compromised security.

The number of audit records or number of days that constitute short-term and long-term behavior can be set by the security officer through the specification of a "half-life." The NIDES developers will provide "rules of thumb" for the specification of the half-life. For example, a security officer who wants short-term behavior to be on the order of 200 audit records should specify a half-life of approximately 100 audit records. This will assure that the 200th audit record has only one-quarter the influence of the most recent audit record, the 400th audit record has only one-sixteenth the influence, and so forth. Similarly, a reasonable half-life for a long-term profile might be 30 days.

In comparing short-term behavior to long-term behavior, the statistical component will be concerned about long-term behaviors that do not appear in short-term behavior as well as about short-term behaviors that are not typical of long-term behavior. For example, if a particular file is the object of 10% of file accesses in the long term, whereas in the current short-term behavior only 1% of file accesses involve that file, then this discrepancy will contribute towards a finding of abnormality. Similarly, if a particular file is the object of only 1% of file accesses in the long term, whereas in the current short term this file is the object of 10% of file accesses, then this discrepancy will contribute towards a finding of abnormality.

2.1.3 Time Horizon of Short-Term Behavior in NIDES

The NIDES statistical component keeps track of many different measures (or aspects) of behavior, such as commands used, files accessed, and remote hosts accessed. For each of these measures, it keeps track of short-term behavior separately. Therefore the half-life of short-term behavior for commands used refers to the last (say) 100 commands issued, whether these commands were executed in the last minute, hour, day, or week. The half-life of short-term behavior for files accessed refers to the last (say) 100 files accessed, whether these files were accessed in the last minute, hour, day, or week. Thus, the chronological time frames for the different measures can be different. In addition, it is not theoretically necessary that short term behavior for each measure refer to the same half-life (although the current implementation of NIDES does not allow easy specification of measure-specific half-lives). For example, the half-life of short-term behavior with respect to remote host logins could be 20 logins, whereas the half-life of short-term behavior with respect to file accesses might refer to the last 500 file accesses. We hope in the future to provide automated support to setting half-lives, so that the number of audit records that constitute the half-life of short-term behavior for each measure is large enough to ensure stability in the measure (making it easier to achieve a low false positive warning rate) and small enough that the measure will react rapidly to an intrusion.

2.2 Statistical Approaches in Other Intrusion-Detection Systems

2.2.1 Sequences of Events in NIDES and Wisdom and Sense

The concept of a sequence of events is central to the Wisdom and Sense (W&S) system [7]. To set the stage for discussing how (W&S) handles sequences of events, we first discuss how both the statistical and rule-based components of NIDES handle events.

The NIDES statistical component can be considered as a tool for examining whether the "events" in short-term behavior are similar to the "events" in long-term behavior. Currently we define an "event" as a rather low-level concept, such as the execution of a particular command, logging on from a particular terminal, or accessing a particular file. A common feature of these events is that their occurrence can be determined from individual audit records. However, the statistical algorithms could be modified to handle higher-level events requiring information from multiple audit records. For example, we could define a compound event such as "logs onto a remote host as guest and then changes user name." Later, we discuss how higher-level events could be used to improve the detection capability of the NIDES statistical component. For now, we note that if higher-level events are added to the NIDES statistical component, these should not be restricted to those relatively few events that are recognized as being precursors or indicators of an attempt to compromise system security. Instead, the enumeration of suspicious events should be made in the rule- based component. As soon as an identified suspicious event is detected, the rule-based component can alert the security officer or take other preventive or monitoring actions. Again, the purpose of the statistical component is to detect a change in behavior rather than to identify the occurrence of a specific behavior.

The NIDES statistical component decides whether current behavior is anomalous based on the entire group of events that have occurred in a rapidly aged past (generally, the past few hundred audit records). Thus, the decision is based on a group of events, rather than any single event. NIDES raises an alarm when the most recent event pushes the short-term profile "over the limit" in the sense that the short-term profile becomes so different from the long term profile that the statistical component considers the short-term profile to be anomalous. Although the NIDES statistical component bases its decision on a group of events, it is not particularly influenced by the order in which those events occur (except that more recently observed events exert a greater influence on profile scoring than less recent events). Strictly speaking, therefore, the NIDES statistical component cannot be said to monitor ordered events.

The NIDES rule-based component does handle event sequences. In most cases with which we are familiar, the interest in sequences of events concerns very specific sequences that are associated with compromises of system security. The rule-based component can monitor the occurrence of different events, and when a sufficient portion of the sequence has occurred, the rule-based component can notify the security officer that it is likely that the user in question is attempting to compromise system security. In accordance with our general design philosophy, if a sequence has been specifically identified because it has been associated with compromised security, it belongs in the rule-based component. If, on the other hand, we are examining sequences so that we can better understand normal behavior, and therefore contrast short-term behavior (i.e., recent behavior) with long-term behavior (i.e., historical behavior), the proper place to examine the sequence would be in the statistical component.

Many rule-based systems (including the NIDES rule-based component) monitor sequences of events. The only statistical intrusion-detection system with which we are familiar that uses sequences is Wisdom and Sense, and the types of sequences considered by W&S are fairly elementary. The W&S system builds its own sequences and must consequently deal with a huge number of possible sequences. It does so by building a massive tree structure (the branches of which are the sequences) and then pruning the tree so that it contains only the most frequently used sequences. However, to reduce the overhead required to build the tree structures, the W&S system usually does not build tree structures for each individual user (instead, building a single tree for all users) and the tree structure involves data from only the past few days. Recent behavior of the user is compared to this tree structure in the following sense — whenever the user has executed a sequence of n events, the first $n - 1$ of which are in the tree, the *nth* event is evaluated to determine whether or not it is also in the tree. If it is in the tree, it typically nudges W&S toward concluding that the user is behaving normally (particularly if the last node representing that event has a high conditional probability of occurrence given the first $n - 1$ nodes). If it is not in the tree (or if the conditional probability of the last node is low), the last event nudges W&S towards concluding that the user is behaving abnormally. Typically, sequences in W&S are less than four to six events long. In addition, typical events are fairly elementary in nature. For example, an event might be a user name, an action, an object for an action, or a time of occurrence for an action. Thus, the mega-event that user A logged in from terminal A at 8:20 a.m. would constitute a four-event sequence represented in a four-node tree structure.

Owing to its complexity and the relative lack of documentation concerning W&S, we do not know whether the W&S system of considering sequences of elementary events and then evaluating conditional probabilities for the last node of those sequences is more or less powerful than the NIDES statistical component. We have not chosen to implement the W&S approach, primarily because the W&S system tends to profile a large group of users over very few days, whereas we wanted to base our statistical comparisons on individual profiles of longer-term behavior. However, the W&S approach does have merit, and its approach to handling sequences could presumably be incorporated into NIDES by building hooks into W&S's code.

Because of the potential value of sequences of events, we are also interested in determining how sequences could be incorporated into the NIDES statistical component. The easiest way of doing this would be to define "mega-events.”For example, the higher-level or compound event considered earlier (i.e., log onto remote host as guest and then change user name) is a simple sequence of events. Mega-events could be incorporated into the NIDES statistical component without a change in the current methodology. However, the definition of those events, and the development of software to recognize those events, would be a substantial research task.

