Detection of Anomalous Computer Session Activity

H. S. Vaccaro

Safeguards Systems Group
Los Alamos National Laboratory
Los Alamos, NM 87545

G. E. Liepins

Energy Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831

ABSTRACT

This paper briefly discusses Wisdom and Sense (W&S), a computer security anomaly detection system developed at Los Alamos National Laboratory (LANL). Anomaly detection provides another layer of defense against computer misuse after physical security and access security. W&S is statistically based. It automatically generates rules from historical data and in terms of those rules, identifies computer transactions that are at variance with historically established usage patterns. Issues addressed in this paper include how W&S generates rules from a necessarily small sample of all possible transactions, how W&S deals with inherently categorical data, and how W&S assists system security officers in their review of audit logs.

I. INTRODUCTION

It is widely presumed that computer misuse manifests itself as anomalous behavior and can be spotted as anomalies in activity logs. Despite recent major improvements to operating system security, available computer security features still are not good enough to detect many anomalous behaviors by computer "users" in time to prevent or even minimize any damaging activity. The risks from stolen passwords and privileges, for instance, are still of great concern. Current computer security systems do not in general protect against:

- An imposter who gains access to a legitimate account and environment;
- A legitimate user taking advantage of mistakes in the configuration of system security measures or other system vulnerabilities;
- A highly privileged user behaving destructively;
- An executable program that has been tampered, through other means, to perform some new, improper function (Trojan horse);
- Computer viruses and worms.

In practice, existing computer security systems are not necessarily configured effectively, so additional weaknesses typically exist.

W&S is an anomaly detection system; it seeks to identify usage patterns at variance with historical norms. In the context of computer security, anomaly detection identifies events shown in computer activity audit records that are inconsistent with routine operation and therefore may be indicative of an intrusion into the computer, serious human errors, or malicious behavior by a legitimate user. Access by an intruder, execution of "Trojan horses" and "viruses, as well as malicious, destructive behavior are all assumed to produce anomalous events that are recorded in a computer audit trail. This audit trail, perhaps with augmented data collection capabilities, can be processed in real-time to detect such events, alert a knowledgeable computer security officer to the threat, and help resolve the situation. LANL has developed Wisdom and Sense (W&S) to perform this task.

Within the larger framework of computer security, anomaly detection's proper place is as an additional layer of security after physical and access security. With few exceptions, current anomaly detection relies on a human security officer manually reviewing computer transactions, as captured in computer audit log records.* These logs are voluminous, and only a small fraction of the information might suggest system misuse; rarely is the security officer equal to the overwhelming task of sorting through the large number of complex and interacting patterns to identify those few that deserve close scrutiny.

Anomaly detection systems are intended to ease the system officer's task so that he can manage "by exception". Hopefully, an anomaly detection system can reduce the number of transactions requiring human attention by a factor of two to three orders of magnitude while retaining any transactions truly indicative of misuse. Because of a dearth of human expertise regarding what transactions are anomalous or represent system misuse, anomaly detection must be based largely on comparisons with historical patterns.

All anomaly detection systems must address two seemingly contradictory issues: the historically available transactions necessarily represent but a small fraction of all possible transactions; yet, the full historical data far exceed practical memory limitations and real time processing capabilities of today's computers. The challenge is to abstract and summarize usage patterns in a fashion

*Generally this paper will not distinguish between the events, or "transactions", and the records which record them in the audit log.)

CH2703-7/89/0000/0280$01.00 © 1989 IEEE

that allows for ready comparisons without over simplification. W&S achieves this goal through a carefully selected forest of instantiated rules automatically generated from the historical data. In the process, W&S also naturally deals with categorical data, a potential source of difficulty for many other anomaly detection systems.

This paper will not provide a detailed comparison among the known existing computer security anomaly detection systems currently under development: two at Lawrence Livermore Laboratory[1], two at TRW Inc.[2], one at Tracor Applied Sciences Inc.[3], one at the National Security Center[4], one at SRI International[5], and W&S at LANL. To be sure, a substantiative comparison is sorely needed, but the sensitive nature of several of the systems has closed them to scrutiny and analysis. Lunt provides the only available survey, but it was not comparative.[6] In addition, this paper will not provide the full theoretical justification of W&S; that remains for a following paper. The reader is urged to view this paper as a description of an evolving system, for W&S is continually being extended and improved.

II. BACKGROUND

Conceptual work on W&S began in November 1984, with early proof-of-concept software tests the following spring. Initially, the tool was applied to a related problem in nuclear materials control and accountability; but by the spring of 1987, tests on computer audit data had demonstrated it was capable of enhancing computer security.

The very first audit log analyzed with W&S contained evidence of a serious security flaw compounded by an administrative error. This flaw resulted in a string of events detected as anomalous by W&S.* More recently, W&S detected what appeared to be break-in attempts on the system administrator's terminal. Further analysis determined that the administrator's terminal line was noisy. In fact, in every audit log we have analyzed to date, W&S detected anomalous activity that justifiably caused concern.

The number of anomaly detection systems under development is not an indication of "waste", but rather it demonstrates the seriousness of the intrusion and misbehavior problems, and the difficulty of solving them. Moreover, there are significant differences in the various approaches.

