SANDIA REPORT

SAND2005-7293
Unlimited Release
Printed November 2005

Profile-Based Adaptive Anomaly Detection for Network Security

Nancy A. Durgin and Pengchu Zhang

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-7293
Unlimited Release
Printed November 2005

Profile-Based Adaptive Anomaly Detection for Network Security

Nancy A. Durgin and Pengchu Zhang

Abstract

As information systems become increasingly complex and pervasive, they become inextricably intertwined with the critical infrastructure of national, public, and private organizations. The problem of recognizing and evaluating threats against these complex, heterogeneous networks of cyber and physical components is a difficult one, yet a solution is vital to ensuring security. In this paper we investigate profile-based anomaly detection techniques that can be used to address this problem. We focus primarily on the area of network anomaly detection, but the approach could be extended to other problem domains. We investigate using several data analysis techniques to create profiles of network hosts and perform anomaly detection using those profiles. The "profiles" reduce multi-dimensional vectors representing "normal behavior" into fewer dimensions, thus allowing pattern and cluster discovery. New events are compared against the profiles, producing a quantitative measure of how "anomalous" the event is.

Most network intrusion detection systems (IDSs) detect malicious behavior by searching for known patterns in the network traffic. This approach suffers from several weaknesses, including a lack of generalizability, an inability to detect stealthy or novel attacks, and lack of flexibility regarding alarm thresholds. Our research focuses on enhancing current IDS capabilities by addressing some of these shortcomings. We identify and evaluate promising techniques for data mining and machine-learning. The algorithms are "trained" by providing them with a series of data-points from "normal" network traffic. A successful algorithm can be trained automatically and efficiently, will have a low error rate (low false alarm and miss rates), and will be able to identify anomalies in "pseudo real-time" (i.e., while the intrusion is still in progress, rather than after the fact).

We also build a prototype anomaly detection tool that demonstrates how the techniques might be integrated into an operational intrusion detection framework.

This page intentionally left blank

Table of Contents

List of Figures

1 Introduction

As information systems become increasingly complex and pervasive, they become inextricably intertwined with the critical infrastructure of national, public, and private organizations. The problem of recognizing and evaluating threats against these complex heterogeneous networks of cyber and physical components is a difficult one, yet a solution is vital to ensuring security.

Detection of an intrusion is the first step towards assuring the network security against the threat from malicious actions. There are two basic approaches to intrusion detection: misuse detection and anomaly detection. Most current intrusion detection systems (IDSs) are based on misuse detection: they detect malicious behavior by searching for known patterns of intrusive behavior and raising an alarm when a known violation is detected. This is also known as pattern-based intrusion detection, since it works by performing pattern matching between current data and a database of known attacks. But a sophisticated and determined attacker cannot be expected to rely on known attacks – he will attempt to infiltrate a system with stealthy, sophisticated attacks using behavior that won't be detected by a static pattern-based scheme.

In contrast to the signature-based approach, anomaly detection measures the deviation of a group of activities from an established behavior baseline. If the deviation reaches a specified threshold, an alarm is generated. Anomaly detection techniques, therefore, have the potentials to detect new and unforeseen types of attacks. In addition, activity that may appear to be normal, in terms of the contents of the network messages, could still be part of an attack performed by an insider who is abusing their privileges. Anomaly detection can be used to detect this type of threat, where the anomaly isn't in the contents (payloads) of the network traffic, but perhaps the time of day, or the host or user that is accessing the information.

Traditionally, IDSs focus on a single component, such as physical security, or network security at the border firewall. As evolving technologies allow these components to be increasingly interconnected, we can no longer look at systems in isolation. Many IDS systems output a simple yes/no result – either an event raises an alarm, or it does not. There is often limited ability to adjust the alarm threshold, so the result is either too many false alarms, or failure to detect real attacks. An anomaly detection system allows the triggering threshold to be adjusted according to conditions, and can also enable combining threat levels from multiple components to make more intelligent decisions about whether to raise an alarm.

The objective of this research is to establish profile-based anomaly detection techniques that can be used to recognize the threats in the complex network. The "profiles" will reduce multi-dimensional data vectors representing "normal behavior" into fewer dimensions, thus enabling pattern and cluster discovery. New events will be compared against the profiles, producing a quantitative measure of how "anomalous" the event is. A set of criteria will be developed and used to determine whether an event is abnormal.

The goal of this project is to research techniques for profile-based network anomaly detection that can be used to address some of the problems outlined above. The anomaly detection approach is to distinguish between the abnormal events in a large event space and in a constantly changing environment. It is impossible to establish the baseline behavior profiles for host computers or networks with *a priori* patterns. The profiles should be formed from those data that represent the host and network behaviors. To achieve this, an unsupervised learning approach should be used. Unlike supervised learning, which requires *a priori* knowledge in its learning process, unsupervised learning establishes its categories, patterns or classes, according to the similarity of the data. When the profile, consisting of the patterns or classes which reflect the relative relationships among the data attributes and frequencies of records is built, knowledge of the observed object can be discovered from the formation of such profile.

Much of the academic research in network anomaly detection relies on publicly available "test datasets", including the 1998 and 1999 DARPA Intrusion Detection Evaluation Data Sets and the KDD Cup 1999 Data from The Third International Knowledge Discovery and Data Mining Tools Competition. These datasets are not necessarily ideal; for example, anomalies within the data are not clearly demarcated in the DARPA data, which makes training problematic. Many times the approach used in other research was artificially modified to accommodate for the test data.

The network data used in this project were collected from a Sandia-developed tool, NetState. NetState passively collects, and stores in a database, information about the host computers on a target network, including versions of operating system, applications on each host, and the high-level details (ports, IP addresses, times, amount, and direction of data transferred) about each host's network activities (Durgin et al., 2005).

One of the unique aspects of our approach is the dataset we focused on. Much of the current anomaly detection research have been focusing on examining low-level data (packet payloads or system logs) while we will focus on high-level session data. The low-level datasets tend to have tens or hundreds of features to examine, while our datasets can be limited to at most a few dozen features. This subtle difference should positively impact the results, even when employing similar data analysis algorithms. Furthermore, we assume that our profile-based anomaly detection system would be used in conjunction with packet and session-based systems, so we do not try to discover anomalies that are reliably found by those tools. The focus is on unsupervised learning techniques – that is, the training data will contain examples of "normal" events only, rather than both abnormal and normal events. In this work, we will not be inclined to make unnatural adjustments to analysis techniques to compensate for less than ideal data for our accessibility of the source of the data.

The rest of this report is organized as follows: In Section 2, we discuss background and related work. In Section 3 we discuss our work with Self-Organizing Maps. In Section 4 we discuss background on Adaptive Resonance Theory (ART), and in Section 5 we discuss our application of ART to the anomaly detection problem. In Section 6, we discuss our anomaly detection results from our ART prototype. Finally, in Section 7 we summarized the results obtained from this research and discussed the work needed in future.

2 Background

2.1 Related Work

Several anomaly detection techniques based on machine-learning or statistics based approaches have been developed for the purpose of network security, but the basic challenge still remains and needs to be clearly and adequately addressed: how to accurately model a subject's normal behavior while it changes over time in a continuous manner known as concept drift (Liao et al., 2004). The network activities and the users' behavior could alter over time caused by various changes such as application, mission, job assignment and et al. Modeling a host computer or a network's normal behavior in the presence of concept drift is a difficult task because the underlying data distribution is not known *a priori*, unexpected changes may happen at any time, and therefore a long term 'normal behavior' may not be strictly predicted and can't be expected that will be in the scope of the current 'normal behavior'.

An effective anomaly detection system should be capable of adapting to normal behavioral changes while still recognizing anomalous activities. Otherwise, a large number of false alarms would be generated if the model failed to change adaptively to accommodate the new behavior patterns (Maxion and Townsend, 2002). Continuously modifying the normal behavior model through adding normal behavior patterns seems obvious if one applies the anomaly detection in the practical environment. However, this may not be computationally feasible with the current techniques for their high cost in generating and storing models, for example, the approach mimicking the organism's immune system (Warrender et al., 1999).

The rule learning based adaptive approaches uses the rule learning concept, for anomaly detection in the dynamic environment: inductively generating sequential patterns (Teng et al., 1990), creating a nearest neighbor classifier (Land and Brodley, 1998), developing probabilistic model with mixture models (Eskin et al., 2000), using reinforcement learning method (Cannady, 2000), combining classification models and mining architecture with fuzzy association rule (Hossain and Bridges, 2001).

EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) from SRI (Stanford Research International) is a statistics based and ambitious project that has yielded mixed results in terms of adaptively detecting anomaly in network traffic data. EMERALD combines signature-based methods with statistical profiling, and consists of a set of modules which can be distributed throughout a network and individually tuned. EMERALD uses the method of aging a use profile to move a time window in which the most recently observed behavior is to compare with and update its profile. However, the user's recurring behavior can cause unnecessary update along with its complex statistical model.