Another method for embedding sequences into the statistical component would be to rewrite the code so that the statistical component builds its own sequences. For example, the statistical component could identify and track all three-event sequences and monitor their likelihood of occurrence. The difficulty with this approach is that the number of different sequences being monitored could become excessively large. For example, even if there were only 20 basic events, the number of possible sequences of three events would be 20 × 20 × 20 = 8000. Unless the users' behavior were concentrated on a very small portion of these events, or the events could be collapsed into a smaller number, the sequences would not be useful for examining abnormality (because each individual sequence would have an extremely small probability of normal occurrence). Methods for collapsing (or clustering) behaviors into a reasonable number of mega-events or sequences would be necessary. We believe that it might be possible to develop a reasonable collapsing strategy, and that the resultant methodology would be more powerful than the one used in W&S, because it would not require the asymmetry inherent in the W&S approach (wherein most measures serve as “conditions”and only one measure at a time can act as the "outcome”) and would more fully exploit the “multidimensionality” of the data.

2.2.2 Haystack's Statistical, Approach

In many respects, the Haystack statistical approach appears to be similar to the NIDES statistical approach, although there are some quite distinct differences, and we discuss both these similarities and differences here. We caution that we have not been able to obtain complete documentation concerning Haystack [ll], and therefore our conclusions regarding this system are preliminary. Steve Smaha, the Haystack system developer, also recommended that we note that the Haystack system is a number of years old.

In Haystack, as in NIDES, individual measures (called features) are monitored and compared to historical behavior to determine their abnormality. However, in Haystack this assessment appears to take place at the conclusion of the session, rather than in real time or across sessions. In addition, it appears that only “counting”measures (such as amount of I/O, amount of CPU, or number of files) are supported. (NIDES also supports “categorical” measures such as name of terminal, names of files accessed, names of directories changed, and names of commands executed.) For each measure, the Haystack system determines a range of values that are considered “normal.”For example, if the measure is number of files accessed, the normal behavior for a user consists of a range (such as 0 to 34 files) containing at least 90% of all observed values for number of files accessed in the sessions belonging to the particular security group to which the user belongs.

In contrast to NIDES, the Haystack statistical system contains a set of six generic types of computer abuse (attempted break-ins by unauthorized users, masquerade attack, penetration of the security control system, leakage, denial of service, and malicious use). For each type of computer abuse there is a set of weights (from 0 to 9) indicating the extent to which each measure is related to that type of computer abuse. For example, for denial of service, the measure “number of pages printed” might be given a weight of 9, whereas a measure "password errors”might be given a lesser weight. When Haystack analyzes a session, each session feature outside of the predefined range causes a weight for that feature to be added to the session's score for that computer abuse. Each of the features is assumed to be independent of the other features. Under this assumption, it is possible to calculate the probability distribution of the computer abuse score (called a suspicion quotient) and alert the security officer when that score is too large.

Haystack also contains a simple “trend” test to determine whether the counting features are trending to larger or smaller values. This test (the Mann-Whitney procedure) is applied to contrast the values of the feature in the most recent 5 sessions with the values of the feature in the most recent 20 sessions. Basically, this test looks for changes in the median values. This test would not be applicable to categorical measures, nor to more complicated types of changes in counting distributions (for example, a trend towards both larger and smaller values but not a change in the median value).

Although we have never conducted any tests with the Haystack system, we believe that the NIDES methodology for handling individual measures is preferable to the Haystack methodology. For counting measures the difference in effectiveness is probably marginal, except when the measures are bimodal (such as would occur if a user engaged in two primary types of activity, one with low CPU usage per session and one with high CPU usage per session), in which case the SRI approach should yield better results. However, we believe that the most powerful measures will be the categorical measures, which the Haystack system does not support. In addition, the NIDES system supports nearly real time intrusion detection, whereas the Haystack system supports only retrospective analysis of entire sessions.

The Haystack system combines functions of the NIDES statistical component (i.e., identifying abnormal behavior) with functions of the planned NIDES resolver (i.e., drawing conclusions as to whether a particular behavior is suggestive of a type of computer abuse). Rather than combine these two distinct functions, we have chosen to separate them, believing that by doing so each will be more efficacious. For example, the planned resolver functions, to include model-based definitions of suspicious behaviors, could potentially be far more powerful than Haystack's relatively simple approach.

The degree of success of Haystack's approach for drawing conclusions depends on the validity of the weights used to obtain scores for computer abuse types. Unfortunately, we have not been able to obtain those weights (or even a complete listing of the measures used). We would be interested in examining the Haystack weighting system to assess its sensitivity in detecting simulated intrusions. (In addition, we note that in the absence of further development of model-based reasoning in the resolver, the Haystack weighting system could be implemented fairly readily within the NIDES resolver, while maintaining real-time intrusion detection and the use of categorical measures.)

2.3 Alternative General Statistical Approaches

2.3.1 Pattern Recognition

Pattern recognition [2] is a fairly well defined set of procedures when applied to the problem of identifying physical objects in two- or three-dimensional visual representations. Thus, when we discuss "pattern recognition" with respect to the problem of having a robot navigate a room filled with objects or the problem of identifying objects in a high-altitude photograph, it is fairly clear what set of procedures we are considering.

In other realms of statistics, pattern recognition does not refer to a well-defined set of procedures. A very wide range of statistical procedures has been used, in some sense or another, to find "patterns" in the data. As an extreme example, even regression analysis (which is not usually considered to be a form of pattern recognition) bears many characteristics of pattern recognition, since it is an optimal procedure for determining how well the patterns of values in the independent variables match (or fit) the pattern of values in the dependent variable. Thus, one of the main difficulties in asking "Why didn't NIDES use pattern recognition as its statistical approach" is that so many different statistical procedures qualify as pattern recognition approaches.

In fact, we could easily consider the NIDES statistical approach to be one of pattern recognition. The NIDES statistical component builds a long-term profile (or pattern) of user behavior and then compares this pattern to the short-term profile (or pattern) of user behavior. Thus, the NIDES statistical component is not only constructing patterns, but is also comparing patterns.

Probably, the statistical procedures that are most closely associated with pattern recognition are cluster analysis [l] and factor analysis [8]. Both of these procedures are essentially exploratory in nature. We use cluster analysis to find clusters in multivariate space (typically higher than three dimensions) and factor analysis to reduce the dimensionality (i.e., number) of outcome variables.

With respect to factor analysis, it is difficult to conceive how such an approach would be applied to intrusion detection. Presumably, each event would be characterized according to measures such as the amount of CPU used, the number of files accessed, and the time of day of the activity) and then this data would be processed to find fewer "factors" that explain the correlation among these measures. However, we do not believe that the results of such a factor analysis would be interpretable, and would probably only confuse the security officer.

It might be possible to use cluster analysis for anomaly detection. In this case, we would characterize events according to specific measures (such as amount of CPU used, the numbers of files accessed, whether the user accessed some other user's directory, and the time of day of the activity). We would then locate events gathered over a fairly long period of time (say weeks or months) in n-space (where *n* is the number of measures used in the event characterization) and form clusters. A probability would be assigned to each cluster corresponding to the number of events in that cluster (relative to the total number of events). The events in short-term behavior would also be identified, where we might define short-term behavior as the last 20 events. A running score would be computed for each user. That score would increase if the last event engaged in by the user did not belong to any previously identified cluster or was identified with a cluster with low probability of occurrence. Similarly, that score would decrease if the last event engaged in by the user belonged to a previously identified cluster with a high probability of occurrence. When the score became sufficiently large, the security officer would be notified that the user was engaging in anomalous behavior.

