A Hybrid Approach to the Profile Creation and Intrusion Detection

Jack Marin, Daniel Ragsdale, and John Surdu
Information Technology and Operations Center, United States Military Academy
{jack-marin | daniel-ragsdale | john-surdu}@usma.edu

Abstract

Anomaly detection involves characterizing the behaviors of individuals or systems and recognizing behavior that is outside the norm. This paper describes some preliminary results concerning the robustness and generalization capabilities of machine learning methods in creating user profiles based on the selection and subsequent classification of command line arguments. We base our method on the belief that legitimate users can be classified into categories based on the percentage of commands they use in a specified period. The hybrid approach we employ begins with the application of expert rules to reduce the dimensionality of the data, followed by an initial clustering of the data and subsequent refinement of the cluster locations using a competitive network called Learning Vector Quantization. Since Learning Vector Quantization is a nearest neighbor classifier, and new record presented to the network that lies outside a specified distance is classified as a masquerader. Thus, this system does not require anomalous records to be included in the training set.

1. Introduction

Intrusion detection may be defined as "the problem of identifying individuals who are using a computer system without authorization (i.e., 'crackers') and those who have legitimate access to the system but are abusing their privileges (i.e., the 'insider threat') [1]." Thus, intrusion detection approaches may be divided into anomaly detection systems and misuse detection systems, or some combination of the two. Misuse detection involves identifying patterns of known "bad" behavior, while anomaly detection looks for patterns of activity that appear to be abnormal [2]. In this research, we concentrate on anomaly detection and describe a hybrid approach to establish profiles, assign legitimate users to those profiles, and subsequently identify individuals who attempt to masquerade as a legitimate user.

Our hybrid approach begins with an application of unsupervised clustering technique to assign legitimate users to clusters based on the frequency of commands used in an actual network environment. Next, we reduce the dimensionality of the data using a genetic algorithm, and finally, we refine the relationship of the legitimate users and clusters using a variation of a competitive neural network named Learning Vector Quantization (LVQ).

The remainder of this paper is organized as follows: Section 2 provides a review of other anomaly detection systems, to include both intelligent and statistical systems. Section 3 describes our hybrid approach, and section 4 provides results from a series of experiments. Section 5 contains a summary and areas of future research.

2. Background and Previous Work

The assumption that underlies all user-based anomaly detection schemes for intrusion detection is that intrusive behavior is, by its very nature, anomalous. Under such schemes, if it can be established that a given user is acting in an abnormal manner then the actions of that user (or someone who is masquerading as that user) can be classified as intrusive. In these approaches, behaviors can be determined to be abnormal through a comparison against a *user profile* that represents a user's typical behavior. This user profile, which can take on many forms, is based upon either an individual's behavior and/or the typical behavior of the individuals in a functional group. Some anomaly detection systems also maintain a model of typical system behavior. In these systems, a model of system performance metric is maintained. Any time that the system is not operating in a *normal* manner there is an increased likelihood that an intruder is (or was) present on the system.

Anomaly detection techniques have been applied to the problem of detecting intrusion since the research field of intrusion detection was first formalized with the publication of Anderson's seminal report in 1980 [3]. In 1987 Denning proposed the first comprehensive model of intrusion detection systems [4]. In this frequently cited model she includes *anomaly records* as one of the six basic components of a generic intrusion detection system. Since the publication of her model, intrusion detection researchers have applied a wide variety of methods to detect anomalous activity.

The earliest proposed methods for intrusion detection focused on the application of statistical methods to identify anomalous activity [5]. Many early systems [6, 7, 8, 9] employed this method. In addition, a number of on going projects [10, 11, 12, 13] continue to employ statistical methods for anomaly detection, typically in combination with other methods.

More recent anomaly detection methods employ a wide variety of classification schemes to identify anomalous activities. These schemes include, among others, rule induction [14, 15, 16], (artificial) neural networks [17, 18, 19], fuzzy set theory [20], *classical* machine learning algorithms [21, 22], artificial immune systems [23, 24], signal processing methods [25], and temporal sequence learning [26, 27].