W&S, in particular, uses a unique approach to anomaly detection for computer security. (Perhaps this is a result of its different roots in nuclear materials accountability and an empirical approach to problem solving.) W&S has a fundamentally categorical view of data and relies on its own examination of historical data to generate a rule set describing historical patterns. There are no rule templates to start with; rather the form of the rules is determined by the software's ability to detect patterns in the data. Not only does W&S come up with rules that an expert might suggest, but it also develops creative, often non-intuitive rules. Although a human often might reject such rules as nonsense, they have been found to play an important part in detecting security flaws.

For example, the security flaw mentioned earlier was detected in part by W&S-generated rules specifying that highly privileged users do not stay logged on for more than 12 hours, dial-up users do not stay on for more than 8 hours, and privileged users do not work over dial-up lines during normal working hours. Note that the flaw was that the operating system configuration would not terminate privileged sessions on dial-up lines. Oddly, unprivileged sessions terminated correctly. Things were compounded when a "regular" user was accidentally given system privileges. This user's privileged session continued for almost 2 days after he lost the telephone connection due to line noise from a nearby lightning strike. Until the system shutdown for maintenance, the user's privileged session remained active, and anyone dialing that line would have accessed the active session. (No one did.) The accident went unnoticed until W&S detected anomalies using rules that seemed "stupid" when we first reviewed them.

As it turned out, it was normal for this user to stay logged on for 24 hours or more, and, despite administrative rules to the contrary, system administrators occasionally operated with privileges over dial-up lines. Our own "expert" rules to cover this systems and its users would have been ineffective.

W&S seems to make up for lack of "expert knowledge" with creativity and diligence. W&S rule bases are very large with (10^4 to 10^6) rule instantiations, many of them seemingly irrelevant. We successfully tackled the problem of efficiently generating and using such large data structures. A typical generation takes less than an hour running on an inexpensive computer workstation, and the rule base normally can be searched in less than 0.05 second. This makes W&S capable of real-time anomaly detection on most systems. We are not aware of other computer anomaly detection systems that attain this performance.

In addition to its ability to generate rule bases without human guidance, W&S is also unique in its approach to historical data statistics. Other statistically-based computer anomaly detection systems operate on continuous, metric data. Categorical, or non-metric, data is mapped to some artificial metric so that the system can operate on it. By contrast, W&S has a fundamentally categorical view of data. It sees the universe of possible transactions as a collection of events, rather than as a continuous ordered space.

The majority of raw information available from computer audit trails, as well as from nuclear materials accounting systems, is in fact categorical. Thus, from the start W&S was conceived to handle predominately categorical data such as locations, object types, days of the week, person names, etc. Moreover, those continuous data items we analyzed, such as material batch weights, material processing times, CPU time usage, and computer input/output volumes, were often multi-modal, and typically skewed.

Rather than create metrics for fundamentally non-metric data and treat the continuous data as if it were uni-modal, normal data, we developed

*These examples are not from a computer system that handled classified data.

heuristics for dealing with the data in its categorical form and for mapping skewed, multi-modal continuous data to categorical data. After appropriate clustering, the various modes of the continuous data were treated as just separate categorical values as well.

It turns out that this data model works well for computer audit log records. Moreover, it allows us to develop rule bases that are quite human-readable. Categorical data are represented by unique character strings such as user "Bob" or program "FORTRAN.EXE", while continuous data is mapped to closed ranges such as "0 to 1" or "10 to 200" seconds or "0.1 to 0.5" grams. As a result, W&S rule bases can easily be supplemented or modified by a human expert. For example, the expert might add a rule such as "when the user is Bob and the program is FORTRAN.EXE, then with probability 0.99 the CPU time used will be between 0 and 1 or 10 and 200 seconds." Further, the expert might indicate that "when the CPU time is in the 0 to 1 second range, the program probably terminated with an error status."

Thus, W&S incorporates both human and machine-generated rules in a single rule base. The software recognizes any inconsistencies as the rule base is supplemented by human experts, and the human can use his own rule, the machine's rule, or a reconciliation of the two.

Viewing the rule base itself is an instructive effort. Note that rules can be retrieved by specifying templates with "wild cards" such as "? implies program AUTHORIZE" or "operator privilege & ? implies user ?." We often look for rules we expect, only to find that the historical data, as embodied in the rule set, disagrees with what we expect. Thus, we discovered rules indicating that someone other than the system administrator routinely uses the AUTHORIZE command (OK, after all), and that a certain user normally runs with operator privilege (not OK).

III. SOLUTION APPROACH

If behavior (for example, of a user or an executable program) differs from normal patterns, and if data indicating the difference is collected, it should be possible to compare the new audit data with normal patterns and detect the anomaly. The problem can be solved by creating, in effect, specialized profiles of computer users, terminals, executable software, privileges, time slots, etc., and determining whether the new data violates these profiles.

W&S receives audit data from the operating system and periodically (say every 2 weeks) processes it into a rule forest. The rules in the forest specify legal (historically acceptable) values of each field in turn, conditioned on observed values in various combinations of the other fields. A rule forest such as this has considerable redundancy, but this redundancy is necessary because it's impossible to tell ahead of time at what specificity an incoming transaction will match.