Both statistical and rule-learning based approaches for anomaly detection, as mentioned above, are generally supervised learning processes, requiring *a priori* knowledge of the underlying data. In other words, supervised learning requires presenting training data that represents both anomalous and normal cases, which necessarily requires knowing what an "anomalous" case (an attack) looks like. To detect unknown attacks and the

unforeseen behavior of host computers, the recent attention in developing anomaly detection systems has been focused on the use of unsupervised learning processes, to establish profiles with no requirement for *a priori* knowledge.

2.2 Algorithm Evaluation

To select the promising techniques for profiling the host/network behavior, we first tested and evaluated a number of algorithms that have been used in data mining and machine-learning with emphasis on unsupervised learning algorithms.

The K-means algorithm is a method of cluster analysis (Orengo et al., 2002). It is an unsupervised learning algorithm that tries to uncover patterns in the set of input fields. Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar. This algorithm works by defining a set of starting cluster centers derived from data. It then assigns each record to the cluster to which it is most similar, based on the record's input field values. After all cases have been assigned, the cluster centers are updated to reflect the new set of records assigned to each cluster. The records are then checked again to see whether they should be reassigned to a different cluster, and the record assignment/cluster iteration process continues until either the maximum number of iterations, or the change between one iteration and the next fails to exceed a specified threshold. The K-means (sometime called X-means) algorithm seems suitable for creating profiles, but the scale-up and adaptive difficulties in K-means limits its application in our case, which involves a large dataset and has the goal to develop a tool that can adaptive to continuously changing host behavior.

The Two-step algorithm is another unsupervised learning method for clustering (Burges, 1998). It first compresses the data into a manageable number of small sub-clusters, then uses a statistical clustering method to progressively merge the sub-clusters into clusters, then merging the clusters into larger clusters, and so on, until the minimum desired number of clusters is reached. Again, difficulties of scale-up, stability and plasticity in applying this algorithm disqualify Two-step as a candidate for this project.

The C5 algorithm is a technique that is used to build up a decision tree or a rule set for the purpose of predicting (Dunham, 2002). This model splits the records based on the field that provides the maximum information gain. The lower layer records, meaning after the previously split, is split again depending on the selected field. The process repeats until the split can't be continued. Then, the lowest level splits are recursively reexamined and those that do not contribute significantly to the value of the model are removed. A set of rules is also associated with the decision tree. Apparently, C5 can't be used for profiling and anomaly detection in this case: the data used here to describe the host computer's behavior covers a large space and the differences among the data in a field can be ten orders of magnitude, thus the size of the tree (depth and number of branches of the tree) will be too large to be practically applied.

Methods based on artificial neural networks generally use supervised learning methods and are not suitable for profiling the network/host behavior. However, the two techniques we tested, Self-Organizing Map (SOM, Kohonen, 2001) and the Adaptive Resonance

Theory (ART, Carpenter, 1976), adopt the concepts used in artificial neural network, but they use unsupervised learning methods. These two algorithms have the capabilities of scale-up and dimension reduction as well as having the features of plasticity and stability, thus, they were selected for further detailed evaluation. Ultimately, an ART algorithm was employed to develop an anomaly detection prototype in this project.

This page intentionally left blank.

3 Profiling with Self-Organizing Maps

The Kohonen Self-Organizing Map (SOM) (Kohonen, 2001) is an unsupervised learning technique for data classification. SOM uses the "winner take all" strategy in its learning process, while continuously modifying the winner and its neighbor to build clusters that allow the records in the dataset to be classified into the clusters. A winner is one of the nodes in a user defined two dimension map. The combination of the clusters provides a profile describing the object which provides the data for SOM learning. If data from a new object can be distributed into the previously built profile corresponding to that of the original object, the new object can be considered similar to it. The concept here is used to build a profile of the behaviors of each individual host computer.

SOM is of particular interest here for its efficiency and its ability to express topological relationships. There are a number of reports on SOM applications in the area of cognition: sound filtering, hand-writing recognition and design flaw detection. A recent report indicates the potential of using SOM in detecting network intrusion by viruses (Lichodzijewski et al., 2002). However, there are few reports on SOM profiling network/host behavior found in the literature.

The general algorithm used to develop host profiles using SOMs is outlined in Figure 1. We developed an individual profile (map) of each host by selecting a set of training data for that host, normalizing the data, and then building a SOM for that particular host. After the SOM is validated it can be used as a profile of "normal" behavior in an anomaly detection system.

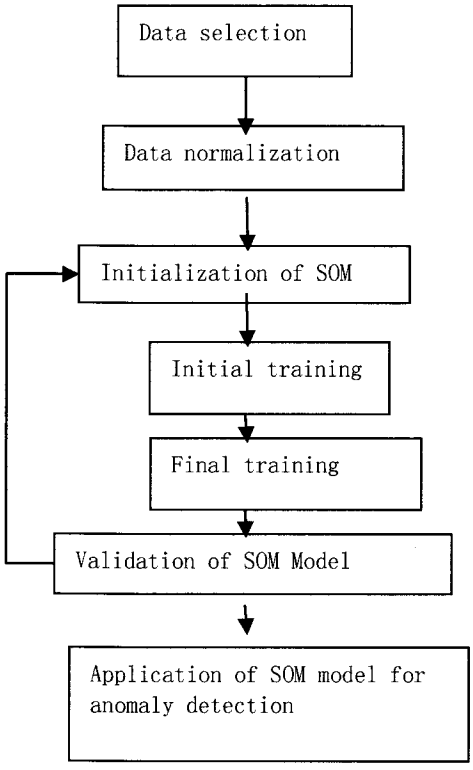

Figure 1. Procedure for building profiles with SOM.

3.1 Data Preparation for SOM Training and Testing

Several datasets used in this project were extracted from the database associated with NetState and represent the 'normal behaviors' of those host computers operating in a period of time. In each data set, a record for a host computer is the activity occurring between two computers, where the host of interest is either the source or the destination for that session. The following features (attributes) are included in a record for each session:

start date/time
end date/time
source IP address
destination IP address
bytes of input
bytes of output
port number

A subset of the features available in the NetState data is selected to be used in the SOM training datasets. From the fields of in/out data size, start/end of connection date/time we derived a series of numerical inputs for model training and testing:

bytes_in bytes per second
bytes_out bytes per second
bytes_in per event

bytes_out per event
time interval per event
number of transaction per event
frequency of the host serving as a client hourly
frequency of the host serving as a client daily
frequency of the host serving as a server hourly
frequency of the host serving as a server daily

It should be pointed out that not all of the features derived are used in all of the SOM model training.

Several data normalization methods have been tested for the data. The Logarithm Scale was selected for data normalization for SOM training. The data values in some fields vary in a wide range and the Logarithm Scale brings the data in the range of negative singles to positive tens. Other methods such as a statistic-based method were also tested.

3.2 SOM Model Training

3.2.1 Clementine Datamining Tool Bench

The Kohonen model tool in Clementine (Clementine 8.0, SSPS, www.ssps.com) was initially used to train a model with the selected normalized data. There are a number of basic training parameters that can be selected for the training process. The outputs of the tool give each record two integers that position the record in the trained SOM profile (model). The density of the points in the map is the number of 'hits' to a node, where that node was selected as the winner, in the map. The location and the density of the records consist of the patterns of the dataset for a host computer and the patterns are considered as the behavior of the host.

An example in Figure 2 shows the different distributions of records ('hits') in four maps from four different hosts. Among the four maps, the upper left map is the SOM map for the test dataset from the same host as the original training dataset for that map. From the distribution of the nodes in the map and the density of the 'hits' in the nodes, which represent the patterns of a host's behavior, the upper left map clearly differs from the other three, which were built from training datasets from three other host computers. For example, the map at the left bottom does not show the 'hits' in the nodes of the third column, thus the behavior represented by this map is different from that at the upper left map. This illustrates that the SOM method is capable of distinguishing the behavior of one host from others, and that it has the capability to reduce the multi-dimensional data into a two dimensional map, thus making it easy to visualize and understand. Another way to describe this is that if the behavior of the host changes, the distribution of records on the trained SOM map will be different, or show abnormal behavior.

Figure 2. Distribution of data from four host computers into a trained SOM profile.

However, the Clementine tool has a limited capability in SOM training parameter selection and the output from this tool can't be easily deployed for data analysis and result interpretation. Plus, a long computation time is required for training and testing. Therefore, other tools were investigated.