A challenge that all developers of anomaly detection-based intrusion detection classifiers must address is feature selection/data reduction. Clearly, the inclusion of too much data will adversely impact the performance of the system, while the inclusion of too little data will reduce the overall effectiveness of the system. In addition, most anomaly detection approaches must address the problem of *conceptual drift* [28]. In this domain the problem of conceptual drift manifests itself in that a user's behavior changes over time. An effective anomaly-based intrusion detection system should adapt to this change while still recognizing intrusive actions and not adapting to those.

3. Hybrid Approach to Anomaly Detection

3.1. Introduction to the Hybrid Approach

3.1.1. The Data. Our approach to misuse detection is based on the simple premise that legitimate users, over time, will establish a profile based-on the number and types of commands they employ. Thus, if a system can discern this profile, and commands are employed "outside" of this profile, then the system should flag this action as a potential masquerade.

The data used to build the profiles in this research consists of 5,000 individual commands entered at a UNIX prompt for a set of 50 users [29]. Each user is designated as User1, User2, and so on, and the commands appear in the sequence in which they were entered. Command line arguments are not available, and there is no reporting of the time of day or time between the individual commands.

In order to create user profiles, we calculated the percentage of commands employed by a user for a specified total number of commands. Table 1 provides an example of the data for two users assuming commands are collected in groups of 1,000. (Note, since there are 5,000 commands per user, then in this example each user will be represented five times). We address the methodology used to select the appropriate number of commands, and which commands to exclude in subsequent sections.

3.1.2. Notation. We will use the following notation throughout this paper. A user is denoted with letter *x*, and is actually a vector representing the percentage of each command the user employed (note, in this paper, we will denote vectors with bold type). We consider the number of commands to be the dimension of the data, *d*, thus a complete set of data is a *c* x *d* matrix where *c* represents the number of users and *d* is the dimension of the commands. We assume a user can be classified as legitimate (ω_L) or a masquerader (ω_M), where $\Omega = \{\omega_L, \omega_M\}$. Thus, the goal of this research is to derive a decision rule, *d(x)* such that *d(x)*: $x \rightarrow \Omega$, which assigns users as legitimate or masquerader based solely on the number and types of commands. Note the class of legitimate users, ω_L, actually consists of many profiles associated with legitimate users. We denote the class of legitimate individual user profiles as $\omega_L = \{\omega_1, \omega_2, \ldots, \omega_n\}$ where *n* denotes the number of legitimate user profiles.

3.1.3. Methodology. We employed the following four-step methodology in building the legitimate user profiles. Each step will be subsequently explained in the following paragraphs.

Step 1: Reduce the dimensionality of the data using expert rules.

Step 2: Cluster the data using K-means.

Step 3: Further reduce the dimensionality of the data using a genetic algorithm.

Step 4: Refine the cluster locations using Learning Vector Quantization.

Table 1: Example data. The table entries represent the percentage a legitimate user employed a given command, based on groupings of 1,000 commands per user

	cpp	**xrdb**	**mkpts**	**env**	**csh**	**kill**
user1	1.97%	15.79%	11.54%	15.79%	17.51%	15.79%
user1	2.62%	21.05%	15.38%	21.05%	20.28%	21.05%
user1	5.25%	42.11%	26.92%	42.11%	34.56%	42.11%
user2	0.00%	0.00%	15.38%	0.00%	0.00%	0.00%
user2	14.10%	0.00%	7.69%	0.00%	0.00%	0.00%
user2	21.64%	0.00%	3.85%	0.00%	0.00%	0.00%
user2	24.59%	0.00%	0.00%	0.00%	0.00%	0.00%
user2	27.21%	0.00%	3.85%	0.00%	0.00%	0.00%

3.2. Dimension Reduction Using Expert Rules

To facilitate the clustering algorithm discussed in Part C, it was necessary to reduce the dimensionality of the data. The data included the number of times that 250 different commands were typed by each of the users. An expert system was used to eliminate 100 of these commands so that the commands that contained the most information about user profiling were retained. This expert system used a number of "fuzzy" rules.