The use of cluster analysis in the way described above would be compatible with the current NIDES statistical approach. In essence, we currently track the probability of each event by itself and increase or decrease the anomaly score function on the basis of that individual probability. The cluster analysis approach would allow us to cluster events into larger collections of events and then assign probabilities to that cluster (which would be the sum of the probabilities of the individual events within that cluster).

The difficulty in applying cluster analysis is that it is more an art form than a set of definitive procedures. We are aware of and have used a wide variety of cluster analysis procedures on other projects. Unfortunately, a large amount of a statistician's time and attention is necessary to make the clusters meaningful. For example, the definition of the distance metric plays a critical role in determining which events would cluster together and in determining whether we get clusters that look more like "balls" or "strings" of events. We believe that it would be a fairly substantial research effort to identify reasonable metrics and clustering procedures. Worse, the clustering would have to be done repeatedly (say every few nights) on hundreds or thousands of individual users, because it is doubtful that clusters would be generalizable across systems or user communities. This would require storage of immense amounts of data (for example, the last 10,000 or so events for each user of the system) and might involve exorbitantly long processing times since cluster analyses are time-consuming. Unfortunately, the cluster analysis would probably frequently fail. Cluster analysis procedures tend to be very sensitive to the identification of the proper "seeds" for the clusters, and often it takes a trained analyst to identify which seeds to use, based on the results of an unsuccessful clustering attempt. Thus, although cluster analysis is a very useful exploratory tool we do not believe that it would be appropriate for use in an anomaly detection system.

2.3.2 Discriminant or Classification Analysis

Classification analysis relies on previous examples of behavior to develop a set of rules upon which current behavior can be classified. For example, if we had historical examples of "normal" and "abnormal" behavior for a user, we could use those examples to classify new behavior as either normal or abnormal. Two commonly used procedures are discriminant analysis and classification tree analysis [6].

In our experience, classification analysis is an extremely efficient and powerful methodology for classifying current behavior as normal or abnormal. The principal problem in using this analysis procedure is in finding the examples of abnormal behavior. We simply do not have a sufficient number of examples of behavior that compromise system security. Nor do we trust the few examples that we have to be exhaustive, in the sense that they cover all types of behavior that could compromise system security. Even if we had exhaustive examples of known behavior that compromised system security, we would have to be concerned about the extent to which unknown or new behavior that compromised system security would resemble previously identified behavior that compromised system security.

Because actual examples of behavior that compromises system security are difficult to come by, analysts sometimes use the normal behavior of other users to characterize abnormal behavior for the host user. The first prototype for IDES [4] used both discriminant analysis and classification tree analysis in this way, and we showed that it was relatively easy to distinguish between the normal behavior of different users. We did not choose to continue the development of a classification approach because we had doubts concerning the extent to which behavior intended to compromise system security would resemble the normal behavior of other users. However, this approach may still have some value in identifying masqueraders, and we have performed some preliminary evaluations showing that there may be merit in adding a discriminant analysis into NIDES.

It is possible to heuristically relate the current NIDES statistical approach to a classification approach. Basically, in the current NIDES statistical approach we calculate the absolute probability of each type of event (as determined by examining the historical behavior of the host user) and then score recent behavior on the basis of that absolute probability. Thus, if the absolute probability of an event (as measured by the past history of events for that user) is small (say on the order of 0.01%), and the user has recently engaged in that event, we nudge the anomaly score upwards towards a conclusion of "anomaly." On the other hand, in classification analysis we calculate the relative probability for each type of event, and then score recent behavior on the basis of that relative probability. For example, if the user engages only in a particular type of event with a probability of 0.0l%, but the comparison population (i.e., all other users or intruders) engages in that same type of event with a probability of 0.000l%, then the occurrence of that event is 100 times more likely for the host than for the comparison population. Therefore, if that event occurs in recent behavior it is evidence in favor of normal behavior rather than abnormal behavior. Thus, it is possible (although not highly likely) that classification analysis and the current NIDES analysis procedures could lead to opposite conclusions. The critical role played by relative probabilities means that the definition of the comparison population and the inclusiveness and representativeness of the examples of the behavior of the comparison population are critical to the success of classification analysis.

2.3.3 Markovian Transition Analysis

In Markovian transition analysis [5], we are typically concerned with the definition of different states and the probabilities of transitions from one state to another.

In intrusion analysis, Markovian transition analysis might be used to anticipate to which state a user would move. This would be important if the states were associated with compromised security and it was necessary to take preventive action prior to a particular state's being reached. However, in this event it is necessary to calculate this probability only once. Thereafter, a rule could be entered into a rule-based component that simply states that if state A occurs the probability that the next state will be associated with compromised security is higher than the allowable threshold, and therefore preventive action should occur.

Another way to use Markovian transition analysis would be to characterize the normal behavior of a user on the basis of which states he or she moves between and with what probabilities. This approach would be absolutely consistent with the current NIDES approach, where an “event” would be the transition between two particular states. NIDES would track the probabilities of these events (i.e., the transition probabilities between states) and would score a user on the basis of whether or not the transitions were typical for the host user. The only additional effort needed to implement this approach is that involved in defining the states that we want to track movement between, and writing the software to detect, that a state has been achieved. This is not an easy exercise; we note that for N states, there are N^2 transition probabilities, and therefore the transition matrix approach can quickly become unmanageable.

2.3.4 Bayesian Decision Analysis

The NIDES statistical team has recently implemented one of the most complicated and state-of-the-art Bayesian decision analysis systems in use today, and is therefore familiar with the advanced theory of Bayesian decision analysis as well as the implementation issues involved. [10] We believe that this analysis procedure (or a similar procedure such as evidential reasoning) should play an important role in NIDES. However, we believe that the proper role is in the resolver, which will be able to accept information from both the statistical and rule-based components and then draw conclusions with respect to the likelihood that system security is being compromised and the seriousness and type of the potential breach.

To some extent, a Bayesian decision analysis can be considered the next step beyond the traditional rule-based system. In the traditional rule based system, rules either fire or they do not. That is, each state represented by the rule has a probability of either 0 or 1. There have been a number of attempts to integrate probabilities into rule-based systems, so that we can infer the probability that a state is true, rather than definitely judging it true or false. Most of these systems work to some extent, but sooner or later run afoul of the rules of probability theory. (For example, these systems rarely handle contradictory information very well.) Bayesian decision analysis systems can be considered to be rule based systems that do obey the rules of probability theory (although the rules are so cleverly disguised in the relationships among nodes in the probability network that it is often difficult to realize that there are even rules present).

To be truly successful, a Bayesian decision analysis system must incorporate a substantial amount of knowledge concerning the different types of attacks that can be used to compromise system security as well as the conditional probabilities that various well-defined events will occur given that those attacks are in progress. We believe that the intrusion-detection community is only at the first stages of trying to assemble this type of knowledge.

Because the Bayesian decision analysis system approach relies very heavily on our current knowledge of methods for compromising security, we do not believe that it is a substitute for the current NIDES statistical component, which attempts to identify any behavior that is not consistent with past behavior.

Chapter 3

Evaluation of Statistical Approaches for Appropriateness to Intrusion Detection

In the process of developing the NIDES statistical component, and in reading the literature on intrusion detection, we had the occasion to consider a number of different statistical approaches that might have been used. Although we did not develop a set of formal evaluation criteria, it quickly became evident to us that most statistical approaches were not suitable for intrusion detection. For example, the method that we developed for NIDES, while it bears some similarity to chi-square analysis, has undergone substantial modification and is much different from that approach as found in the statistical literature.