3.2.2 SOM training with SOM Toolbox and SOM_PAK on MATLAB

SOM_PAK, a group of classes in C for SOM training, and SOM Toolbox, a package of programs for SOM written in MATLAB, were developed by a research group in Helsinki University (SOM Toolbox for MATLAB, http://www.cis.hut.fi/projects/somtoolbox/). Both packages can be executed individually. The SOM_PAK routines, which are compiled from C, can also be called directly from the MATLAB SOM Toolbox package. This decreases executions times significantly compared with using the MATLAB SOM Toolbox routines.

As recommended by T. Kohonen (Kohonen, 2001), the "hexes" topology for constructing a map has been used. Map training was linearly initiated.

Two phases of training, coarse and fine, were conducted.

First phase of training:

> Dimensions: X and Y are initialized to the range of 10 to 70, however, there is a function in SOM_PAK or SOM Toolbox that can initiate the dimensions of the map and other training parameters

Length of interaction: depending upon the number of records in the training set, as general rule, it is equal to or close to the number of records

Learning rate: 0.7 – 0.9

Distance from the central point: less than the X or Y dimension number

Second phase of training:

Length of interaction: 1000 times of the product of X and Y

Learning rate: 1/20 to 1/30 of the learning rate used for the first phase

Distance from the central point: 2 is generally used

Two properties were used to measure the quality of a trained map:

data representation accuracy
data set topology representation accuracy

The first one, data representation accuracy, is measured using average quantization error between data vectors and their BMUs (Best Match Unit) on the map. For dataset topology representation accuracy, several measures have been proposed: the topographic error measure and the percentage of data vectors for which the first and second BMUs are not adjacent units. Both methods are implemented in the SOM Toolbox and are used in our SOM training processes.

In this map training process, the quality of the trained map is also measured by the ratio of the distance between a training data (or test data) vector and the "winner" in the trained map. The "winner" here is the neuron node that is "closest" to the training data vector, meaning it is the node whose data vector has the shortest distance to the training data vector. Thus each training record (the vector of data) is assigned to the nearest neuron. A well trained SOM is expected to have 90% of the data records in the training dataset within a pre-set threshold distance from the corresponding winner.

3.2.3 Data for training, testing and validating

The training datasets were generated by randomly extracting records from the entire dataset. Since the available dataset for some hosts were larger than for others, the size of the training datasets for a particular host depends on the size of the original dataset for that host. Generally, there were 3,000 records for a training dataset. For a host dataset with less than 2,000 records, 50% of the records are taken for the training dataset.

In addition to the training dataset, test and validation datasets were also built by randomly sampling from the original dataset. The records in those datasets were generally different from that in the training datasets.

3.2.4 Results and Discussion

The data in Table 1 can be considered as a 4x4 matrix which is used to demonstrate the capability of the trained map in describing itself and distinguishing itself from other maps for different hosts. The values are the percentages of records in the test datasets that have an acceptable distance (1.5 of the distance unit used here) from a node (winner) in a trained SOM map. The test datasets are sampled from datasets A, B, C and D and are compared against the SOMs built for A, B, C and D. Ideally, the values in the **diagonal** (as bolded in the first matrix below) should be high and rest of it should be low.

Table 1. Percentages of records in the four test datasets classified into the four SOMs that were trained with the corresponding four host computers data.

	Test Data set A	Test Data set B	Test Data set C	Test Data set D
SOM A	**100**	70	0	0
SOM B	95	**75**	5	0
SOM C	5	15	**95**	70
SOM D	5	5	80	**100**

The following graph (Figure 3) may further help to describe the argument made above. If a map is well trained, it means the map is able to describe the hosts' behavior and capable of detecting any other data points that do not belong to its own host. Only four columns (model A: data A, model B: data B, model C: data C and model D: data D) should be the dominants (higher than 90%). From the graph, one can observe that Model A and B can clearly distinguish themselves from C and D and vice versa. However, Model A and B can't distinguish each other very clearly. Model C and D can not only distinguish themselves from A and B they can also distinguish between each other very clearly. The similar observations seem to be true to most of the experimental results in above data matrixes.

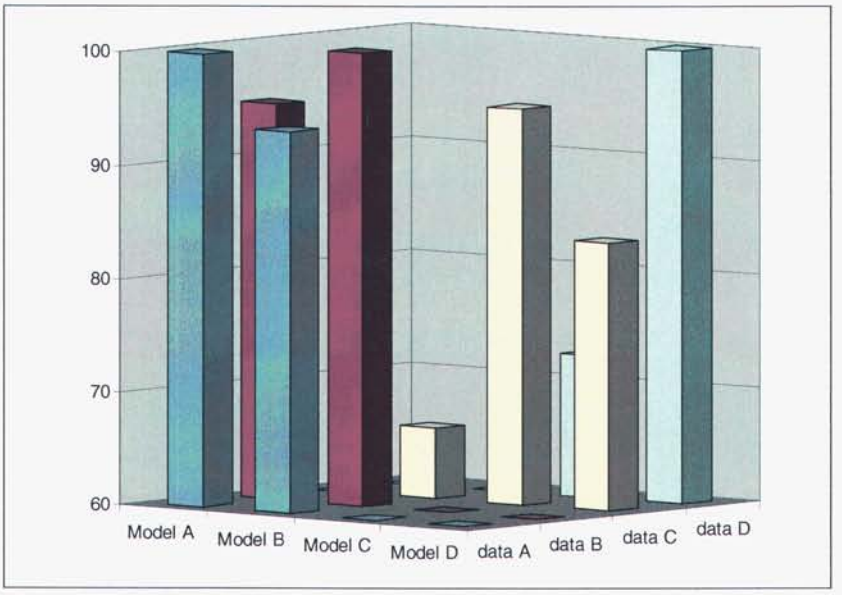

Figure 3. Rates (in %) of data points lower than 1.5 distance units.

From the results of these experiments, a trained SOM can generally be used as a tool to profile a host computer behavior and to detect behavior that does not agree with that host. However, the sensitivities of the trained SOM maps may not be high enough in distinguishing the behaviors of the host computer from others. If the behaviors of host computers A and B are similar, the maps should generate similar profiles. Note that there weren't an adequate number of data records available for dataset B to have enough records for map training and testing, therefore the quality of the map B is uncertain.

In general, from these preliminary SOM experiments we can conclude that the SOM is capable of profiling behavior for host computers but not sensitive enough to clearly distinguish the behavior profiles among the host computers. One reason for this is that the experimental set-up was overly simplified in dealing with data that have such high dimensions (attributes) and a very large number of records. Some attributes that are highly important in determining the behaviors may be over shadowed by the large number of attributes. A multi-layer SOM hierarchical framework may be the solution in which features that have an internal connection can be grouped together for a SOM building and then at a higher level a SOM can be built with the sublevel SOMs' data. Thus, the effects of important attributes will not be over shadowed in constructing the behavior profiles.

This page intentionally left blank.

4 Adaptive Resonance Theory (ART)

4.1 Background

Adaptive Resonance Theory (ART) was introduced as a theory of human cognitive information processing (Carpenter and Grossberg, 1987). The theory has since led to an evolving series of real-time neural network models for unsupervised category learning and pattern recognition. These models, including ART1, ART2, ART3, ARTMAP, Fuzzy ART and Fuzzy ARTMAP, are capable of learning stable recognition categories in response to arbitrary input sequences. ART1 can stably learn to categorize binary input patterns in an arbitrary order. ART2 can stably learn to categorize either analog or binary input patterns presented in an arbitrary order. ART3 is based on ART2 but includes a model of the chemical synapse that solves the memory search problem of ART system embedded in network hierarchies. It can carry out parallel search of distributed recognition codes in a multilevel network hierarchy. ARTMAP can rapidly self-organize stable categorical mappings between m-dimensional input vectors and n-dimensional output vectors. Fuzzy ART, which incorporates computation from fuzzy set theory into the ART neural network, is capable of rapidly and stably learning to recognize categories in response to arbitrary sequences of analog or binary patterns. Fuzzy ARTMAP, the combination of ARTMAP with Fuzzy ART, can rapidly learn stable categorical mappings between analog or binary input and output vectors (Carpenter et al., 1991).

4.2 Fuzzy ART

Fuzzy ART is a member of the ART algorithm family that incorporates fuzzy set theory into adaptive resonance theory. This combination enables the unsupervised categorization learning and pattern recognition processes which are faster and more stable for ART algorithm in responding to arbitrary input sequences (Carpenter et al, 1991).

Fuzzy ART clusters input vectors into patterns based on two separate distance criteria, match and choice. For input vector X and pattern j, the match function is defined by:

$$S_j(X) = \frac{|X \wedge W_j|}{|X|}$$

where W_j is the weight vector associated with pattern j. The fuzzy AND operator ^ is defined by:

$$(X \wedge Y)_i \equiv \min(x_i, y_i),$$

and the norm |.| is defined by

$$|X| = \sum_i |x_i|.$$

The choice function is defined by

$$T_j(X) = \frac{|X \wedge W_j|}{\alpha + |W_j|},$$

where α is a small constant.