Moreover, we include additional redundancy to enable decisions even when much of the audit record is missing or suspect. The rule generator seeks to extract the maximum information from the historical data and store it in a reasonably small data space that nonetheless permits rapid searching.

Important issues include properly estimating the significance of individual rule violations and combining the evidence across the many rules. It is desirable to consider the evidence in an "even handed" way, and the best way to do this is still being investigated. In the interim, the current W&S implementation is as follows.

Selected historical audit log data is used to generate a tree-structured, instantiated rule forest describing historical behavior patterns that were statistically significant. The form of these rules depends on what patterns can be extracted from the historical data. Thus, there are no a priori templates, such as Username X implies Terminal Y. Rather, the rules define only what was observably normal for the values in particular fields.* This rule-making process is repeated for various combinations of conditioning values in the other fields (Fig. 1). If the historical data do not yield a good prediction as to what is normal, then the rule base makes no assertion.

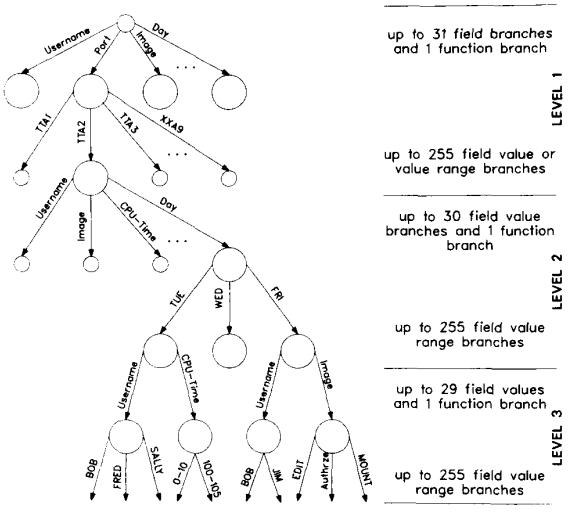

Fig. 1. W&S rule base structure.

Because of the scarcity of observed data and the drift in user activity patterns, W&S was designed to make rule generation simple. The historical data are acquired, then a rule base is generated in an unattended two-step operation. The total generation time is on the order of 1 hour; thus, it is reasonable to generate new rule bases daily. However, experience with actual data indicates that this is not necessary; rather, a scheduled regeneration every 1 to 4 weeks seems more appropriate.

*W&S can also work with values derived from a sequence of related audit records, say for a particular logon, and analyze these.

The form of our rule base makes it easy for a human to interface with the anomaly detection system. The rules are human-readable, and experts can add to or modify the rule base using an English-like syntax. An expert can verify the computerized decision by displaying the rules in that decision. Finally, W&S can use the learned patterns as a predictive tool to help the expert resolve anomalies by presenting non-anomalous alternatives to the observed activity.

IV. EXPLANATION

The audit log records we have worked with contain data about the execution of individual processes on a computer. One record is stored each time a process terminates. The data covers such things as who invoked the process, the name of the process executed, what privileges it possessed, and what system resources it utilized.

In our solution, a rule base creation program examines a history of audit log records to create and fill in-memory data structures with graded, instantiated rules. These structures compose what we call a rule base or rule forest. Rule bases can be written to disk storage or read from disk storage on demand. Naturally, the data used to build the rule base contains some anomalies, but these are heuristically filtered out of generated rules. The rule base may be further edited or supplemented by an expert only while it is loaded in memory. If desired, the modified rule base is saved so it can be reused.

A rule base is applied to audit log data records either in batch mode or real-time, to determine whether any activity, or series of activities, is abnormal when viewed against the rule base.

The rule base also reflects the quality of the behavior patterns it has learned. Patterns that occurred more often or with less noise have stronger grades.

A rule's grade is stored along with the conditions under which it applies (called the left-hand-side or LHS) and the implied conclusion (right-hand-side or RHS). A LHS in our approach is a series of (a) field values or value ranges, (b) computed values based upon data in a series of related records (for example, mean time between some event type), or (c) subroutines returning a boolean value. A given audit record satisfies a LHS and "fires" the rule if its values match those for the fields in the LHS and any subroutines in the LHS return true. Then we determine whether the record satisfies the rule's RHS conclusions.

We refer to the RHS as the rule's restriction as it restricts what is considered normal for a transaction. (The absence of rules means anything is considered normal.) Thus, our approach generates rules about the appropriate contents of audit record data fields based on the contents of other fields in the same record or data derived from a sequence of related records.

The natural observational unit for W&S is the transaction; but to develop statistics about the system or a user over time, some sort of aggregation is required. In W&S, this is accomplished through a mechanism we call threads.

A thread class is defined in terms of the data values of specific audit record fields. For example, we routinely define a user/terminal thread class. All records with the same user name and terminal identifier comprise a thread member for the class (for example, all records for "Bob" on "TTA1"). Associated with each thread class is a set of operations to be performed on a member's private data each time a new record is processed. The operations consist of computations such as event type counts, averages, and differences, and the resetting of local counters upon specific events such as logouts.