Here, we present a set of evaluation criteria that should be considered when determining how likely it is that a proposed statistical approach will work in a real-time intrusion-detection environment.

3.1 Criterion 1. Does the Method Depend Upon Distributional Assumptions?

Many statistical approaches require that the data fit a normal or other distribution (such as gamma, chi-square, beta). Unfortunately, these assumptions are rarely true. This can render the statistical approach unsuitable, unless one of two circumstances occurs:

- The approach transforms the data in such a way as to force these distributions to be true. For example, the average of a large number of observations would probably be approximately normally distributed even if the individual observations were not. Another example is in NIDES, where we force the measures to follow what is essentially a chi-square distribution by mapping the empirical cumulative distribution function of the measure into a "half-normal" distribution and then squaring this variable.

- The statistical approach is robust to deviations from the distributional assumption. For example, nonparametric approaches have few or no distribution assumptions.

We note that it is typically not sufficient that the data for which a distribution is assumed pass a goodness-of-fit test (which typically verifies only that the bulk of the probability distribution has the required shape), because we are typically interested in the extreme tails of the distribution of the test statistic.

The NIDES statistical component was developed so that it did not depend on distributional assumptions. We felt that this was necessary because the system was intended to be applied across a wide variety of computer systems, types of users, and types of measures. Our reluctance to assume any distribution constrains the types of statistical procedures that could be used. For example, if the measures could be assumed to follow a true multivariate normal distribution, then there are optimal statistical procedures that could have been applied.

3.2 Criterion 2. Does the Method Assume Independence?

Data from audit streams are notoriously interdependent. For example, once a file has been opened and read, it is highly likely that the following audit records will contain subsequent reads on that file. Furthermore, the correlation that we tend to encounter is not simple serial correlation. For example, if two files are open, then accesses to these files are likely to alternate over a time span of audit data (as would be the case if one file was open for reading and another for writing).

For a statistical approach to be suitable for intrusion detection it must not depend upon the independence of the audit trail records. This is a very strong condition and is one of the primary reasons why statistical approaches fail to be appropriate. The NIDES statistical component has a feature that accommodates the lack of independence of successive audit records without requiring that the type of dependence be modeled. This feature is essentially the empirical observation of the distribution of the evaluation statistic obtained by observing its values over time, rather than postulating its distribution based on the observation of the distribution of its component parts and assuming the independence of those parts.

3.3 Criterion 3. Does the Method Accommodate Categorical Data?

In our experience, the most valuable data in the audit trail are the categorical data, such as names of directories accessed, names of files accessed, commands used, and terminals used. Furthermore, there are sometimes hundreds (or even thousands) of categories for a single measure. Unfortunately, many statistical approaches assume the existence of continuous data and are not capable of handling categorical data well. Such approaches are less likely to extract sufficient information out of the data stream to be useful (i.e., to have a high probability of detecting misuse or an intrusion). Much of the recent work in NIDES has gone into making our approach very friendly to categorical data (in fact, we even transform continuous measures into categorical measures).

3.4 Criterion 4. Does the Method Allow Real-Time Evaluation of Audit Records?

During a typical day, tens or hundreds of thousands of audit records may be generated, and these records will tend to be heavily concentrated in about 10 business hours. If the objective is real-time audit trail assessment, then the statistical component will similarly have to implement tens or hundreds of thousands of assessments each day of whether an intrusion or misuse has occurred. This imposes considerable constraints on the statistical procedure - it must be sufficiently efficient to execute an assessment in a fraction of a second. (NIDES is currently capable of processing approximately 30 to 45 audit records per second.) Although many statistical procedures are capable of this (e.g., discriminant analysis, neural networks), other procedures may not be capable. In particular, there may be a problem if the proposed statistical approach is using a sliding window of data (say the last 100 audit records), and whenever a new audit record is added and one subtracted from the historical queue, the entire statistic needs to be recomputed, rather than merely being given a simple recursive adjustment. Many of the refinements to the NIDES statistical component were implemented in order to speed evaluation, including the use of exponentially weighted "sufficient" statistics.

3.5 Criterion 5. Does the Method Allow for Profiles for Individual Users?

Individual profiles may vary substantially from person to person. In order to achieve the maximum discriminatory power, we believe that in general, it is necessary to develop individual profiles. In developing NIDES, we set as a requirement that each user have his or her own profile. This requirement affects both the time necessary to update profiles (since there are many profiles to update) and memory requirements. Profiles must be capable of being stored in a reasonable amount of space, or an excessive amount of time will be spent swapping profiles in and out of memory. Currently, NIDES requires about 100kb per profile in the Sun-Unix environment, which has a rich set of measures available. We suspect that the NIDES profiles could be stored in less space when using audit record data from other computer systems.

In some environments job classifications may be tightly prescribed, and individuals in the same job classification may use the computer in essentially the same way. In this case it may be possible to develop useful group profiles, and individual profiles may not be necessary. A statistical approach used in such an environment would not need to allow for individual user profiles, because group profiles would be sufficient. However, because NIDES was developed as a general-purpose intrusion-detection system, we developed a statistical component that could efficiently process individual profiles.

3.6 Criterion 6. Does the Method Allow for Profiles to Periodically Update without Human Intervention?

User profiles are constantly changing as the legitimate work of the user changes. Therefore, it is necessary that the statistical procedure be capable of updating user profiles at least once daily. Because there may be thousands of user profiles, efficiency in updating is important.

If we assume that updating should occur once per day, and that during the 10 business hours of the day the statistical system will be too busy processing audit records to update profiles, then the profiles will need to be updated during the remaining 14 hours (i.e., 50,000 seconds) of the day. If there are 2000 user profiles (i.e., a decently large system), each profile will need to be updated within 25 seconds. (With fewer users the profile updating time can take longer, and with more users the profile updating time will need to be less.) If we put inà safety margin of 50%, then profiles for a 2000 user system should each update within 12.5 seconds. While most statistical algorithms are capable of updating in this time frame, some may require more than this amount of time. The NIDES statistical component can update all profiles for an individual user in a second or so, and consequently if we initiate updating at midnight, we are typically finished within tens of minutes later.

It is also important that the updating procedure not require the guidance of a trained statistician, but rather be completely automated. Some statistical procedures, such as cluster analysis, typically require the intervention of a statistician to obtain decent results. A statistician will not typically be available when updating is required, and even if he or she were available, human intervention would slow the updating process unacceptably. The NIDES statistical component does not require human intervention to update profiles.

3.7 Criterion 7. Does the Method Allow for Multivariate Statistical Inference?

It is our belief that much of the ability to detect interactions rests in the interaction of different aspects of behavior. For example, a set of particular commands may be associated with a particular host computer or with action on a particular set of files. The statistical approach should therefore be able to extract additional information about the normality of different actions taking place essentially simultaneously (e.g., invoking commands and accessing files).

The use of highly dimensional multivariate data involves many difficulties that affect the ability of the statistical approach to satisfy other criteria. For example, the multivariate cross product of only three measures, with only 100 categories each, results in a matrix of 1,000,000 possibilities. Even though the matrix would probably be sparsely populated, storing and processing such a matrix would tend to make the statistical algorithm too slow and the storage requirements too large. In addition, development of stable probabilities for the matrix cells would require long training periods — perhaps sufficiently long that behavior would change before training was completed.