For each input vector X, Fuzzy ART assigns it to the pattern j that maximizes $T_j(X)$ while satisfying $S_j(X) \geq \rho$ where ρ is the vigilance parameter, $0 \leq \rho \leq 1$. The weight vector W_j is then updated according to the equation:

$$W_j^{new} = \beta(X \wedge W_j^{old}) + (1-\beta)W_j^{old}$$

where β is the learning rate parameter, $0 \leq \beta \leq 1$. If no such pattern can be found, a new pattern node is created. This learning and categorizing procedures is illustrated in Figure 4.

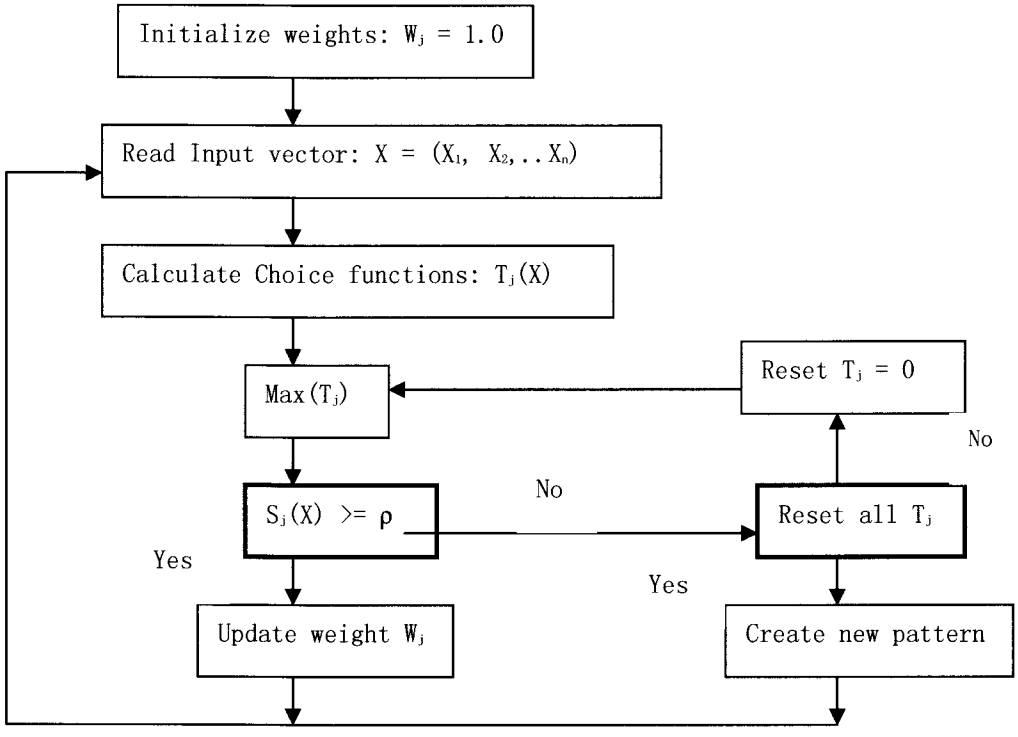

Figure 4. Fuzzy ART learning algorithm.

In order to avoid the pattern proliferation problem, Fuzzy ART used a complement coding technique to normalize the inputs. The complement of vector X, denoted by X^c, is defined as:

$$(X^c)_i \equiv 1 - x_i.$$

For an n-dimensional original input X, the complemented coded input X' to the Fuzzy ART system is the 2n-dimentional vector:

$$X' = (X, X^C) = (x_1, x_2, \ldots x_n, x_1^c, x_2^c, \ldots x_n^c).$$

Using the ART model in profiling the behavior of a host computer for the purpose of anomaly detection may help to establish a pattern network with unsupervised learning for the dataset that contains unknown number of patterns. More importantly, through adjusting the value of vigilance, the parameter in controlling ART learning process, the generality of the network can be adjusted to recognize objects in various specific levels: recognize a dog from other animals and also recognize the dog from its brothers. It should be pointed out that most above description of Fuzzy ART are taken from Carpenter et al. (1991) and Liao et al. (2004).

This page intentionally left blank.

5 ART Anomaly Detection Prototype

5.1 Data Preparation for Fuzzy ART Profile Training

In addition to the four host datasets used for SOM training, several more datasets were added to the Fuzzy ART exercises. The new datasets were also from NetState. The numerical values representing the host computer's activities are required to train an ART model. Five features are derived from the datasets for each of the applications:

bytes_in	input bytes for an event
bytes_out	output bytes for an event
duration	seconds of an event
hour_work	operation hour, 1- 24
day_work	operation day, 1- 7

The host computer's applications are categorized based on the destination port numbers. By reviewing the frequency of application in most datasets used, six application categories were developed (application (port number(s)):

http(80, 443), ssh(22), smtp(25), ftp(21), pop(109), other(any other port numbers)

It should be pointed out that the application categories can be extended to make the training dataset more generic. In this project, the first five are the most frequent in the datasets, thus more representative to the host computer's behavior. Any other applications represent a small proportion of the activities and lumped into the 'other' category.

Thus, the dataset for an ART model learning, testing and validating test contains 6 x 5 columns for each record. Since there is only one application for a particular record, five numerical values are assigned to an application and the rest of the 25 columns contain the value of zero as showing the example in Table 2

Table 2. Sample data format used for building host computer behavior profile (11 of 30 columns).

http					ssh					smtp
Byte_in	Byte_out	Duration	Time_work	Day_work	Byte_in	Byte_out	Duration	Time_work	Day_work	Byte_in
112	123	122	22	3	0	0	0	0	0	0
0	0	0	0	0	45	456	667	12	7	0

By arranging the dataset format like this, it allows ART to learn the host computer's behavior for separate application and establish a profile that contains any cluster that the training dataset may have. For example, if the training dataset contains six applications, among them 99% of the records might belong to http with the rest of them distributed into the other five applications, with at least one record for each application. The profile established by the ART learning process includes at least six clusters that correspond to each of the six applications, assuming there is no other significant difference among the records of an application. When the trained profile is applied for anomaly detection, if the

tested dataset contains a significant amount of applications that only can be classified into the cluster of 'other', the relative portion of 'other' will increase in the profile and an abnormal behavior can be detected.

Another reason for selecting this data format is that data normalization is required for training ART framework to establish profile. The typical form used for data normalization is:

$$d_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}}$$

where v_{min} and v_{max} are the minimum and maximum values for a attribute. It is more proper to normalize the input data within that application. For example, the minimum and maximum values for bytes_in and bytes_out in HTTP may not be comparable to those expected for SMTP.

5.2 The Fuzzy-ART Program and Interface Development

The open source package of Fuzzy-ART Neural Network Implementation was developed by A. Garrett (v 1.0, 2002, www.mathworks.com/matlabcentral/fileexchange) at Jacksonville State University. This package was written in the MATLAB language and consists of two ART models: Fuzzy ART and ARTMAP. Modifications have been made to allow the methods in the package to take input that is different in format from what the package expects. A series of utility methods have been developed for data normalization, model training, testing and result calculation and reporting.

MATLAB has been designed for mathematical computation. In particular, the built-in matrix operations make data input easy, since it is in the format of a table, with the columns representing the attributes and each row representing a record. MATLAB simply reads the table in as a matrix. Since MATLAB is built as matrix-based to conduct computation, many matrix manipulation functions in MATLAB make the Fuzzy ART training and application efficient to implement and execute.

Training Parameters:

Unlike other learning algorithms, Fuzzy-ART or any other ART algorithms need only two parameters to control the learning process: the vigilance, and learning rate. For vigilance, ρ, in general, the lower value of vigilance leads to a coarse learning process and high vigilance value will lead to establish a profile that can distinguish more detail of an object. But the tradeoff includes a higher learning time and higher level of noise. The learning rate is set to 1.0, the value typically used for Fuzzy ART training.

5.3 Datasets for Training, Testing and Validating

A set of three datasets were used for Fuzzy-Art training, testing and validating, respectively. The records in the datasets were randomly extracted from the raw dataset of a host computer. The records in the three sets, although coming from the same host computer dataset, were different from each other.

The size for the training, testing and validating datasets are: 10,000, 1,000 and 2,000. These sizes were selected based on a series of preliminary tests.