For any collection of historical audit records, the derived rule base must have at least one thread. An anomaly is detected for a thread whenever a new audit record for that thread is unusual and pushes the thread member's score over a threshold. The thread's score is referred to as its Figure Of Merit (FOM). The thread FOM is basically a time-decayed sum of FOM's for that thread's audit records. For the user/terminal thread mentioned above, a new logon session for the user on the same terminal is viewed as a continuation of the thread for that user/terminal. Thus, slightly anomalous transactions for the same user and terminal across several logons can lead to a thread anomaly. Program/user threads and privilege threads also are of high interest.

The RHSs are limited to three basic forms:

- A list of acceptable categorical, non-metric values for a particular record field (non-metric fields are those for which arithmetic operations are meaningless, e.g., normal work days of the week).
- A list of acceptable ranges for a continuous, metric record field (e.g., the normal amount of disk I/O activity).
- A list of user-defined functions to be executed until one returns a true value, meaning the RHS is satisfied, or the end of the list is encountered, meaning the RHS is not satisfied.

In the absence of rules restricting, say, normal computer terminals, any terminal is considered normal. This is, in fact, the same approach that humans take to most rules.

We find that W&S generally produces tens to hundreds of thousands of rule instantiations on a computer system's activity. The rules vary from very general ("the valid terminals are T1, T2, ... Tn") to very detailed ("on Tuesday between 6:00 am and 7:00 am, when the user has system operator privileges and is using terminal T3, only commands that cause very little direct disk activity are used") (Fig. 2). Also as with human experts, very specific rules carry more weight in making a decision, provided that they are still based on clear behavior patterns (that is, a low ratio of variation in the data to number of observations).

An expert usually takes many paths to arrive at a conclusion (e.g., the normal disk activity might be inferred from the user and time of day, or from the account number, or from the program being executed). A rule base is built the same way--it is highly redundant. Thus, the inferencing process reaches its conclusion about the normality of an audit record along many different reasoning paths, resolving conflicts through a weighting and scaling process.

Left Hand Side	Grade	Right Hand Side
Username AMY Privl 10148001	=7>	Image AUTHORIZE SET SHOW
Privl 10148001	=6>	Image AUTHORIZE SET SHOW
Privl 10148001 Priv2 00000008 DIR_IO 0:378	=7>	Image AUTHORIZE SET
Priv2 00000008 Privl 10148001 CPU_time 0:370	=7>	Image AUTHORIZE SET
Priv2 00000008 Privl 10148001	=7>	Image AUTHORIZE SET SHOW
Day "Wed" Privl 10148001	=6>	Image AUTHORIZE DELETE SET SHOW SYSGEN
Terminal TXC3 Privl 10148001	=7>	Image AUTHORIZE SET SHOW
Username OREN	=6>	Image AUTHORIZE COPY DELETE DIRECTORY LOGINOUT MAIL QUEMAN SET SHOW SUBMIT VMSHELP

Fig. 2. Examples of image "AUTHORIZE" rules generated by W&S.

An audit record field that violates many conclusions (RHSs) as to its normal content is considered anomalous. If the record contains several anomalous fields or a highly anomalous field, the record is anomalous. If a series of related records, say those for a particular user and terminal, are anomalous, the entire thread is considered anomalous.

V. SOFTWARE IMPLEMENTATION

These concepts are now implemented in three main W&S software sections: a data preprocessor, a rule base generator, and a transaction analyzer. Our implementation of these concepts has enabled the rule base to be stored in memory as a highly compressed tree forest using 6-7 bytes per rule, and the inferencing process to be real-time, firing roughly 20,000 rules per second on a $10,000 computer workstation.* Typical rule bases require 0.5-1.0 Mbyte of memory and can process about 20 audit records per second on the same workstation.

A. Design Criteria

We designed the anomaly detection software to embody the following capabilities:

- Reduce raw audit data to more usable forms;
- Build its own rule base without human guidance;
- Store and use very large, instantiated rule bases efficiently;
- Tolerate conflicting rules;
- Deal with uncertain and erroneous knowledge;
- Continue to learn from experience, and adapt to transient conditions;
- Accept human modifications to its rule base, but not be overly dependent on scarce human expertise;
- Make real-time, graded decisions regarding anomalous behavior;
- Provide human-readable feedback on anomalies to aid in anomaly resolution;
- Create minimal interference with the real functions of its host system;
- Be portable to different applications, operating systems, and hardware.

*All performance figures are for an IBM RT Model 6151-125 with an Advanced Floating Point Accelerator. The operating system is IBM's AIX Version 2.1.

Most of these design criteria have been attained in our software. However, there remain many gaps in our ability to detect anomalous computer activity and determine whether the anomalies are significant. We need more experience in operating environments, and with simulated intrusions, before we can design additional analysis tools for this purpose and properly tune W&S.

A difficult problem is that computer operating systems do not generally capture the right data for analysis. Though it is tempting to develop our own software to collect better audit data, few, if any, installations are permitted to modify their operating systems for the purpose of improving anomaly detection, unless the changes are made by the operating system vendor. Furthermore, the amount of data potentially available can easily overwhelm any anomaly detection scheme, so we will have to choose data of the greatest value once we have more extensive operating experience.

B. Data Preprocessing

For any given application, W&S is configured to read a specified fixed record format sequential audit log. The historical audit log, used by W&S in building a dictionary, condensed file, and rule base, generally contains 10,000 to 100,000 historical audit records (Fig. 3). A current activity file is of the same form, but it contains new records to be processed only through the inference engine.