Fuzzy set membership is a useful tool for capturing the semantic imprecision of human expert heuristics. Fuzzy logic is based on the notion that objects belong in sets. Unlike traditional set theory, however, the boundaries between sets are gradual, or fuzzy. The height of people provides a good example of fuzzy set membership. Clearly a seven-foot tall man would fit in the TALL set, and a four-foot tall man would fit in the SHORT set. In what set would a man be if he were five feet, six inches tall? Five feet, ten inches? Some people might regard him as short, while others would consider him tall. The boundary between TALL and SHORT is not crisp. An arbitrary, crisp boundary can be chosen, say five feet, ten inches. The problem with this crisp approach is that in many real-world applications, as the value of a parameter changes a very small amount but passes some crisp threshold, the behavior of the system can be radically different. Fuzzy set theory uses the notion of degree of membership in a set (a real number between 0 and 1, inclusive) to deal with imprecision or uncertainty. The degree of membership in the TALL set for a man who is five feet, six inches tall might be 0.5 [30]. The real benefit of fuzzy logic is that it provides a way to "develop cost-effective approximate solution to complex problems by exploiting the tolerance of imprecision." [31]. Fuzzy logic also provides a good way to help push through the knowledge acquisition bottleneck [32] associated with building expert systems. When building a fuzzy rule base, the knowledge engineer can allow the human expert to describe the problem in semantic terms, like "very fast," "somewhat fast," and "not fast."

In building the expert rules used to reduce the dimensionality of the data, it was important to retain any commands that are typically used frequently by *malicious masqueraders* but not as frequently by *normal* users. The commands that are used frequently by malicious masqueraders can be divided into four categories: information collection, remote access, file upload and download, and compilation support. Information collection commands include: *ps, find, ls, gethost, netstat, uname, ping, finger, which, head, tail.* Remote access commands include: *telnet, rlogin, and rsh*. File upload and download support commands include: *download, unpack, ftp*. Finally, compilation support commands include: *cpp, make, emacs-20, javac, cc1, gcc*. The expert system, therefore, contained rules to ensure that these commands were not removed. The remainder of the rules used a combination of easily computed statistics and fuzzy heuristics to remove other commands from the list.

The output of the application of these fuzzy rules is a reduced set of commands to be used in the clustering algorithm discussed in part C. Because this clustering relies on the fact that each element shares that same set of dimensions, the expert rules needed to analyze the information content of each command with respect to all users. In other words, a command was removed only if it provided little or no information about any of the users.

Several rules were used to screen out invalid commands. The data set included 5000 commands typed at the Unix prompt for each of 25 users. The data included commands that were mistyped, symbols that are not valid commands, and in some cases erroneous command-line arguments (e.g., "]" and "FALSE" were listed as commands. The difficulty in doing this was to ensure that commands that are not standard Unix commands but were valid, such as a user-written executable or local system scripts, were not eliminated.

Other rules involved making decisions about commands based on easily computed statistics. For the dimensionality reduction stop, the data for each user was grouped into groups of 1000 commands so that there were several instances of User 1, User 2, etc. For the data set used in this experiment, this resulted in five instances of each user as shown in Table 1. Then the mean of these five instances was computed. For example, for User 1, in Table 1, the mean of "cpp" is 2.49. In some cases, the standard deviation of these means was computed across all users. Given these simple statistics some rules used are shown in Table 2. Recall, however, that commands typically used by hackers but not by normal users are protected. With the sample data we used for this experiment, the expert rules reduce the number of commands from 250 to 150.