The NIDES approach makes limited use of multivariate data, and we consider this to be one of the statistical component's primary limitations. However, we have developed the theoretical approach necessary to incorporate multivariate data (at least to some extent), and could incorporate this approach in future versions of NIDES.

3.8 Criterion 8. Roes the Method Require the Existence of Defined Jobs or Sessions?

Computing used to be defined in terms of batch jobs. Now most processing takes place in continuous sessions, which may last for many hours. (In the extreme case, the user may never log off.) The statistical approach should not require that there be "jobs" or "sessions" or their equivalents, but rather should make assessments about data from a continuous stream of audit records. The lack of a natural time unit of analysis (that is, job or session) is directly responsible for the use in NIDES of exponentially weighted statistics.

3.9 Criterion 9. Does the Method Require the Existence of Simulated or Actual Intrusions?

Many statistical classification methods such as discriminant analysis or some neural networks require that training sets exist and the audit data in the training set be preclassified. For example, one portion of the training set might consist of audit data from users who are known not to be misusing the computer system and who own accounts that have not been subjected to intrusions. The other portion of the training set might consist of audit data gathered during simulated or actual misuse or intrusions. By comparing the "normal" and "intrusive" audit records in the training set, the statistical method can determine what features in the audit data best classify the data as normal or intrusive.

Unfortunately, data sets of simulated or actual intrusions are not widely available. (If such data sets had been available, we would probably have included a statistical classification method in NIDES.) A researcher who proposes to use a classification approach that requires the existence of simulated or actual intrusions should have a good idea where such data would come from or how to generate it. In addition, the ability of the statistical method to detect intrusions will generally be limited by the type of intrusions that it is trained on. That is, it will be able to detect new types of intrusions only to the extent to which they share features with the types of intrusions in the training set. Therefore, the generality and completeness of the simulated or actual intrusions in the training set should be assessed.

3.10 Criterion 10. Does the Method Develop Its Assessment Based on Differences between Users?

Because training sets of simulated or actual intrusions are not typically available, the researcher might propose to apply the statistical classification method to a training set consisting only of normal users. For example, to develop an algorithm for identifying abnormalities in User A, one portion of the training set might consist of user A's audit data and the other portion consist of a mixture of audit data from other users (which might be identified either as "non-A" or by their individual user ID S). The statistical method would then be trained to classify a new audit record either as A or as non-A.

Statistical approaches that distinguish normal users from one another might also be capable of identifying misuse or intrusive activity in an audit stream. We believe that such methods are worth investigating. However, it is not clear that such methods would actually be able to detect unauthorized activity. For example, for there to be a high probability that unauthorized activity in User A's account would be identified as non-A, that unauthorized activity must much more closely resemble the typical activity of other users than the activity of user A. In the event that the intrusive activity does not closely match either user A's activity or the activity of other users, statistical approaches such as discriminant analysis might fail to classify the activity as not belonging to A.

The NIDES methodology continuously compares a user's recent activity with his or her long-term profile. It is in this sense self-referential and independent of the activity of other users on the system. As such, it makes no assumption as to the nature of the difference between legitimate and unauthorized activity.

3.11 Criterion 11. Does the Method Require Assumptions about the Distribution of Intrusive Behavior?

Because data on actual and simulated intrusions are scarce, and a number of statistical procedures have been proven to have optimality properties when comparing two known distributions, it is tempting to make assumptions concerning the distribution of intrusive behavior. These assumptions are typically of one of two types. The first is an assumption that intrusive behavior always tends towards larger (or sometimes smaller) values of a particular measure of resource consumption (e.g., CPU time, I/O, file accesses, "finger" commands). The second is an assumption that intrusive behavior is uniformly distributed across all possible behaviors.

Although these assumptions allow "optimal" tests to be derived, unless the correctness of these assumptions is verified, the optimality of these tests must be severely discounted. In fact, most "reasonable" statistical tests can be shown to be optimal or near optimal versus some assumed distribution of intrusive behavior. This can be done by finding the set of behaviors that would result in the declaration of an anomaly and then assuming that intruder behavior is concentrated on those outcomes. (Unfortunately, the math involved in making such assessments is quite complicated.)

The NIDES statistical algorithms were not postulated with a specific distribution of intrusive behavior in mind, nor have we reverse-engineered the set of intrusive behaviors for which our statistic is optimal. However, the form of our test statistic (e.g., similar to a chi-square statistic) suggests that we spread our detection power over many possible intrusive behaviors and so would do particularly well versus a flat or uniform distribution for intrusive behavior.

3.12 Criterion 12. Does the Method Provide the Security Officer with Adequate Information to Conduct an Inquiry?

By their very nature, statistical approaches to anomaly detection will raise false positive alerts. That is, users do change their behavior over time — and sometimes abruptly so. Statistical algorithms will detect these changes and issue alerts even though no misuse activity is actually transpiring. Because the vast majority (i.e., 99% or more) of activity on a typical computer system will be legitimate, the number of false positive alerts will tend to outnumber the true positive alerts. A good deal of the system security officer's time will therefore be devoted to investigating false positive alerts. The statistical procedure must be capable of providing the security officer with information that can be used to identify the reason why the alert was issued. This will tend to greatly reduce the time involved in the security officer's determination of whether misuse has actually occurred.

The requirement for "understandability" may restrict the type of statistical procedures that can be applied. For example, factor analytic procedures tend to transform the space of measures in complicated ways, and it can become quite difficult to unscramble the statistical results in such a way that they can be explained to the security officer.

Because of the need for understandability, the NIDES statistical component uses the concept of long- and short-term profiles, which are probability distributions for categorical measures. Although the presentation format for these probability distributions still requires improvement, the concept of frequency distributions is easily grasped. For example, the security officer can see how frequently different commands have been issued in the past and which commands have been more recently issued. It is relatively easy then to identify those commands that are now being issued, but have not been issued in the past (or were issued much less frequently in the past) as well as those commands that were frequently issued in the past but are now conspicuously absent. The use of relatively simple individual measures (e.g., commands used, file names, hosts used, number of access denials) makes the results more comprehensible.

Chapter 4

Answers to Specific Questions about Features of the NIDES Statistical Algorithm

The NIDES statistical approach has evolved over many years. This evolution has been motivated by a mixture of theoretical considerations, practical considerations (dealing principally with issues of computing speed), and practical experience in exercising the algorithm. Here, we identify some of the major decisions that shaped the statistical approach and describe some of the specific features of the algorithm. We have adopted a question-and-answer format, where the questions deal with different features of the NIDES statistical approach, and the answers offer insight into why those features were incorporated. Occasionally, we group a set of questions that have closely related answers. We have also formulated a number of the questions (particularly the questions that appear early in this section) so that they illuminate the relationship between NIDES and the approach of Helman et al.[9]

4.1 Q: Is the NIDES Statistical Approach Based on an Assumed Intruder Behavior?

A decision made early in the NIDES development process is that we should not assume any knowledge of how intruders would compromise system security (or how a user would alter his or her behavior to compromise system security). All knowledge of this nature was to be incorporated into the expert system. We made this decision for four principal reasons:

- The expert system is typically more accurate in detecting known intrusion approaches.