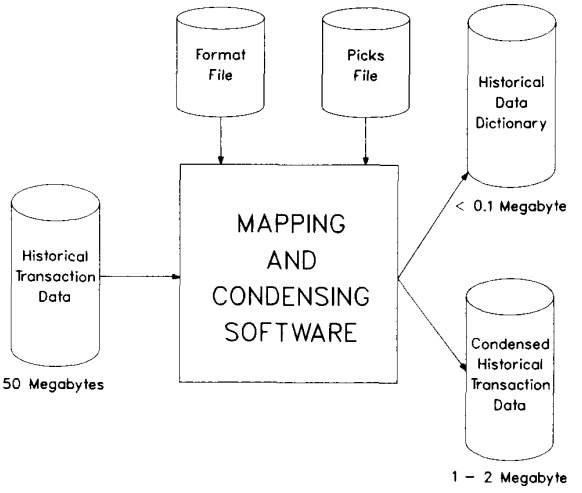

Fig. 3. Condensing an historical transaction file (typical file sizes on a 100,000 record VMS historical transaction file as input).

W&S is given a description of the history and activity files via a format definition file. Users can create this format file interactively from within the software.

The VMS ALAP (Audit Log Analysis Package) version of W&S uses "image termination" records from the DEC VMS accounting log as its audit record source. W&S extracts 16 fields from the standard

VMS image termination records, 13 of which are used for rule base generation, and the rest for display only (Table I).

TABLE I

IMAGE TERMINATION RECORD FIELDS
USED IN VMS ALAP W&S

Field Name	Metric	Comment
Priv1	no	first 32-bits of the privilege mask
Priv2	no	last 32-bits of the privilege mask
Status	no	32-bit program return code
Dir_IO	yes	Direct I/O - 512 byte blocks
Buf_IO	yes	Buffered I/O - 512 byte blocks
CPU time	yes	CPU milliseconds used
Username	no	User's login name
Image	no	Full name of the executed program
Day	no	Day of the week
Hr	no	Hour of the day
Terminal	no	I/O port name
Node name	no	Network node name, if any
Node ID	no	Network user ID, if any

Those 13 fields used for rule base generation are identified by a "picks" file, also created interactively or modified from within W&S. The user specifies a format file, then picks up to 32 named fields in the format definition.

For each picked field, an optional mapping function can be designated. Mapping covers operations such as ASCII to integer conversions, integer to ASCII, numeric scaling, shifting character strings to upper case, string truncation, floating point to integer conversion, etc. Proper mapping and data typing is essential to obtaining a good rule base. For example, the day of the month in the time stamp of an audit record could be treated as a metric integer. We find it far better to map this to a day of the week and hour of the day, and then treat those as categorical data (Fig. 4).

C. Rule Base Generation

We obtain high performance in rule base generation by condensing the historical file, so that it can reside in random access memory during the entire process. This condensed historical data is processed through the W&S rule base generator module, which builds the forest of rule trees. The rule base structure contains nodes of two alternating types. One type designates fields and can have

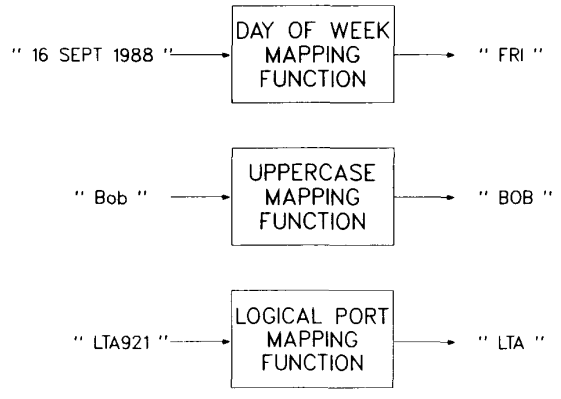

Fig. 4. Example mapping functions.

up to 32 branches. The other designates acceptable field values, and has at most 255 branches (Fig. 1). Together, the two node types compose a rule base "level"; thus the topmost level can have up to 8160 (32 x 255) branches. We typically limit depth to 4 or 5 levels, so the tree forest is very broad (with an upper limit of 2.6×10^{19} leaves at level 5). Pruning algorithms ensure that the actual generated rule base is of acceptable size.

Rules stored in this forest require an average of 6-8 bytes each. This is achieved by massive sharing of rule base data by rules with similar form, and through the use of the data value dictionary created when the historical data is condensed. A dictionary is typically small, on the order of a few tens of kilobytes.

The rules themselves are generated by repeatedly sorting the historical data and examining the frequency of field values within sorted subsets of the records. Because we represent our condensed historical data via 1- or 2-byte fields, very fast, in-memory, linear-time radix sorts[7] can be applied.

1. Metric Data Clustering. Rule-building on metric fields uses a simple, ad-hoc clustering algorithm. The sorted data are viewed as a histogram, with the histogram bucket widths being variable, but with each bucket containing roughly the same number of points. The target number of points per bucket is given by an empirically developed function of the total number of points and an estimated anomaly fraction. The widest buckets represent the portions of the number line with the lowest data point density (i.e., the unlikely values) and are tagged as "anomalous" ranges until a target for total anomalies is approached as closely as possible. Adjacent non-anomalous ranges are combined to yield contiguous "accepted" ranges (Fig. 5). The rule, assuming it is not pruned, then defines these ranges as normal values for the element given the LHS.

