

# User Profiling for Computer Security

David L. Pepyne, Member, IEEE, Jinghua Hu, and Weibo Gong, Fellow, IEEE

**Abstract**—Systems and control theory was largely responsible for realizing the Industrial Revolution. We believe that systems and control theory will also play a key role in the Information Revolution underway. In particular, systems and control theory should have much to offer network security. This paper explores the use of queueing theory and logistic regression modeling methods for profiling computer users based on simple temporal aspects of their behavior. The goal is to develop profiles for very specialized groups of users, who because of the nature of their work would be expected to use their computers in a very similar and regular way, e.g., bank tellers, insurance adjusters, etc. Encouraging proof-of-concept results suggest the potential of both the idea of user profiling and the use of simple temporal features for grouping users.

## I. INTRODUCTION

THERE is no question that the systems sciences has resulted in very successful technologies for controlling all manner of electromechanical systems and played a major role in realizing the Industrial Revolution. We believe that the systems sciences will also have much to offer for controlling the information technologies that drive the current Information Age.

In recent years there have been many special sessions at the major systems conferences (e.g., CDC, ACC, IFAC) describing applications of systems and control theory to the *performance optimization* of information (computer) networks. In this paper we explore the application of systems and control theory to *network security*, a much more difficult problem than performance optimization.

In computer networks, malicious software programs can spread over the network to “infect” the vulnerable

This work was supported by grants from the U.S. Army Research Office (contract DAAD19-01-1-0610), the U.S. Air Force Office of Scientific Research (contract F49620-01-1-0288), and the U.S. Justice Department, Office of Justice Programs (contract 2000-DT-CX-K001). All views expressed belong to the authors and do not necessarily represent the official views of the U.S. Army, U.S. Air Force, or the U.S. Justice Department.

David L. Pepyne is with the Division of Engineering and Applied Science, Harvard University, Cambridge, MA 02138 USA (phone: 617-495-8911; e-mail: pepyne@hrl.harvard.edu).

Jinghua Hu is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 (e-mail: jhu@ecs.umass.edu).

Weibo Gong is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 (e-mail: gong@ecs.umass.edu).

computers attached to it. Nearly every day we hear about, or become victim to, a new Internet worm or email virus. In fact, even as this paper is being written the SoBigF and Blaster worms continue to wreak havoc on the Internet. In addition to malicious software, we also know that there are malicious people attempting to “hack” into private computers and private computer networks. It’s unclear how many of these attacks succeed because most organizations (particularly commercial ones) don’t publicly report successful penetration for fear of alarming customers [2].

In other work, we have explored the use of feedback control theory for controlling the spread of Internet worms [19], [20], [11]. This would reduce the number of computers that become infected by malicious software. However, a comprehensive defense-in-depth security system still needs some way to detect and mitigate the effects of malicious software and malicious hackers that are able to penetrate a networked computer. Moreover, in addition to attacks from the “outside”, there is the even more serious problem of “insider” attack, where malicious or corrupt people who have legitimate inside access to a computer cause damage or steal important data. It is estimated by many computer security professionals that insider attack is a much more serious threat than outsider penetration. Insider attack is especially problematic in industries such as banking, insurance, and credit cards, where *identity theft* is reaching epidemic proportions (an estimated 33 million people have been affected so far [17]). Again, fearing customer alarm, only a small fraction of these incidents ever become public.

Detecting penetration from the outside is the goal of *intrusion detection systems* (IDS) and detecting malicious software and malicious people working on the inside is the goal of *misuse detection systems* (MDS). Broadly speaking, IDS/MDS are divided into signature-based approaches that model the behavior associated with specific known attacks and then look specifically for matches to the attack models. The alternative is to model a user’s normal behavior and look for anomalous departures from the model. Most IDS/MDS strive to be general purpose, able to detect attacks and anomalies in any population of computer users and software applications. This is clearly a very hard problem and there are some that argue that in trying to live up to this grand challenge IDS/MDS research

is losing practical relevance [3].