- There were insufficient data on intrusion attempts to train a statistical approach.
- The typical (normal) usage of other users was not believed to be a reasonable surrogate for intrusion attempts to train a statistical approach.
- Future intrusions need not follow historical methods.

This decision was probably the single most important and influential decision in shaping the statistical approach. If we had been willing to model intrusive behavior, then we could have applied the Neyman-Pearson Lemma (and the decades of statistical research that followed from this lemma) to develop an optimal statistical approach. The Neyman-Pearson Lemma (proven in the 1930s by two of the founders of modern statistical theory) states that if there are two known probability distributions, then the optimal statistical test for distinguishing between these two distributions is based on the ratio of their values. For example, let x denote an outcome (derived from audit records), let $n(x)$ denote the probability of that outcome for the host user, and let $m(x)$ denote the probability of that outcome for the intruder. Suppose that we want to determine the set of outcomes X such that X occurs very rarely for the host user (say no more than 0.1% of the time) but much more frequently for the intruder. The optimal selection for X is the set of values of x with the highest values for the ratio $m(x)/n(x)$. That is, we order all values for x according to the ratio of $m(x)/n(x)$ and add x values into X until $n(X)$ equals 0.1%. A restatement of this lemma in a Bayesian framework can be found in Helman et al. (1992).

If we had chosen to model intrusive behavior, then by access to the Neyman-Pearson Lemma we could have developed an optimal approach for detecting intrusive behavior. This would have allowed use of many standard statistical approaches (such as discriminant analysis, Hotelling's T^2 tests) that stemmed (either directly or indirectly) from this lemma. Instead, we were faced with the task of defining a region X without reference to assumed intruder behavior, but rather based on the fact that such behavior was unusual for the user (whether or not it would be unusual for the intruder).

4.2 Q: What Does the Space *X* of Outcomes Look Like? How Has the Unusual Aspects of this Space Influenced the NIDES Statistical Component?

Statistical intrusion approaches accept audit record inputs that define the outcomes of subject behavior. These outcomes define a space, which we have denoted as X. A member x belonging to X is the particular behavior that has been exhibited by a subject. We want to decide whether x is representative of the normal behavior of the user or whether x is anomalous (and therefore indicative of misuse). For example, Helman et al. (1992) define two stationary stochastic processes on the outcome space — *N(x),* which denotes normal or legitimate transactions, and M(x), which denotes misuse transactions - that map values of x into probabilities or probability densities, and then later show the optimal test statistic for determining whether the particular value of x that was seen came from $N()$ or $M()$.

The ease in theoretically specifying that there is a space X belies the practical difficulties in actually defining X. (This difficulty directly affects the feasibility of implementing the Helman approach and has also influenced the NIDES statistical approach.) The following problems arise:

- Individual audit records typically do not contain complete values x within X. We observe a sequence of audit records, each of which contains limited information about user activity. Let us denote these audit records as r_t where t is the unique time stamp on the audit record. For example the value for r_t might denote that at the time that the last audit record was generated (i.e., time t) a particular command was being executed, a particular file was being accessed, the user was logged on from a particular terminal, and so forth. One method for' specifying a value x is to let x be the complete past history of audit records or, perhaps more reasonably, all audit records generated over the last 10 minutes. That is, $x = \{r_t, r_{t-1}, r_{t-2}, r_{t-3}, r_{t-4}\}$, **etc.**

- A substantial number of the values of r_t are missing. For example, a particular audit record might not contain information about files being used at that moment, or about I/O, memory, or CPU usage. In addition, because a user can (intentionally or unknowingly) spawn a number of processes - all of which can be simultaneously using computer resources (even on a single machine) - and each audit record is tied to a specific process, each audit record gives only partial information about the totality of activity that is going on in a single users̓ account at the time of audit record generation.

- Because audit records from separate processes can be intermingled (as different processes swap in and out of memory or are actually simultaneously executed on different processors), user behavior as recorded on audit records can appear to be quite erratic. This type of behavior is not well explored in the literature (which usually assumes rather well-behaved time series or stochastic processes).

- Because there may be long periods without any audit activity and periods of time with very intense activity, it is not immediately apparent how many audit records should be included in x. For example, if x is defined in terms of clock time (say 5 minutes), there will be periods in which x is composed of thousands of audit records, and many other times when x is empty. On the other hand, if x is defined in terms of audit records (say a few hundred audit records), there will be times when x denotes activity over a very short period of time (i.e., seconds or minutes) and times when x denotes activity over much longer periods of time (i.e., hours of near inactivity).

This discussion demonstrates the following:

- The dimensionality of x is potentially very large. At any particular time there may be many different processes executing (or temporarily in abeyance). Each individual process can have many different files open. Often, each process can have executed numerous commands. Whenever audit records address only very limited single actions (such as is the case with Sun-Unix audit records), we would want x to span a reasonable period of time. There is considerable uncertainty concerning whether to measure time by the clock or number of audit records, and either way can lead to undesirable results.
- The space X is at least as complicated as any outcome space considered in the statistical literature. The vast majority of statistical literature deals with very simple outcome spaces (e.g., independent or serially correlated outcomes, each of a fixed constant dimensionality) that cannot be applied to X without considerable modification.

In defining the NIDES statistical component we found it useful to let X be the space of all prior behavior (extending backwards in time to the first actions of the user). However, we weighted the audit records within x so that the more recent audit records were much more influential in our decision making. Furthermore, we found that by defining a set of statistics (i.e., long-term profiles) that were easily updatable exponentially weighted sums, we were able to avoid much of the complexity in the space X. For example, we could update our statistics only by looking at the most recent audit record, without having to consider the relationships of the current audit record to previous audit records.

4.3 Q: How Has the Size of the Space *X* Affected the NIDES Statistical Algorithm?

One tempting approach to anomaly detection is to enumerate all of the possible outcomes, observe the empirical probability for each of those outcomes for the normal user, and then declare a recent outcome (i.e., an audit record or set of audit records) to be anomalous if it is one of the outcomes that historically has rarely been seen. Unfortunately, the size of the space X precludes this approach.

The size of the space X is huge. Consider only two measures — command name used and file name used. A typical user might use 100 different commands and have a file system with 1000 different file names (excluding temporary files that are generated by processes executed by the user). In this event, the space *X* has 100,000 possible values (although only a fraction of these possible values might actually have been experienced). Add measures for host used, time of day, CPU time used, I/O amount, memory usage, and so forth, and the potential size of the space *X* becomes enormous. Conservatively, the size of the space *X* would contain trillions of possible values. Even if we limit the space *X* to those values actually experienced by the user, *X* would contain millions or billions of values. Simply enumerating these values (so we can keep a record of their historical non-zero probabilities) might overwhelm our storage capability, and processing this list would unacceptably slow down our algorithm.

In addition to processing capability problems inherent in considering the full space of outcomes, there is an even greater problem concerning the sample size requirements. As noted by Helman et al., “the sample size mandated by the enormity of the event space [*X*] . . . might exceed memory limitations, or perhaps, the [normal and misuse] processes . . ." cannot be assumed to be stationary over the time required to accumulate the requisite historical database.” By the time we gather enough data to obtain reliable probabilities for the events in *X,* the user will have changed job assignments, changed jobs, or possibly died. Even if user behavior were extremely stable over time, we doubt that any system security officer would be willing to wait one or more years before starting anomaly detection on a new user.