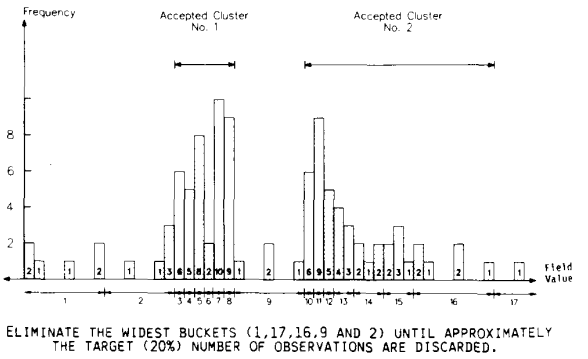

Fig. 5. Metric data clustering heuristic (100 observations, bucket size 6).

The metric element algorithms were originally tested on simulated binomial, normal, and multinormal distributed data with good results. Importantly for this application, the algorithms work

well with multi-modal data and do not require normally distributed data. Furthermore, the computation time is small.

2. Non-Metric Data Clustering. For non-metric fields, the least frequent values are tagged as "anomalies" up to approximately a target percentage. This target is based on expert judgment as to the likely number of anomalies in each field. (Choosing 0% for each field would mean that the expert believes there are no anomalies in the historical data for that field.) The values not tagged are considered acceptable, and make up the RHS of a new rule, if the rule is kept.

3. Rule Grades. Each rule has a grade which is a measure of the historical accuracy of the rule. This grade, G, is calculated as:

$$G = (\text{int})\log_2 \left[\frac{(T + 2) \times D}{A + 1}\right] ,$$

where T is the total number of observations, A is the number tagged as anomalous, and D is the current rule base depth. The +2 in the numerator and the +1 in the denominator result from an assumed uniform a priori distribution for the grade.* See Ref. 8 for computational details. (We are currently working on an improved grade calculation.)

The grade can be thought of as approximately:

$$\log_2[\text{historical odds of violating the rule}] + \log_2[\text{depth}] .$$

The second term biases the grades in favor of more specific rules, i.e., those with longer LHS's.

4. Pruning. Tree pruning ensures that the rule base contains predominantly "worthwhile" rules, and that it avoids exponential growth. Ideally, each rule should add at least some specified minimum amount of new information to the rule base. Here is what we have found practical for small (10,000-40,000 records) VMS accounting log files:

(1) Rules with a grade below a threshold (typically 3) are discarded, pruning the tree at that point.
(2) Rules with too many different acceptable values are discarded. The thresholds depend on the current tree depth. We currently use:
(# unique values) x (depth - 1) < 13
The first unconditional rules generated from the root of the rule base are at depth 1, and therefore are permitted to have the maximum number of branches (acceptable values)--255. Rules at depth 4 could have only 4 acceptable values.
(3) A rule whose LHS is a permutation of another rule's LHS is discarded.
(4) A rule whose RHS matches the RHS of another rule and whose LHS is a superset of that rule's LHS is discarded unless it has a higher grade.
(5) The rule is kept, but further growth is pruned if there is only 1 value allowed by the rule;
(6) Any rule's value with fewer than
30 + 50/(depth+1) observations
is not permitted to grow new branches;
(7) No rules below level 6 are generated.

*This is equivalent to the finite sample correction for the maximum likelihood estimator.

D. Transaction Analysis

The Sense modules (Fig. 6) provide an interactive, windowed interface to:

- the kernel's inference engine,
- transaction analysis tools,
- configuration settings, and
- rule base maintenance routines.

Its most important function, the detection and display of transaction anomalies, makes use of the rule base and dictionary generated by W&S and the inference engine described below.

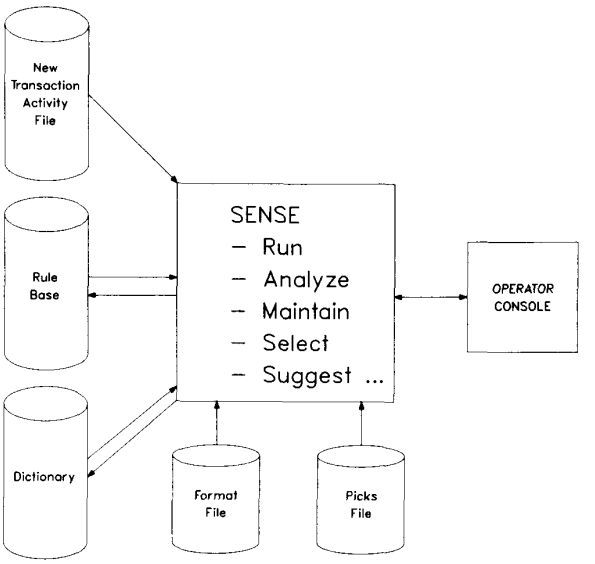

Fig. 6. Inferencing, analysis, and maintenance modules.