4. Classification Using Learning Vector Quantization

4.1. Theoretical Underpinnings of Profile Creation using LVQ

Learning Vector Quantization (LVQ) is a neurally inspired, nearest neighbor classifier based on Kohonen's work with self-organization [33, 34, 35]. Quantization can be defined as mapping a broad range of input values to a smaller number of output values. In LVQ, the input values can be thought of as the decision boundaries between a set of classes (i.e. ω_L), and the output values are a predetermined number of nodes or reference vectors. A reference vector is defined as $\mathbf{q}_r$, where $\mathbf{q}_r \in \Re^d$; r = 1,…,R, and a set of reference vectors is denoted as $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{q}_R\}$. If we divide the reference vectors into *n* groups, where *n* is the number of legitimate profiles then $(\omega_1, \ldots, \omega_n \in \omega_L)$, and $(\mathbf{q}_{11}, \ldots, \mathbf{q}_{1n1} \in \omega_1)$, $(\mathbf{q}_{21}, \ldots, \mathbf{q}_{2n2} \in \omega_2)$, and so on such that each reference vector is associated with a particular pattern class, or in this case legitimate user profile. The strategy behind LVQ is to effectively train the reference vectors to define the Bayes Optimal decision boundaries between the classes. Correctly positioning the reference vectors in LVQ is accomplished in a supervised manner by presenting a training pattern to an input vector and adjusting the position of selected reference vectors in accordance with a set of learning rules, as will be described later in this paper.

The training algorithms associated with LVQ attempt to adjust the position of the reference vectors so that each input pattern has a reference vector of the right category as its nearest neighbor. Classification, subsequently, is carried out using a nearest-neighbor method. Kohonen argues (35) that, asymptotically, the reference vectors approach the centroids of their resulting Voronoi tessellation. A Voronoi tessellation is a partition of $\Re^d$ into disjoint polytopes such that all training patterns within a polytope have the same reference vector as their nearest neighbor. Thus, the goal of LVQ is to approximate the boundaries of the Voronoi polytopes, therefore approximating the decision surfaces of a Bayesian classifier. Reference [36] provides a rigorous mathematical description of the LVQ process and proves convergence of the algorithm under certain asymptotic conditions.

Since LVQ relies on a nearest neighbor scheme for classification, our misuse detection system uses this to our advantage. Assuming the reference vectors are trained using the LVQ algorithm, then the minimum distance for a given user to the closest reference vector is defined as:

$$D_{w_L}(\mathbf{x}, \mathbf{q}_{w_L}) = \underset{All\ R w_L}{MIN} D(\mathbf{x}, \mathbf{q}_{wr})$$

where D(•) represents the Euclidean distance between a given reference vector and a user. In this system, if D_{wL} is greater than some distance D^* ($D_{wL} > D^*$), then the user exceeds the bounds of an accepted profile and is considered to be a potential masquerader. It is important to note this methodology does note require the generation of masquerader data. We create profiles and train the system based-on legitimate user data, and measure the deviation of subsequent users from these profiles. Thus, the problem of negative (masquerader) data is obviated.

4.2. The LVQ Algorithm

There are several versions of LVQ reported in the literature [37]. While each LVQ algorithm attempts to approximate the Bayes decision surfaces between classes, the learning rules associated with each LVQ algorithm are slightly different. In this initial research, we choose to use the original LVQ algorithm referred to as LVQ1. Future work will assess the impact of different LVQ algorithms on the classification process.

LVQ1 employs a perceptron-like learning law that rewards a correct classification, and punishes an incorrect classification. Specifically, denoting $\mathbf{q}_c(t)$ as the closest reference vector at iteration t, the learning law for LVQ1 is:

If the closest reference vector belongs to the same class as the input vector then:

$$\mathbf{q}_c(t+1) = \mathbf{q}_c(t) + a(t)(\mathbf{x}(t) - \mathbf{q}_c(t))$$

If the closest reference vector belongs to a different class then:

$$\mathbf{q}_c(t+1) = \mathbf{q}_c(t) - a(t)(\mathbf{x}(t) - \mathbf{q}_c(t))$$

For all other reference vectors:

$$\mathbf{q}_r(t+1) = \mathbf{q}_r(t) \quad \text{if } r \neq c$$

where $a(t)$ is a monotonically decreasing function of time, such that $0 < a(t) < 1$, however, usually $a(t) << 1$.