On the basis of the above discussion, we conclude that *in general* the “tempting” procedure of enumerating all possible approaches is infeasible. However, there is one important exception. If the event space *X* can be reduced in size sufficiently, then the approach is implementable. This appears to be the case in the Wisdom and Sense algorithm. Because W&S relies on higher level auditing information (essentially information at the network level about logins) the size of the space *X* is severely restricted. Even in this restricted space, processing is a challenging task, and the W&S developers deserve praise for the clever algorithms that they use to process and encode the data. We decided not to pursue this method of restricting the scope of data to process statistically with NIDES for three reasons: (1) we felt that the expert system could do a fairly good job of detecting many types of unusual login activity, (2) we wanted NIDES to be capable of processing more detailed information about user activity, and (3) NIDES was designed to be extensible and allow the inclusion of modules such as W&S.

Our approach to the problems raised by the size of *X* was to assess the unusualness in each measure individually. (After we had assessed the unusualness of each measure individually, we summed these assessments to obtain an overall assessment of unusualness.) This approach dramatically lowers storage and the sample size requirements. The corresponding loss is that we cannot distinguish changes in the combinations of values of measures. For example, if a user only executes command "a" with file “b” and command “c” with file “d”, then an intruder onto this user’s account who executes command “a” with file “d” and command "c" with file “b” will not be flagged as anomalous. (This limitation is not inherent in the NIDES algorithm. Although not currently part of NIDES, the NIDES statistical algorithm could be rewritten so that any two measures that are currently considered separately could also be considered jointly. For example, command name and file name could be jointly considered as a single measure, which takes categorical measures that are specified by both the command name and the file name.)

4.4 Q: What is the Relationship of the NIDES Statistical Algorithm to the Work of Helman et al.?

As mentioned earlier, the approach of Helman et al. is essentially to derive an optimal test statistic under the assumption that the probability distribution for misuse behavior is known. Helman et al. realize, however, that in practice probability distributions for misuse behavior will probably not be known. They therefore discuss two assumptions that can be made that dramatically simplify forms for these probability distributions:

- Independence - Under this assumption each measure (denoted by Helman as an attribute) is independent of each other attribute.
- Uniformity — Under this assumption, misuse behavior for each measure is assumed to be uniformly distributed over the possible values for the measure.

Because the NIDES statistical component does not assume a known probability distribution for misuse behavior, it cannot be easily related to Helman's approach. But we can make some observations:

- We do not believe that assumptions of independence or uniformity of misuse behavior are justified. Optimal procedures for detecting independent and uniform misuse behavior might be very poor in detecting actual misuse behavior. Nevertheless, it is desirable, in the absence of knowledge about misuse behavior, that an anomaly detector be at least capable of detecting independent and uniform misuse behavior.
- Because the NIDES statistical component uses a chi-square-like statistic to measure the differences between the short-term and long-term probability distributions for each individual measure, it will be capable of (and probably fairly good at) detecting uniform and independent misuse behavior.
- The NIDES statistical component can detect misuse behavior that is neither uniform nor independent. This is particularly true with respect to nonuniformity. Chi-square-type statistics are good at detecting an extremely broad range

of probability distributions that are not like the normal probability distribution. Their principal deficiency is that they are not as powerful as statistical tests formulated on the basis of a priori knowledge of the specific probability distribution of misuse behavior. That is, the chi-square-type statistics spread their power over a broad (in fact, infinite) range of probability distributions of misuse behavior.

- The NIDES statistical component does not assume independence of measures under either the normal or misuse behavior, nor does its validity depend on the independence of measures (that is, its false positive and false negative rates are uninfluenced by the dependence of measures). However, because the NIDES algorithm as currently configured does not place great emphasis on the correlation among measures, it is quite compatible with the assumption of independence of misuse behavior. In combination with the comments earlier about the uniformity assumption, we would expect the NIDES statistical algorithm to perform well when the misuse behavior is specified to be uniform and independent, although it will not perform as well as the optimal statistical test.

- Another type of misuse behavior that has not been explicitly considered by Helman is when the misuse behavior does not affect the marginal distributions of any measure, but does affect the correlation among the measures. As currently configured, the NIDES statistical component will not be particularly powerful in detecting this type of misuse behavior (unless the direction of change is to increase the correlation coefficient among the measures). However, it is possible to modify the algorithm in such a way that correlations are explicitly considered, thus, considerably strengthening the ability to detect changes in correlations. We view this as a potentially valuable area of future research.

4.5 Q: Can the NIDES Statistical Algorithm Consider Pairs of Measures Simultaneously (for example, command name and file name used)?

Theoretically, the NIDES statistical component can consider pairs of measures simultaneously, although this requires defining a new measure that is the cross product of two other measures. For example, consider the measure of name of file accessed and the measure of time of day. A composite measure might be defined as the two-tuple consisting of the name of the file accessed and the time of day of the access (perhaps categorized by hour of day). Each value for the two-tuple is a category for the new composite measure. The composite measure could thus, be handled by the NIDES statistical measure as a categorical measure in essentially the same way as any other categorical measure. There is no need to limit this consideration to pairs; this approach would apply equally well for triplets, quadruplets, and so forth.

The theoretical advantage of considering pairs of measures (or equivalently, of constructing new measures that are two-tuples of existing measures) is that interactions between measures can be examined. Continuing our example, suppose that user A (who works 8 a.m. to 5 p.m. on weekdays) accesses file B daily, but only during the hour after lunch. One day, however, file B is used frequently between 9 a.m. and 10 a.m. The NIDES approach as currently implemented would not detect such usage as anomalous, because it is not unusual for user A to use file B, nor is it unusual for this user to access the computer during these hours. The use of a two-tuple approach would be necessary for NIDES to detect as anomalous the use of file B during the morning hours.

We have not added a mechanism in the NIDES computer program for constructing new measures from groups of other measures. The primary reason for not doing so is that we have concerns that the number of categories in composite measures would become so large, and the frequency with which each category is seen would be so small, that the NIDES methodology would cease to function well. The NIDES methodology works best when user behavior is concentrated in relatively few categories.

4.6 Q: Why Does the NIDES Statistical Algorithm Use Exponentially Weighted Sums?

The NIDES statistical algorithm uses exponentially weighted sums, rather than a window of past values, for three primary reasons. First, we want to reduce storage requirements and accelerate processing. Use of a window of past values would require that we store all the past values in the window. Each time a new audit record of the appropriate type arrived, we would have to recall those values, delete the oldest value, and add the newest value to the list. We would then recalculate our test statistic to accommodate the deletion of the oldest value and the addition of the newest value. This is considerably more effort and requires more storage than multiplying our summary weighted sum statistic by a decay constant and adding in the value of the newest audit record. Second, we use an exponentially weighted sum because we do not have any “natural” boundaries that delineate past from current behavior and therefore would not know how long the window should extend into the past. Third, we do want more distant behavior to count less towards anomalies than more recent behavior. This does not occur with a window approach (except that audit records further in the past than the window boundary do not count towards anomaly detection), but occurs smoothly and gradually with an exponentially weighted sum approach.