Sense, the anomaly detection module, looks at a new record, finds the rules that apply, and computes a transaction score. Normally, data that are newer than that used for rule base generation would be processed. However, the historical data must still be representative of the new data; if not, a new rule base needs to be generated from more representative (typically newer) data. This is accomplished by selecting a new historical data file to include the most recent data, then regenerating the rule base.

We are generally opposed to automatic or continuous updates on the belief that a human should determine whether the modified rule base is acceptable. Each new rule base should ideally be tested

against a set of known significant anomalies to assure that it is still effective. Further, the user needs some type of high-level comparison of the new and old rule bases. By contrast, continual rule base updating might allow a malfeasor to progressively weaken a statistical rule base by slowly expanding their envelope of "normal" activities until there is no discernable pattern. The malfeasor might then go undetected when truly damaging activity is undertaken.

The frequency with which rule bases need to be replaced depends on the stability of the system being monitored. On multi-user computer systems, processing new transactions against rule bases more than a few weeks old tends to produce unacceptably high anomaly rates. The number of records needed depends upon the diversity of activity on the system. Number of users is a practical surrogate measure of diversity. We look for roughly 500-1000 records per user. More is better up to say 10,000 records per user.

1. Scoring Transactions. As each record is processed by Sense, it computes a scoring function result for each field, for the transaction as a whole, and for any thread to which the transaction belongs. The score for each picked field is a function of the grades of each rule violated and each rule obeyed. A transaction score, or FOM, of 0 describes a transaction that is "perfectly normal" in every field.

A transaction *fires a rule* if the rule's LHS is satisfied by the data in the record fields. Thus, a rule of the form

> User Bob and Day Tue and Terminal OPA1
> implies with grade 7 either Image BACKUP
> or INIT or AUTHORIZE

would fire only if the record was for Bob using the console OPA1 on Tuesday. If Bob executed an image other than BACKUP, INIT or AUTHORIZE, the rule would be failed.

2. Transaction FOM. The *transaction FOM* is the weighted sum of the FOMs for each picked field (negative FOMs are set to 0 before summing). The FOM for each field is approximately the sum of scores for failed rules minus the expected sum normalized by the square root of the variance of this sum. In calculating these moments, we ignore the depth adjustment to a rule's grade, and we assume that all rules are independent.*

More precisely, for each field i, let:

N_i = number of fired rules,

K_i = number of failed rules, and

G_{ij} = the grade of the j^{th} rule fired on RHS field i.

*In fact, the rules are clearly not independent but are most often positively correlated with other rules. As a result, groups of related rules fire together and tend to be passed or failed as a group. This has a multiplier effect on the FOM for anomalous fields and drives non-anomalous field FOMs toward zero.

S_{ij} = the score ($2^{G_{ij}}$) for the j^{th} rule fired on RHS fields,

S_i = the sum of scores (S_{ij}) for all fired rules, and

F_i = the sum of scores for failed rules.

$$X_{ij} = \begin{cases} 0 \text{ with probability } 1 - 1/S_{ij} \\ S_{ij} \text{ with probability } 1/S_{ij} \end{cases}$$

X_i = the sum of X_{ij} from $j = 1$ to $j = N_i$.

If we view the outcome of each rule ij (pass and score 0 or fail and score S_{ij}) as an independent trial, then the sum X_i is a random variable that describes the cumulative outcome of rule application. (Note that in Sec. C.3 we defined S_{ij} such that the historical probability of failing the rule was approximately $1/S_{ij}$.) This viewpoint is used in computing an aggregate score for each record element i as:

$$FOM_i = \frac{F_i - E[X_i]}{\sqrt{\text{variance } [X_i]}}$$

The expected value can then be calculated from:

$$E[X_i] = \left[E \sum_{j=1}^{N_i} \left(S_{ij} \cdot \frac{1}{S_{ij}} + 0 \cdot \left(1 - \frac{1}{S_{ij}} \right) \right) \right]$$

$$= E\left[\sum_{j=1}^{N_i} 1 \right] = N_i$$

Asserting the assumption of independence of the X_{ij}'s, the variance can also be found as:

$$\text{variance } [X_i] = E[X_i^2] - E[X_i]^2 = E[X_i^2] - N_i^2 \ ,$$

where

$$E[X_i^2] = E[(X_{i1} + X_{i2} + \ldots X_{iN_i})^2] \ .$$

After expansion, the terms can be collected into two groups as

$$E[X_i^2] = E\left[\sum_{j=1}^{N_i} X_{ij}^2 + 2\binom{N_i}{2} \text{ terms of the form } X_{ij_1} \cdot X_{ij_2} \right]$$

$$j_1 \neq j_2$$

$$= E \sum_{j=1}^{N_i} X_{ij}^2 + 2\binom{N_i}{2} \cdot E[X_{ij_1} \cdot X_{ij_2}]$$

By independence, the expectation on the right is simply 1; that on the left evaluates to S_i^2, so we have

$$E[X_i^2] = S_i^2 + \frac{2 \bullet N_i \bullet (N_i - 1)}{2} = S_i + N_i^2 - N_i$$

and

$$\text{variance } [X_i] = S_i - N_i \quad .$$

Thus, our aggregate element score becomes

$$FOM_i = \frac{F_i - N_i}{\sqrt{S_i - N_i}} \quad .$$

The transaction FOM for a record with M picked fields is then given by:

$$TFOM = \sum_{i=0}^{M} \max(FOM_i, 0) \bullet W_i \quad ,$$

where W_i is the importance factor for the field i.