5. Design of Experiment

5.1. Generating Masquerader Data

In order to generate realistic masquerader data we took advantage of the fact that malicious masqueraders typically use particular command far more often than normal users. We identified 24 commands that would automatically be included in the database, and generated probability distributions for each commands based upon the characteristics of normal user's distributions. We then randomly selected records from the legitimate user database, and perturbed the actual number of some commands employed by the user by a random factor between 1 and 4 standard deviations. Due to the stochastic nature in which we generated masqueraders, some of these "anomalous" records more closely resembled legitimate users than others.

5.2. Parameter Values

Within this hybrid approach, there exist several key parameters that must be set by the user. For example, one key to the success of LVQ is the initial position of the reference vectors [35, 38]. Applications involving LVQ usually employ K-means clustering [39] as the basis for initially positioning the reference vectors. K-means requires the user to pre-determine the number of clusters. Other parameter values included the number of commands to include in a user record, the number of training iterations to use in the LVQ algorithm, and the distance threshold outside of which records would be classified as masqueraders.

6. Results

6.1. Common Test Parameter Settings

While we tested our system with many different parameter settings and various masquerader sets of data, the following tests are representative of our overall results. Note, for each test presented here, we used the following common parameter settings:

1. Commands in each user record: 100
2. Training sample size: 1875 records (75% of available records)
3. Legitimate user test sample size: 625 records (25% of available records)
4. Anomaly records created: 200
5. LVQ α value: .02
6. LVQ training iterations: 50,000

6.2. Test Results

For the first two tests displayed below, we employed a genetic algorithm with a multivariate linear regressive fitness function to reduce the number of commands from 164 to 30 key commands. We supplemented these 30 commands with 19 commands that would typically be used by a masquerader for a total of 49 commands. For our first test, we used the recommended rule of thumb of *"5 times the dimension"* [35], so we multiplied the number of commands by 5 and used 200 reference vectors in the K-Means and LVQ training process (note, actually the correct number of reference vectors would have been 49 x 5 = 245, however, we viewed 200 as an appropriate starting point).

6.2.1. Test 1:

For the first test, we used the following parameters: 49 user commands, 200 clusters, masquerader distance threshold = 1.7. We were able to detect 74.4% of the masquarders, which means we missed 25.6% of the anomalous behavior. However, we misclassified 33.5% of legitimate users as masquarders.

It is interesting to note, of the 64 anomalous records classified as legitimate, 44 of these records were classified into the same cluster or reference vector (reference vector 22). Only 14 legitimate users were classified into cluster 22. Thus it appears cluster 22 was unique among clusters and represented somewhat anomalous behavior.

6.2.2. Test 2

For the second test, we used the following parameters: 49 user commands, 100 clusters, masquerader we detected 71.6% of the masquarders (missing 28.4%) and the misclassification of legitimate users as masquarders was 38.1%.

Again, over half of the anomalous records misclassified as legitimate were closest to the same cluster or reference vector. This is curious because of the random nature in which we generated masqueraders would imply there would not exist a relationship between a large number of masqueraders and a single cluster. Also, we see a slight degradation in the classification accuracy attributed to the decrease in the number of clusters from test 1 to test 2.

6.2.3. Test 3

For the third test, we used the following parameters: 164 user commands, 395 clusters, masquerader distance threshold = 2.2. In this test we detected 68.5% of the masquarders, meaning we missed 31.5% of the anomalous behavior. We misclassified 44.6% of legitimate users as masquarders.

As expected, using all available commands did not improve the classification accuracy of this system. In fact, the system performed worse. We believe this is due to the comparatively small training set of 1875 records and the large number of clusters. Other experiments were conducted using all available commands and lesser numbers of clusters, however, the results did not improves significantly.

6.3. Analysis of Results

While on the surface our results may seem lackluster, we believe our approach merits future consideration. The anomalous detection problem and generation of profiles is an exceedingly difficult problem because of the purposely-vague definition of anomalous behavior. As stated earlier, we tested our system against a very high standard, and generated masquerader behavior that in some cases is almost indistinguishable for legitimate behavior.

Some parameter settings within this approach are robust while others are brittle. For example, the number of iterations used in the LVQ training process (plus or minus 5,000) had little effect on the results, while the exact determination of the distance threshold parameter differed in almost every test. However, given the training set, it is relatively easy to determine an appropriate threshold based on the classification of the training records.