Rather than propose a general purpose IDS/MDS, we think that much more success can be achieved by focusing on detecting anomalies in specific classes of computer users who have very well defined roles. Motivated by the growing problem of identity theft, we propose in particular to focus on detecting anomalous behavior by users such as bank tellers, insurance claims adjusters, and credit card transaction processors. These are the employees who are the lowest paid and thus have the most to gain from stealing other people's financial identities. On the other hand, because of the nature of these jobs, the computer activities of the people assigned to them should be expected to have a high degree of statistical similarity, i.e., because all bank tellers are doing the same tasks (handling clients at the window), they should all behave in essentially the same way. For detecting anomalous behavior from bank tellers and other employees involved in such assembly line transaction processing, this paper explores two new concepts for IDS/MDS: the concept of user *profiling* and the concept of queueing system *busy period structure* for selecting the features used for building models of user profiles. For the actual model building process, we use the statistical analysis tool *logistic regression* [18].

A *profile* is a crude model portraying only the significant features common to a specific *group* of users. With profiling, users (or intruders masquerading as them) would be monitored, not by comparing their behavior against models of known attacks or a model of their own past behavior, but rather by comparing their behavior to their group profile, any deviation from which would be cause for closer scrutiny.

The idea of using queueing system *busy period structure* for feature selection comes from the idea that users such as bank tellers can be viewed as parallel servers in a queueing system. Customers arrive and choose a teller who services their transaction. Since customer transactions are statistically very similar and if we assume the tellers are roughly equally utilized, then we could also expect that the *sample paths* (the pattern of *busy periods* and *idle periods*) generated by the various tellers should be statistically similar as well. Our idea then is to profile users such as bank tellers using models constructed from features related to the duration of the idle and busy periods and rough measures of activity levels during busy periods.

A key difficulty in computer security research can be getting suitable data [2]. However, unable to convince a bank to let us monitor their tellers and customers, and unable to use the standard data sets commonly used for IDS/MDS studies because they lack the time-stamp data that we need, we performed a preliminary proof-of-concept study using data collected from a number of graduate students and their professor. Of course the behaviors of

such a population would be expected to differ greatly from that of users such as bank tellers. Even so, because they shared common projects, took classes together, and worked similar projects in the same laboratory, there were some natural groups that our method was able to identify. We were also able to detect structural breaks (statistically significant changes in behavior) that occurred for example during the Winter intersession when the students were not taking classes. Overall our results corresponded well with what we expected to find based on visual analysis of the raw data and our knowledge of our user population. In future work we plan to conduct similar experiments using actual bank data to assess the efficacy of the idea in the real-world setting.

The remainder of the paper is organized as follows. The next section motivates the use of profiling and queueing theory for detecting intrusion and insider misbehavior and gives a summary of related work where temporal behaviors were used in the IDS-MDS setting. Section III defines the temporal features that we used and describes our data collection method. Section III also contains plots of some raw data that we will use to visually confirm the results we obtain. Section IV describes the logistic regression method that we use for building models of user profiles. Experimental results are summarized in Section V. The paper ends in Section VI with a discussion and conclusions.

## II. MOTIVATION AND RELATED WORK

### A. Motivation

Politically correct or not, *profiling* can be very effective for detecting anomalous behavior worthy of deeper examination. For example, groups of employees doing the same job should have very similar behaviors. In particular, bank tellers and data entry clerks processing insurance claims and doing credit checks, should have very similar behaviors simply because the transactions they work on are generally very similar. As a result, one might expect, for example, a *bank teller profile* to emerge. The activities on computer accounts belonging to tellers could then be compared to this profile of the "stereotypical" teller for anomalous behavior that might suggest misuse by an insider or intrusion by an outsider.

Roles like bank tellers are customer service roles. That is, tellers that behave like *servers in a queueing system*. A queueing system can be characterized by its *sample path*, which describes when the server is *idle*, when it is *busy*, and the *activity level* during busy periods. Viewed as servers in a queueing system, one can conjecture that tellers should generate *sample paths* that are statistically similar, i.e., there should be some "prototypical" sample path against which tellers can be statistically compared. This thinking is the motivation behind building models of user

behavior from temporal (time-related) features related to *when* someone uses their computer, for *how long* in one session, and roughly *how active* they are during a session. We also remark that in an actual bank setting, teller activities could also be correlated with activities in the bank lobby, such as the arrival of a customer at the teller's window. The view of a teller as a server in a queueing system makes it clear that these two events should have strong positive correlation.