The thread FOM takes into account the scoring history of each transaction in the thread. FOMs for previous transactions in the same thread are decayed and then added to the current transaction FOM. The result after T transactions in the thread is computed as:

$$FOM_T = \sum_{t=0}^{T} TFOM_t \bullet d^{T-t} = TFOM_0 + TFOM_1 \bullet d \quad ,$$

where $TFOM_T$ is the FOM for the most recently observed transaction in the thread and d is some suitable constant between 0.0 and 1.0. With d near 0.0, only the current transaction carries significant weight. With d near 1.0, the thread FOM approaches the sum of all transaction FOMs for the thread processed so far.

3. Anomaly Detection. W&S finds an anomaly whenever either the transaction or thread FOM exceeds an operator-set limit. Transaction evaluation times are roughly proportional to the log of the number of rules. Thus a very large rule base need not be slow. W&S handles rule bases of up to 500,000 rules, averaging 6.0-9.0 bytes per rule and 20,000-40,000 rule firings per second. Typical transactions on these large rule bases have fired approximately 1% of the rules, resulting in measured performance ranging from 20-40 transactions per second for more typical (for W&S) rule bases of 100,000 instantiated rules.

4. Real-Time Issues. A serious impediment to real time detection, discussed by Denning,[9] is the difficulty of assembling user session and machine activity data into usable transaction records. Existing operating system accounting software simply does not make the job straightforward. As a minimum, the anomaly detection software will probably have to wait until the accounting software has written its data to disk. Highly buffered data may be written too late for real-time analysis. (For example, we experience buffering delays up to 9 minutes on our VAX running VMS 4.5.) Or, if data from separate accounting files (e.g., disk activity, user image executions, and keyboard input) must be matched and assembled, real-time may not be possible, or may be feasible only with a lower detection sensitivity (by treating each accounting data stream independently).

5. Anomaly Resolution. Anomaly resolution is the task of explaining the meaning and likely cause of an anomalous transaction. W&S attempts to provide information useful in this task; nonetheless, it is primarily one that must be accomplished by a human.

W&S currently offers four significant aids to anomaly resolution:

- Identification of the data in a record that appear to have triggered the anomaly;
- Listing of the violated rules that triggered the anomaly determination;
- Providing a thread history;
- Suggesting what data specific fields would have avoided the anomaly determination.

Each of these aids builds upon the inferencing process just described.

VI. CONTINUING EFFORTS

W&S is now undergoing operational tests in two computer security environments and one process monitoring environment. Preliminary results have shown that the software does periodically detect anomalies of high interest even in data thought to be free of such events. Thus far, we have tested only trivial intrusion scenarios (with successful detection). We hope to test the effectiveness of W&S on a wide variety of planted anomalous events during the current year. Futhermore, several enhancements, such as hybrid rule bases consisting of user-defined rules inserted into the generated rule base, will require extensive evaluation.

Nonetheless, it is already clear that the anomaly detection approach in W&S is effective for a wide range of applications where large volumes of repetitive data are generated by some chemical, mechanical, electrical, or biological system and where anomalous events are of interest. The heuristics employed in W&S make a reasonable compromise between computational accuracy and full use of available information, especially categorical and threaded data.

ACKNOWLEDGMENTS

We would like to thank the intrusion detection team at SRI International, especially Teresa Lunt and Hal Javitz, for their encouragement and constructive comments on W&S. At Los Alamos National Laboratory, we are especially indebted to James Tape and Jack Markin for urging us to pursue this work and for their continuing support of our R&D efforts.

This work was funded by the U.S. Department of Energy, Office of Safeguards and Security.

REFERENCES

[1] D. Mansur, Network Security Monitor, presentation notes for IDES Workshop, October 21. 1988.

[2] TRW Defense Systems Group Intrusion-Detection Expert System Feasibility Study, Final Report 46761, 1986.

[3] Stephen E. Smaha, "Haystack; An Intrusion Detection System," in Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[4] Michael M. Sebring, Eric W. Shellhouse, R. Alan Whitehurst, "Expert Systems in Intrusion Detection: A Case Study," in Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[5] Teresa F. Lunt, R. Jaganmathan, Rosanna Lee, Sherry Listgarten, David L. Edwards, peter G. Newmann, Harold S. Javitz and Al Valdes, "IDES: The Enhanced Prototype - A Real-Time Intrusion Detection Expert System," SRI International, Menlo Park, CA, SRI-CSL-88-12, October 1988.

[6] Teresa F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[7] Donald E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms (Addison-Wesley Publishing Company, Reading, Massachusetts, 1969).

[8] Ronald A. Howard, "Decision Analysis: Perspectives on Inference, Decision, and Experimentation," Proceedings of the IEEE, Vol. 58, No. 5, 632-643 (May 1970).

[9] D. E. Denning, "An Intrusion Detection Model," IEEE Transactions on Software Engineering, Vol. SE-13, No. 2 (February 1987).