We are presently conducting further research into the high false alarm rates. As described earlier, many of the false alarms were closest to a single reference vector, thus, it is possible the characteristics of some of the legitimate users are extremely close to masquarders, or possibly our method of generating masquarder data is flawed. We are conducting a statistical analysis of the masquarders that appear to be grouped together to see if there exists a statistical underpinning to this misclassification.

7. Conclusion

In this paper we described preliminary results that we achieved through the application of a variety of techniques to develop a hybrid anomaly detection methodology for intrusion detection. We first applied expert rules to reduce the dimensionality of the data. We then performed an initial clustering of the data using K-Means. Finally we refined the cluster locations using a competitive network called Learning Vector Quantization. Our initial results, while not especially impressive, indicated the approach proposed here shows promise and merits future consideration. Recall that we tested our system against a very high standard and, even so, we were able to achieve classification rates, in some cases near %80 with misclassification rates less than 20%.

In future research we refine this methodology in order to increase classification rates and decrease misclassification rates. We will attempt to do so through by adjusting the following parameters: number of commands in each user record, training sample size, legitimate user test sample size, number of anomaly records created, LVQ α value, and number of LVQ training iterations. In addition we will evaluate the degree to which this methodology can be generalized by applying it other data sets.

8. References

[1] Biswanath, M., Heberlein, T, and Levitt, K. “Network Intrusion Detection,” *IEEE Network*, 8, pp. 26-41, May/June, 1994.

[2] R. G. Bace, *Intrusion Detection*. MacMillan Technical Publishing, 2000.

[3] Anderson, J. P., "Computer Security Threat Monitoring and Surveillance," J.P Anderson Co., Fort Washington, Pennsylvania, Report Number 79F296400, April 15 1980.

[4] Denning, D. E., "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. 13, pp. 222-232, 1987.

[5] Helman, P. and Liepins, G. E., "Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse," IEEE Transactions on Software Engineering, vol. 19, pp. 886-901, 1993.

[6] Javitz, H. S. and Valdes, A., "The SRI IDES statistical anomaly detector," presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1991.

[7] Vaccaro, H. S. and Liepins, G. E., "Detection of Anomalous Computer Session Activity," presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Location TBD, 1989

[8] Lankewicz, L. and Benard, M., "Real-time Anomaly detection Using a Nonparametric Pattern Recognition Approach," presented at Proceedings of the of 7th Computer Security Applications conf., San Antonio, TX, 1991

[9] Anderson, D., Lunt, T. F., Javitz, H., Tamura, A., and Valdes, A., "Detecting Unusual Program Behavior Using the Statistical Components of NIDES," SRI International, Menlo Park, CA, Tech Report SRI-CSL-95-06, May 1995.

[10] Endler, D., "Intrusion detection Applying machine learning to Solaris audit data," presented at Proceedings of the Computer Security Applications Conference, 1998

[11] Jou, Y. F., Gong, F., Sargor, C., Wu, S. F., Chang, H. C., and Wang, F., "Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure," presented at DARPA Information Survivability Conference and Exposition, Hilton Head Island, SC, 2000.

[12] Porras, P. A., "EMERALD Network Intrusion Detection Project Description,", vol. 1999, 1999.

[13] Neumann, P. G. and Porras, P. A., "Experience with EMERALD to Date," presented at 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, 1999.

[14] Teng, H. S., Chen, K., and Lu, S. C., "Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns," presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1990.

[15] Habra, N., Charlier, B. L., Mounji, A., and Mathieu, I., "ASAX: Software Architecture and Rule-based Language for Universal Audit Trail Analysis," presented at Proceedings of the European Symposium on Research in Computer Security, Brighton, England, 1992.

[16] Helmer, G., Wong, J., Vasant Honavar, A., and Mille, L., " Intelligent Agents for Intrusion Detection and Countermeasures," presented at IEEE Information Technology Confe rence, Syracuse, NY, 1998.