5.4 Representative of Behavior Profile

A profile obtained from the Fuzzy ART learning process is an array of patterns in which each of the patterns is a collection of records with similar features. The number of patterns and the distribution of the patterns' sizes in a profile characterize the observed object, in this case, the behavior of the host computer. Figure 5 shows an example of a profile. The bar graph shows that there are a total 18 patterns in the profile and the distribution of the patterns in the profile. The X axis of this graph represents the number of patterns in the profile and the Y axis is the measurement of the relative proportion of each pattern in the profile.

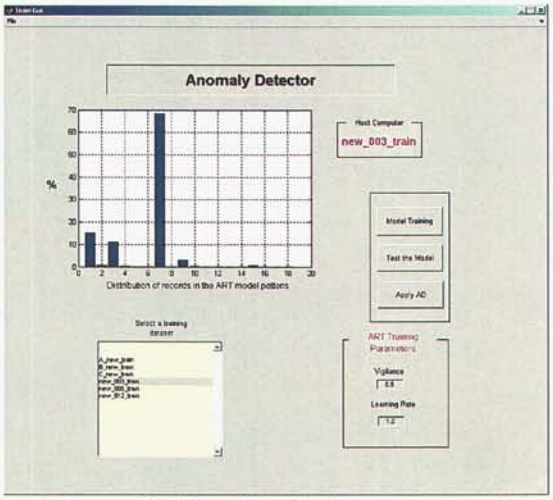

Figure 5. Example of profile of a host computer behavior

Considering the behavior of a host computer and the way used to establish the profile with Fuzzy ART, a pattern can carry the information that describes the activities with a particular application (e.g., http), that is similar in the amount of information exchange (bytes in and out), similar in operation time (duration) and time and date the host computer operation (time of day and date of a week). If another computer operates similarly to the host computer, it should have the similar number of patterns and the similar distribution of pattern sizes in the profile when its dataset is used to get a profile with the same Fuzzy ART set-up. By comparing the two profiles, the differences will be used to evaluate whether an anomaly has occurred.

5.5 Criteria Selection for Normal/Abnormal Measurements

To distinguish anomalous behavior from the normal behavior of a host computer, the data collected from a period of time will be used to establish a profile against the profile that Fuzzy ART generates with the normal behavior data. Each of the records in the new

dataset will be evaluated to determine whether it has the same features as that of any pattern in the normal behavior profile. Through this, all records in the new dataset are accepted or rejected by the profile. The record distribution of the new dataset becomes a profile and can be used to compare against the normal behavior profile. If the difference between the two profiles is significant enough, then an alarm can be issued for possible occurrence of anomaly.

However, determination of the significance between the two profiles is arbitrary. Depending upon the problem, one only can develop the measurement through experimental tests for specific application. In this project, three categories are used to measure the similarity between two profiles:

Percentage of Passed Records (PPR), this is the percentage of the records in the new dataset that can be classified into one of the patterns in a normal behavior profile.

Sum of Difference of Profiles (SDP), this the sum of the absolute differences of the two corresponding patterns between the two profiles.

Spearman Coefficient (SC), this is a statistical measure for rank correlation between two sets of values:

$$r = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the ranks of corresponding values and n is the number of pairs of values. If the two groups are identical, the Spearman Coefficient is 1.

5.6 Prototype development

The anomaly detector for the network/host computer prototype consists of a set of graphical user interfaces (GUIs). The basic architecture of the network/host computer anomaly detector prototype is shown in Figure 6.

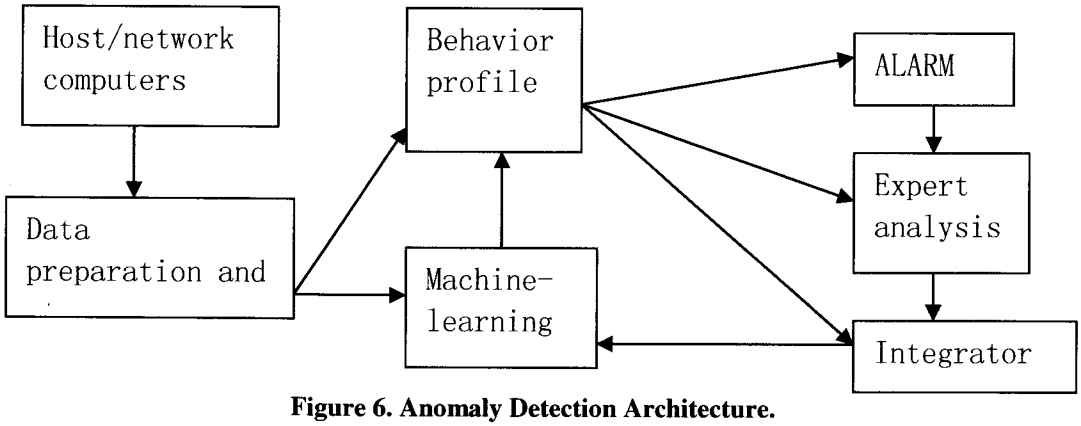

Figure 6. Anomaly Detection Architecture.

The host or network traffic data are collected and stored using NetState. Then the data is formatted and used for machine-learning through the Fuzzy ART algorithm to generate a behavior profile. When the profile is tested and validated, it will be used to compare against the host/network traffic data. If an abnormal behavior is detected, an alarm will be raised and the detailed information for the alarm will be sent to network security personnel for evaluation, and the section of the traffic data that caused the alarm will be stored in a buffer zone. If it turns out that the abnormal behavior is not a threat and is caused by a behavior change, the Fuzzy ART algorithm will be used to retrain the behavior profile in which the new normal behavior will be included. A mechanism can be developed to atomize the processes of training, testing, applying and retraining, thus to atomize adaptation of this system.

5.7 GUI Development for Anomaly Detector Prototype

The anomaly detector prototype is composed of a series of graphic user interfaces (GUIs) which were developed with the MATLAB GUI development tools. These GUIs have the functionalities of learning the behavior from a selected host computer, establishing a profile based on it, testing the profile and retraining the model if necessary, applying the model to the data sent by the host to detect abnormal behavior and reporting the results. The functionality of monitoring a host on-line, or so-called pseudo-real time monitoring is also built to demonstrate that the potential application of the tool on live network traffic.

The algorithm used in this tool is the Fuzzy ART as described in Section 4.2. When building a profile, the user selects a target host he or she wishes to monitor. The values of vigilance and learning rate used in Fuzzy ART algorithm can also be changed from the defaults. In this prototype, a group of training datasets are listed, representing various host computers' behaviors. The profile of a host computer's normal behavior is displayed in a bar chart (Figure 7). The bar chart shows the number of patterns and the distribution of the percentages of records in the dataset for each of the patterns of the behavior profile.

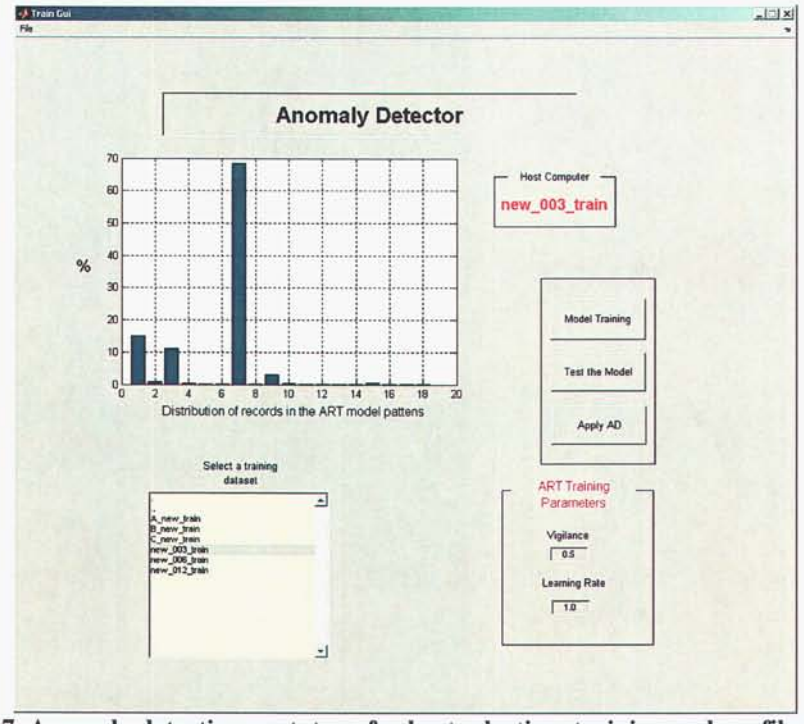

Figure 7. Anomaly detection prototype for host selection, training and profile display.