4.7 Q: Why Does the NIDES Statistical Algorithm Treat Counting Measures as if They Were Categorical Measures?

The NIDES statistical algorithm converts counting measures to categorical measures by defining categories that are ranges of counts. We do so because we wanted to be very general with respect to the types of changes in counting distributions that we would detect as anomalous. For example, consider a user who has two modes of using the computer — reading mail and running large spreadsheets. The first type of activity would result in low CPU usage, and the second type of activity would result in high CPU usage. Therefore, this user's CPU usage would tend to be bimodal, with very few audit records recording intermediate CPU values. In such a circumstance, if we observed a large number of audit records that reflected activity with an average amount of CPU usage (that is, audit records in the rarely occurring center bins of the CPU usage distribution) we would tend to be suspicious. Such activity could not be detected if we only examined statistics such as mean CPU usage. Rather, it would be necessary to examine the entire distribution of CPU usage. We found that the most convenient method for examining the entire distribution for counting measures was to redefine them as categorical measures. This allowed us to use the already-existing methodology for categorical measures.

Other statistical methods for examining the entire distribution of counting measures have been developed in the statistical literature. For example, Kolmogorov statistics allow the comparison of an empirical cumulative distribution with a long-term cumulative distribution. However, such procedures tend to either be computationally intensive or require substantial amounts of storage, and few have been adapted

4.8 Q: Why Does the NIDES Statistical Algorithm Use Exponential Weighting Based on Counts of Audit Records (rather than Clock Time) for Nearly All of the Measures?

The NIDES statistical algorithm differentiates between the volume of audit records that are currently being received, the general type of activity that is represented in those audit records, and the specific type of behavior that is occurring within each measure. For the purpose of monitoring the volume of audit records we use three audit record intensity measures, which currently have half-lives of 1, 10, and 60 minutes. For these three measures the exponential weighting is based on time. A single measure, denoted audit record distribution, tracks what percentage of the most recent records have "triggered" or "touched" different measures. This provides an overview of the general types of activity that have recently taken place. For example, the audit record distribution measure would examine the percentage of recent audit records that concern a file access (that is, touch the file access measure). The audit record distribution measure uses exponential weighting based on counts of audit records. The remaining measures track the type of behavior that has transpired within a measure. For example, the file access measure examines which specific files were recently accessed, and whether or not these were unusual files.

In previous versions of NIDES, we did not subdivide the volume of activity from the specific behavior within that activity. This resulted in an inability to distinguish a relatively low volume of highly suspicious behavior from a high volume of behavior with typical levels of suspiciousness. The heuristic solution was to "normalize" our measures by the volume of activity; this was functionally equivalent to weighting on the basis of audit record counts and separately tracking audit record volume. This change to weighting on the basis of audit record count also facilitated the construction of short-term and long-term profiles, and the use of the chi-square statistic to compare those profiles. (Exponential weighting based on time is not compatible with the chi-square approach to profile comparison, and, as a result, the three intensity measures use a different approach.) There is a secondary advantage to the separation of audit record volume from the type of activity in those records — the measures are easier to interpret. We can separately detect whether the volume of activity is too high, whether the general type of activity within that volume is unusual (and in what ways it is unusual), and whether specific behavior within the activity is unusual (and in what ways it is unusual).

4.9 Q: For What Types of Computer Systems Will the NIDES Statistical Component be Most Applicable? Least Applicable?

The NIDES statistical component was developed to be very general and applicable across a wide range of systems and types of "subjects." Thus, it is our belief that the NIDES statistical component should perform well in many circumstances. However, there is no way of knowing exactly how well the statistical component will perform without actually testing it.

It is important to distinguish between relative and absolute performance. Some circumstances are generally favorable to statistical anomaly detection, including stable subject profiles, users with sharply defined and restricted behavior, and substantial differences in misuse and normal behavior. Statistical systems, including NIDES, will tend to perform better in these circumstances. Conversely, when these circumstances are missing, all statistical systems will tend to yield poor results. We have no way of knowing how well the NIDES statistical component performs relative to other statistical approaches because other approaches have not been available to us to test until relatively recently, nor have we been funded to perform such testing. We believe that such testing could be very beneficial to statistical algorithm development, not so much to determine which statistical algorithms are better, but rather to contribute to an understanding of why they perform well or not, so that we can use such knowledge to improve them further.

Because our most important sources of audit record data were our own Sun-Unix systems and because the immediate client's systems were also Sun-Unix, there are undoubtedly aspects of our statistical component that are rather "tailored" to these types of computer systems and users. We have also had an opportunity to test our system in two other very different environments. One of these environments was IBM mainframe audit record data gathered in field offices of a federal agency, where the predominant usage was among clerks entering and retrieving data from a fixed number of large databases. The second use was data where the subjects were applications, and the intent of using the NIDES statistical component was to differentiate among applications to detect masquerading by more resource intensive applications (constituting misuse of exported computers). In all three environments, the NIDES statistical system has performed very well. The last environment (which was known as "Safeguard" because the intent was to safeguard the usage of exported computers) is particularly noteworthy because it was amost diametrically opposed (in many important statistical ways) to the SRI Sun-Unix environment in which NIDES was developed:

- The subjects in the SRI Sun-Unix environment are sophisticated computer users, while the subjects in Safeguard are applications.

- The volume of audit records in the SRI Sun-Unix environment averages about 2,000 per user-day; the volume in the Safeguard environment averages only about 10 per day.

- Individual audit records in the SRI Sun-Unix environment contain only a small fragment of the information necessary to determine the behavior of a user; each individual audit record in the Safeguard environment contains complete information about an invocation of an application.

- There is no natural beginning and ending to an SRI Sun-Unix user's activity, whereas in the Safeguard environment each audit record represents a complete invocation of an application and is the natural analysis unit.

- In the SRI Sun-Unix environment the most distinguishing measures tend to be those which are categorical, and therefore the NIDES statistical approach was developed with categorical measures in mind; in the Safeguard environment all measures are counting measures.

We were pleasantly surprised to find that the NIDES statistical component performed as well as it did in the Safeguard environment, given its stark differences with the SRI Sun-Unix environment, and the relatively small changes in the algorithm parameters that were implemented. This gives us reason to be optimistic that the NIDES statistical component can be applied in other circumstances, which were not in our minds when the algorithm was developed. However, as mentioned earlier, there is no substitute for actually testing the algorithm in those environments.

Bibliography

[l] Michael R. Anderberg. *Cluster Analysis for Applications.* Academic Press, San Francisco, California, 1973.

[2] Richard O. Duda and Peter E. Hart. *Pattern Recognition.* John Wiley and Sons, New York, 1973.

[3] R. Jagannathan and Teresa Lunt. System design document: Next generation intrusion detection expert system (NIDES). SRI report, SRI International, Menlo Park, California, March 9, 1993.

[4] 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.

[5] S. Karlin. *A First Course in Stochastic Processes.* Academic Press, New York, 1969.

[6] R.A. Olshen L. Breiman, J.H. Friedman and C.J. Stone. *Classification and Regression Trees.* Wadsworth and Brooks/Cole Advanced Books and Software, Monterey, California, 1984.

[7] *Wisdom and Sense Guidebook.* Los Alamos National Laboratory, Los Alamos, New Mexico. Undated.

[8] Donald F. Morrison. *Multivariate Statistical Methods.* McGraw-Hill, San Francisco, California, 1976.

[9] G. Liepins P. Helman and W. Richards. Foundations of intrusion detection. In *Proceedings of the IEEE Symposium on Computer Security,* 1992.

[10] G. Shafer and editors J. Pearl. Readings in uncertain reasoning. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[11] S. Smaha. Haystack audit trail analysis system. Status Report HS_STAT.TXT, Haystack Laboratories, Colorado, August 15, 1990.