[17] Lee, S. C. and Heinbuch, D. V., "Training a Neural-network Based Intrusion Detector to Recognize Novel Attacks," presented at IEEE Workshop Information Assurance and Security, West Point, NY, 2000.

[18] Ryan, J., Lin, M.J., and Miikkulainen, R., "Intrusion Detection with Neural Networks," presented at Proceedings of the 10th Advances in Neural Information Processing Systems Conference, Denver, CO, 1998.

[19] Rhodes, B. C., Mahaffey, J. A., and Cannady, J. D., "Multiple Self-Organizing Maps for Intrusion Detection," presented at Proceedings of the 23rd National Information Systems Security Conference, Baltimore, MD, 2000.

[20] Lin, T. Y., "Anomaly Detection - A Soft Computing Approach," presented at New Security Paradigms Workshop, Little Comp ton, Rhode Island, 1994.

[21] Lee, W., Nimbalkar, R. A., Yee, K. K., Patil, S. B., Desai, P. H., Tran, T. T., and Stolfo, S. J., "A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions," in Recent Advances in Intrusion Detection (RAID 2000), Third International Workshop, Toulouse, France, October 2-4, 2000, vol. Vol. 1907, H. Debar, L. Mé, and S. F. Wu, Eds. Berlin: Springer-Verlag, 2000, pp. 49-65.

[22] Lee, W. and Stolfo, S. J., "A Framework for Constructing Features and Models for Intrusion Detection Systems," ACM Transactions on Information and System Security, vol. 3, November, 2000

[23] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A., "A Sense of Self for Unix Processes," presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1996.

[24] Forrest, S., Hofmeyr, S. A., and Somayaji, A., "Computer Immunology," Communications of the ACM, vol. 40, pp. 88-96, 1997.

[25] Thottan, M. and Ji, C., "Proactive Anomaly Detection Using Distributed Intelligent Agents," IEEE Network, vol. 12, pp. 21-27, 1998.

[26] Lane, T. and Brodley, C. E., "Temporal sequence learning and data reduction for anomaly detection," ACM Transactions on Information and System Security, vol. 2, pp. 295-331, 1999.

[27] Kosoresow, A. P. and Hofmeyr, S. A., "Intrusion Detection via System Call Traces," IEEE Software, vol. 14, pp. 24-42, 1997.

[28] Schlimmer, J. C., "Concept Acquisition Through Representational Adjustment," in Department of Information and Computer Science. Irvine: University of California, 1987

[29] Data assembled by AT&T Labs, in conjunction with Rutgers University, http://www.research.att.com/~schonlau/intrusion.html, 1999.

[30] Henkind, S.J. and Harrison, M.C., "Analysis of Four Uncertainty Calculi," IEEE Transaction son Systems, Man, and Cybernetics, vol. 18, no. 5, May 1989, pp. 700-714.

[31] Yen, J. and Lengari, R., Fuzzy Logic: Intelligence, Control and Information, New York: Prentice Hall, 1999.

[32] Giarratno, J. and Riley, G., Expert Systems: Principles and Programming, Boston, MA: PWS-Kent Publishing Co., 1989

[33] Kohonen, T. Learning Vector Quantization for Pattern Recognition. Technical Report, TKK-F-A601, University of Technology, Helsinki, 1986.

[34] Kohonen, T. Self-Organization and Associative Memory. Springer-Verlag, Berlin, 1987.

[35] Kohonen, T., "The Self-Organizing Map," in Lau, C. (ed.) Neural Networks: Theoretical Foundations and Analysis. IEEE Press, NY, 1992.

[36] LaVigna, A., Nonparametric Classification Using Learning Vector Quantization. Ph. D. Thesis, University of Maryland, 1989.

[37] Kohonen, T., G. Barna, and R. Chrisley, "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," IEEE International Conference on Neural Networks, San Diego, CA, pp. 61-68, 1988.

[38] Flotzinger, D., J. Kalcher, and G. Pfurtscheller, "EEG Classification by Learning Vector Classification." *Biomedizinische Technik* 37, 303-309, 1992.

[39] Anderberg, M.R. *Cluster Analysis for Applications*. Academic Press, NY, 1973.