The user has the options to test the profile (model) or to use the trained profile in anomaly detection. If the Test the Model button is selected, a model testing GUI will display. A group of test datasets is listed for selecting. Testing results are displayed numerically in three categories: percentage of records in the test dataset that can be classified into the patterns of the trained model, the Spearmen coefficient and the cumulative value of the differences in records distribution in the patterns. In addition, a double bar graph visually shows the distribution of records in the patterns of the profile under the normal behavior and the one for testing; and another graph demonstrates the absolute differences (positive or negative) of records distribution in each pattern of the trained profile and the test datasets (Figure 8). If the user is not satisfied with the trained profile, this GUI gives the user the option to retrain the profile: reselect the vigilance and learning rate values.

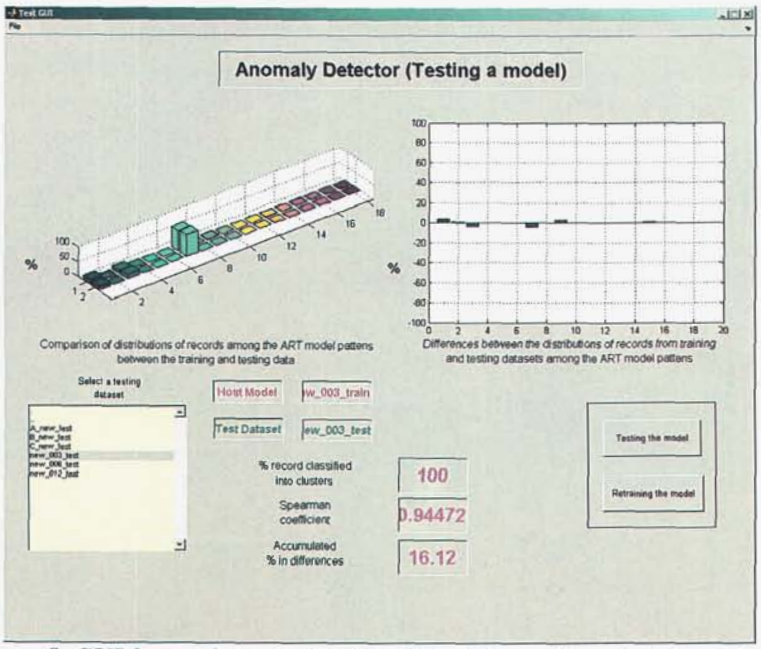

Figure 8. GUI for testing a trained profile, with profile retraining option.

A model validation GUI was developed to use a dataset from the host computer but with different records than those used for training and testing the model. The feature of the host computer behavior is known: normal or abnormal. This GUI is to validate whether the profile is capable of distinguishing between them (Figure 9). A profile retraining option is given if the profile needs to be modified. If the profile is not validated, the user can choose to retrain the profile.

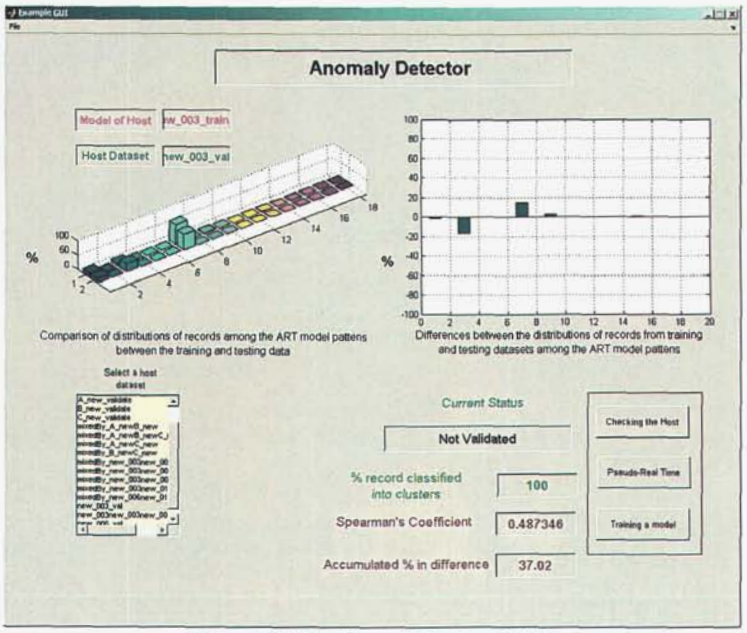

Figure 9. Profile validation GUI.

This GUI can also be used to check the normal/abnormal behavior for a host computer in a time period for which the data is saved in an individual dataset, if the user accepts the profile as the normal behavior profile for the host. The user can select a dataset and then press the "Checking the Host" button, the normal behavior will be reported as "Validated" in the current status report area, In Figure 10, an anomaly detection is performed with the trained profile (new_003_train) against the dataset (new_003_1000). "Validated" indicates that the host computer's behavior is normal during the time period covered by the dataset.

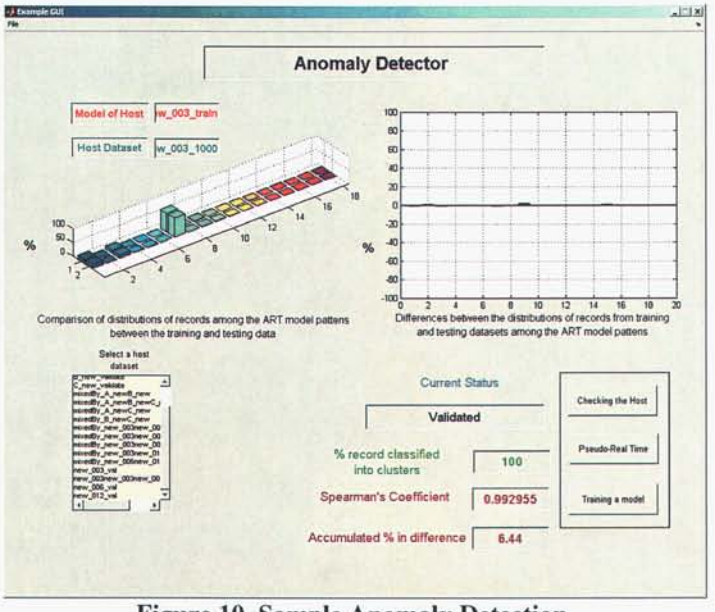

Figure 10. Sample Anomaly Detection.

The function called "Pseudo-Real Time" demonstrates the concept of applying the tool to real time or near real time on-line data. If it is selected, the trained profile will be continuously and automatically used to test against the datasets that represent the real-time or near real-time behavior of the host. The GUI will report the results when the detection process is in progress by updating the fields of graphs, current status, and the three numerical categories, as shown in Figure 11. A list window will display the historical status of the host computer's behavior along with the time period and three numerical categories.

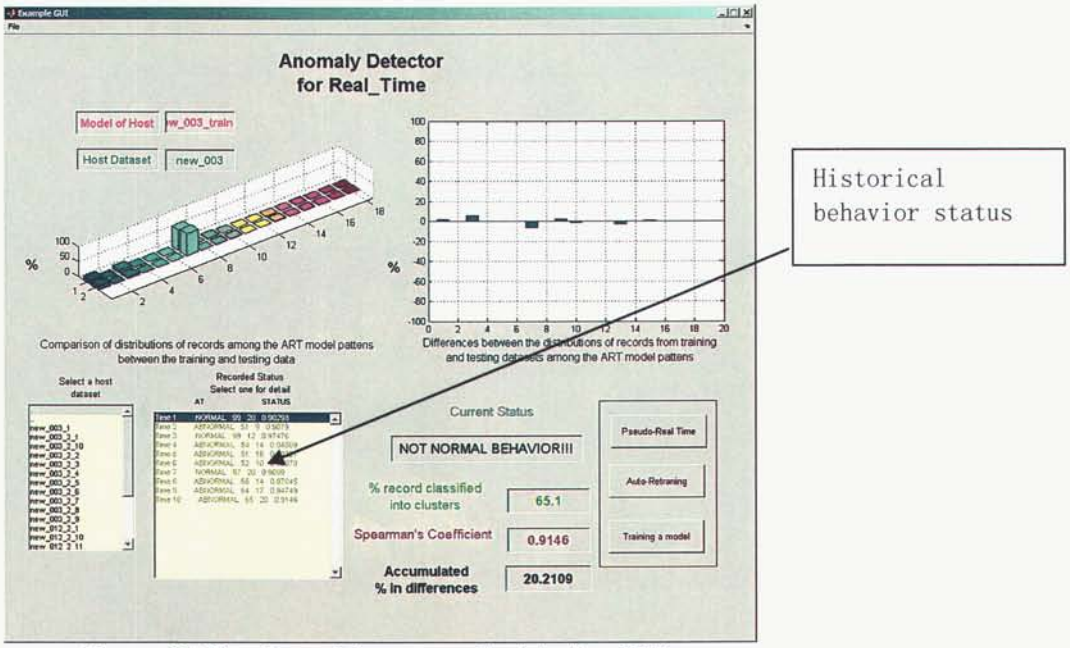

Figure 11. Pseudo real-time anomaly detection GUI.

The above GUIs were developed with the MATLAB interface development tools. The user interaction functionalities included in the MATLAB GUI tools may be limited as compared with tools, such as Java Swing. A better, more user-friendly GUI could be developed if the concepts outlined in the MATLAB prototype were incorporated into a more sophisticated GUI environment.

This page intentionally left blank.

6 Anomaly Detection Results

6.1 Normal vs. Abnormal Behaviors of a Host Computer

The behavior profile used in the prototype was trained with the data collected under normal operation conditions. If the behavior of the host computer does not change, the data collected at any time should construct a profile that should be similar to that with normal behavior. However, if the host computer operates differently, e.g., changes in the characteristics of its normal behavior, the prototype will issue an alarm. To demonstrate this, the following two graphs (Figure 12) are obtained by using the same normal behavior profile to compare against different datasets. The left diagram uses the data obtained from the same host computer (Host 003) without significant behavior change. Visually, one can observe that two groups of bars, representing the profiles from training and testing datasets, or representing the behaviors under normal operation conditions and the behavior to be tested, are almost identical. The differences of each pattern between the two profiles are very small. In contrast, the right one was obtained when the dataset obtained from a different host computer (Host 012) which has a different application from the one used to train the profile. This dataset for Host 12 is assumed (for purposes of this example) to be from the same host computer (Host 3), which has changed its normal behavior. From the left double bar graph, the distributions of patterns in the two profiles are not the same and several significant differences of the individual patterns between the two profiles are found from the graph at the right side. Numerically, only 1.75% of the records that can be distributed into the patterns of the trained profile, the sum of differences is large, 158, and the Spearman Coefficient is far away from 1. Therefore, this tested computer behavior is different from the normal behavior, an anomaly is detected, and the 'Not Validated' in the field of Current Status indicates the abnormal behavior.

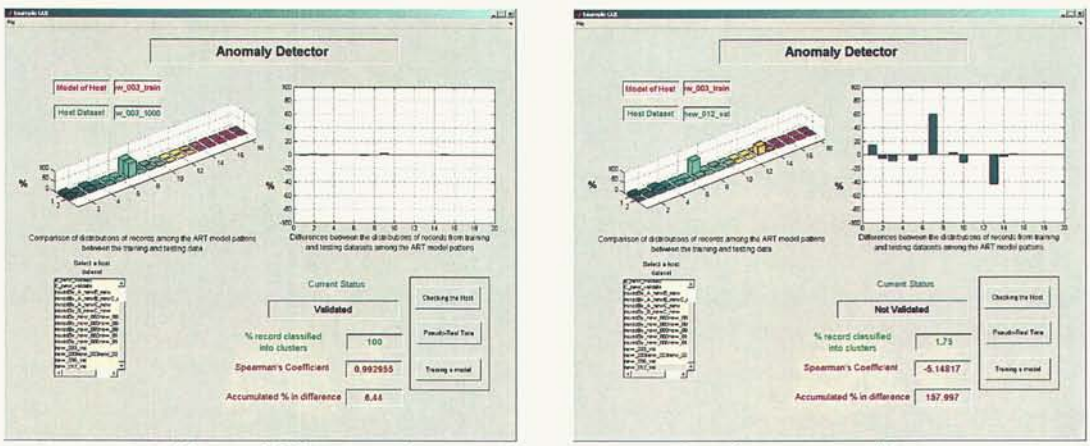

Figure 12. Datasets from two hosts compared to the same profile.

6.2 Effect of category values on anomaly detection

As described previously, determination of an abnormal behavior depends upon three categories: Percentage of Passed Records (PPR), the sum of differences between distributions of records in the tested profile against the profile obtained under normal operation (SDP) and the Spearman's Coefficient (SC). However, the challenge is determining the values assigned to the three factors that can sensitively and correctly detect an anomaly from normal behavior, and thus maintain a low rate of both false positives and false negatives. In principle, with a high of PPR, a low SDP and a SC close to 1, the prototype will have a high sensitivity to distinguish abnormal from normal behaviors. Because the profile built with Fuzzy ART is approximate-based, data in a pattern are similar but not necessarily same. A degree of tolerance for data deviation has to be allowed when the prototype classifies the records to its patterns. If PPR is set to 100%, SDP to 0 and SC to 1, the prototype will essentially recognize every group of records as abnormal if it is not identical or very similar to the training dataset. On the other hand, if the settings are too far from their ideal values, the prototype is not able to distinguish the abnormal from normal.

Choosing the thresholds for the three values should be based on a number of experimental results using data from normal and abnormal behaviors. The objective of this project is to develop an algorithm and prototype that can sensitively determine any deviation from the normal behavior. The adjustment of the sensitivity and correctness of the prototype will be left to future users because the optimal values will likely be host and network-specific. Detailed experiments were not conducted to determine the correlation between the values of the three factors and the sensitivity of the prototype in anomaly detection.

The following two examples demonstrate the effect of the thresholds of the three factors in determining whether an anomaly will be detected. The behaviors of the host computer during this time period were classified as 'Abnormal' by using three sets of combined values used to determine the 'normal/abnormal' behavior:

1. PPR >= 90 and SDP <= 30 and SC >= 0.7
2. PPR >= 90 and SDP <= 80 and SC >= 0.9
3. PPR >= 90 and SDP <= 50 and SC >= 0.8

If the evaluated result for a particular dataset can't satisfy any of the above threshold sets, then behavior is classified as 'Abnormal'.

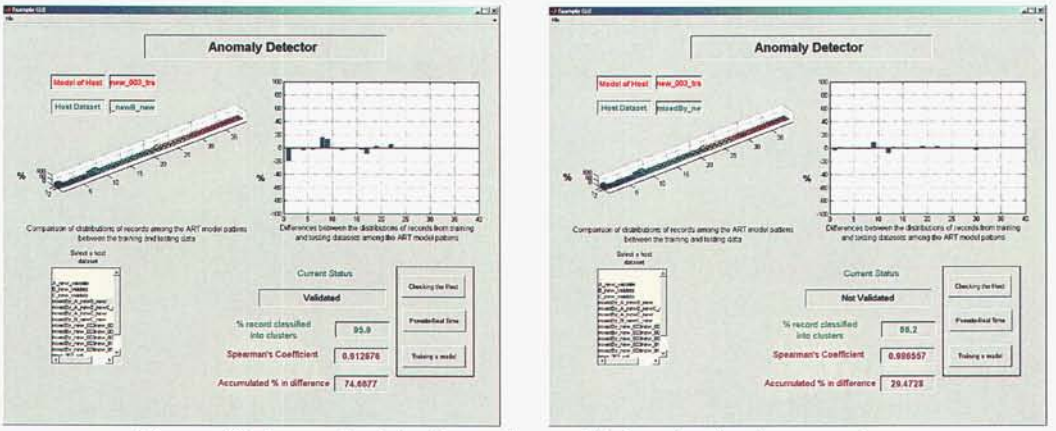

Figure 13. Anomaly detection using multiple sets of category values.

Figure 13 illustrates the behavior difference when two datasets are compared against a trained profile and the 'Normal/Abnormal' behavior is determined by three sets of preset category values. The values of PPR, SDP and SC at the left hand graph are 95.9, 74.67 and 0.91, respectively, which satisfy the second set. It is classified as a 'normal' behavior. The values for the right hand graph are 65, 60.43 and 0.98, respectively. None of above three set of values can be satisfied; even though two out of three values in the right hand side graph are better than that in the left hand side, it is still considered as 'Abnormal'. This example demonstrates the importance of rationally setting the values for the three factors in evaluate the host computer's behavior.

6.3 Effect of the Resolution of a Profile

The resolution of a profile can be thought of as the number of patterns in a profile. If the resolution is high, the individual records in a pattern share a high degree of similarity, as compared with a lower resolution profile.

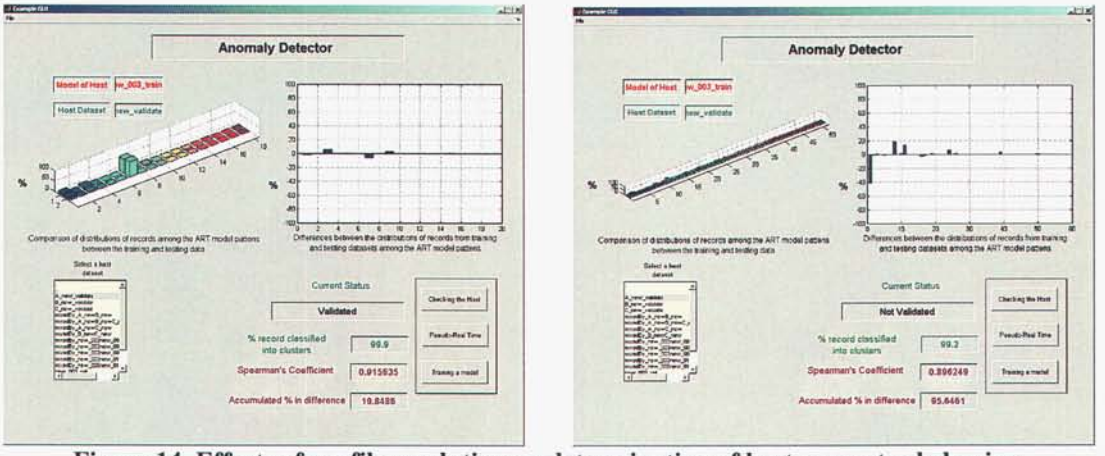

Figure 14. Effects of profile resolution on determination of host computer behavior.

The effects of the profile resolution on the determination of the host computer behavior are illustrated in Figure 14. Here, the same host dataset is evaluated using two profiles that were trained with the same dataset, but with values of vigilance of 0.5 and 0.75,

respectively. When the vigilance of 0.5 is used, the trained profile (left) contains 18 clusters. In this case the dataset to be tested shows small differences from the trained profile and there is no anomaly detected. When the vigilance is increased to 0.75, a profile with 52 clusters is generated. The same dataset used previously for 0.5 is tested against the trained profile and classified as 'Abnormal'. Obviously, the clusters in the high resolution profile contain the records with higher similarity as comparing with those in the low resolution profile. The sensitivity of the profile in anomaly detection increases with the increase of the profile resolution, but so does the noise level. A proper selection of vigilance, thus, is important to the effectiveness and quality of the anomaly detector and the selection is system specific and can only be determined by a series of experiments under various scenarios: normal/abnormal behaviors and/or with/without intrusion or internally abuse and etc.

6.4 Real-time Application

The ultimate goal of developing this prototype is to apply anomaly detection techniques to detecting the internal and external threats in real-time or near real-time, to timely prevent or reduce the damage of those threats. Keeping this in mind, a feature was developed that demonstrates the concept of real-time application. In the following example shown in Figure 15, the prototype continuously and automatically evaluates the incoming data and reports the host computer's behavior status and also keeps the records of the behaviors. Note that both the current and previous host status is displayed. In this prototype, data that was collected at different time periods is stored in the datasets. However, an interface can be built to allow the prototype to directly interact with host computer and collect data using an appropriate protocol.

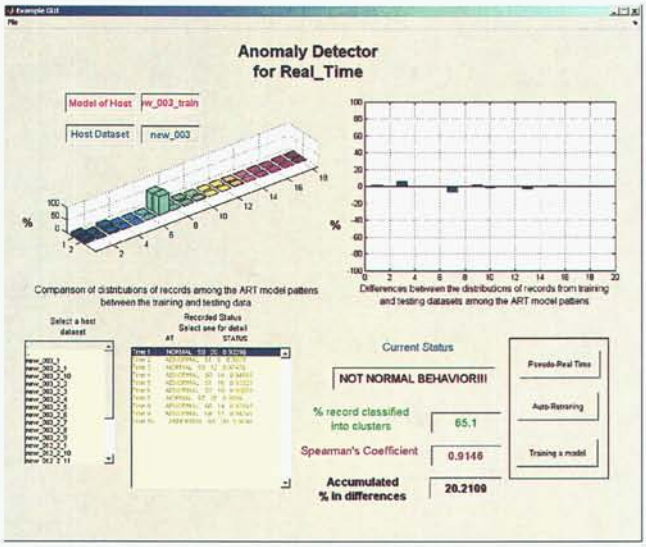

Figure 15. Example of real-time application of the anomaly detection prototype.

7 Summary and Discussion

Application of anomaly detection is relatively new to network security. Although there has been much research in the last decade to develop various approaches for intrusion detection and virus prevention, there is no a workable system based on anomaly detection for network security.

The approaches we reported here focus on a technique that uses of self-organizing and unsupervised learning, because of the difficult nature of evolving attacks and insider threats. In order to establish the behavior-based profile, the high-level session data, only containing a limited number of attributes, were used to avoid sheltering effects from the large quantity of attributes from the low level network traffic data.

After evaluating several data-mining algorithms, the Kohonen self-organizing map (SOM) and the fuzzy adaptive resonance theory (Fuzzy ART) algorithms were evaluated in detail and a prototype of anomaly detector based on Fuzzy ART was developed.

Both SOM and Fuzzy ART algorithms show promise in detecting network abnormal behaviors. The sensitivity of Fuzzy ART seems much higher as compared with that of SOM.

The algorithms and prototype of Fuzzy ART anomaly detection developed in this research are not only limited to being applied to network traffic data, but are ready to be used to evaluate the similarity and difference of datasets that feature a large volume of data in high dimensions. Although not reported here, this prototype has been applied to other non-network traffic datasets and successfully demonstrated its capability for anomaly detection in different systems.

The results from this research are preliminary and a number of issues remaining to be addressed:

We have demonstrated the capability of the algorithms in detecting anomalies, but we have not tested datasets that contains real intrusions or used it to identify abnormal behaviors that pose a threat to the network security;

The selection of criteria and their values used for Fuzzy ART algorithm to measure the behavior of a host computer is still somewhat arbitrary. More research is needed to select the highly representative criteria for measuring the differences between two profiles. These areas would be a good focus for future anomaly detection research.z

This page intentionally left blank.

8 References

Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford, England: Oxford University Press, 1995.

Burges, C.J.C. 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery. 2(2): 121-168

Carpenter, G.A., S. Grossberg, and D.B. Rosen. 1991. Fuzzy Art: Fast stable leaning and categorization of analog patterns by an adaptive resonance system. Neural Networks, V.4:759-771.

Carpenter, G.A. and S. Grossberg. 1987. A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Session, Graphics, and Image Processing, 37:54-115

Dunham, M.H. 2002. Data Mining: Introductory and Advanced Topics. Prentice Hall PTR, Upper Saddle River, NJ.

Durgin, N., Mai, Y., Van Randwyk, J., NetState: A Network Version Tracking System, Proceedings of Freenix/Open Source Track 2005 USENIX Annual Technical Conference, Anaheim, CA, April 10-15, 2005, pages 119-127.

Huang, J., M. Georgiopoulos and G.L. Heileman.. 1995. Fuzzy ART properties. Neural Networks, V8(2):203-213.

Kohonen, T. 2001. Self-Organizing Maps. the Third Edition. Springer

Liao Y., V.R. Vemuri, and A. Pasos. 2004. A general framework for adaptive anomaly detection with evolving connectionist systems. in Proc. of 2004 SIAM International Conference on Datamining. Lake Buena Vista, Florida. April 2004.

Lichodzijewski, P., A. Zincir-Heywood, and M. Heywood. Dynamic intrusion detection using self organizing maps. In 14th Annual Canadian Information Technology Security Symp., May 2002.

Maxion, R.A. and T.N. Townsend. Masquerade detection using truncated command lines. In Proc. of International Conference of Dependable Systems and Networks. Washington DC, June, 2002, pp 219-228.

Orengo, C.A., D. T. Jones, J. M. Thornton, (eds.) 2002. **Bioinformatics**: BIOS Scientific Publishers. Washington, DC

Rhodes, B.C., J.A. Mahaffey, and J.D. Cannady. 2000. Multiple self-organizing maps for intrusion detection. Proceedings of the 23rd National Information Systems.

Warrender, C. S. Forrest, and B. Pearlmutter. 1999. Detecting intrusions using system calls: Alternative data models. In Proc. of IEEE Symposium on Security and Privacy. Okland, CA. May 1999, pp 133-145.

Distribution

1	MS 0455	G. Conrad, 5631
1	MS 0455	W. Cook, 5631
1	MS 0455	S. Rinaldi, 5633
1	MS 0455	R. Tamashiro, 5632
1	MS 0455	P. Zhang, 5631
1	MS 0630	B. Hess, 4610
1	MS 0671	R. Parks, 5612
1	MS 0671	R. Trellue, 5610
1	MS 0785	R. Hutchinson, 5616
1	MS 0975	P. Warner, 4317
1	MS 9011	N. Durgin, 8941
1	MS 9011	J. Howard, 8941
1	MS 9011	J. Van Randwyk, 8941
1	MS 9151	C. Oien, 8940
1	MS 9151	L. Napolitano, 8900

2	MS 9018	Central Technical Files, 8945-1
2	MS 0899	Technical Library, 4536
1	MS 0188	D. Chavez, LDRD Office, 1011

This page intentionally left blank.