### B. Related Work

Although most IDS/MDS attempt to classify users and/or application programs by parsing the operating system commands that they generate, there is also a relatively large literature on the use of temporal features in IDS/MDS. In his comprehensive study of intrusion detection methods, Denning [4] describes methods that use interval timers and event counters to build time-series models. NIDES [1], developed by SRI, uses both rule-based and statistical methods for intrusion detection. The statistical models constructed are used to compare short-term behavior with long-term historical norms. Senelyzov [14], [16], [15] proposed a temporal event algebra and a temporal probabilistic network to handle the temporal regularities in audit data. Lee [9] used data mining techniques to develop an anomaly detection method that learns association rules incorporating both command names and command timing. Li, et al. [10] describe a method that learns temporal association rules for different levels of time granularity, an idea that provides more flexible and precise descriptions than fixed time partitions. An approach for building temporal signatures for software programs from the mean and variance of command interarrival times is developed in [7]. Other work includes [8], where time signatures were used for database security.

The main difference between the work in this paper and the work cited above is that the other work tends to focus only on the temporal behavior during busy periods (the *intra-session behavior*), whereas our work also considers the temporal behavior between busy periods (the *inter-session behavior*). In other words, we focus on the entire *busy period structure* of the sample paths generated by the users, while the other work focuses only on the busy periods, largely ignoring the idle periods. Thus, our profiler would immediately flag an anomaly if a teller account is active late at night since the profile would indicate that the account should be idle when the bank is closed. Unless specifically set up to do so, the other methods described above may not flag after hours activity as unusual.

## III. EXPERIMENTAL SETUP

### A. Temporal Feature Selection

As described, our goal in this paper is to develop models

for the typical behavior of groups of users, such as bank tellers, whose role makes them effectively servers in a queueing system. The behavior of servers in queueing systems is characterized by the *sample paths* they generate, i.e., by the *busy period structure* describing when the server is *idle*, when it is *busy*, and *how active* it is when busy. With this in mind, we begin with the following definition of a *session* (busy period),

**Definition 3.1:** A *session* begins with the onset of user activity and ends when the user logs out or after 1800 seconds (3 minutes) of inactivity.

For each session we generate a single 6-dimensional vector consisting of the following temporal features,

$$X = \{\text{interval, length, output, density, timing.day, timing.hour}\} \quad (1)$$

where,

*interval* = the time elapsed since the end of the previous session;

*length* = the duration of the current session;

*output* = the number of operating system commands generated during the session;

*density* = the mean command rate (in commands/minute) during the session;

*timing.day* = an integer indicating the day of the week when the session began;

*timing.hour* = an integer indicating the hour of the day when the session began.

In other words, *interval* captures the length of idle periods, *length* the length of busy periods, *density* the activity level during busy periods, and *timing* relates busy periods to the daily cycles of human and business activity.

### B. Data Set

A major challenge in computer security is getting good data [2]. In particular, we found it very difficult to convince a bank to let us collect data from their network. We were also not able to use any of the standard data sets commonly used for IDS-MDS studies (e.g., the AT&T data set [13] and University of Calgary data set [6]), because they lack the time-stamp information that we needed.

Lacking suitable data, we collected our own data set for these proof-of-concept studies. Our data came from the Linux command histories (csh/tcsh shell command histories) from a number of Univ. of Massachusetts (Amherst) graduate students and their professor. This data was collected over a period of 15 weeks during the 2001-2002 school year. This 15 week period covered part of the 2001 Fall semester, the Winter intersession, and part of the 2002 Spring semester. We observed that the student's computer use behaviors during the Fall and Spring semesters, when they were taking classes, were very similar. And as expected, their behaviors during the Winter intersession, when they were not taking classes, was

much different. One of our goals was to understand how this structural break would impact the model building process.

### C. Visual Data Analysis

Figs. 1(a) and (b) show some of the raw data collected for this study. The plots in Fig. 1(a) show the session data for 8 different users for the month of Feb. 2002 (Spring semester). In the plots, the horizontal axis is the time of day (24 hour clock), and the vertical axis the day of the month. The dark rectangles show when the users were active (busy periods). The plots in Fig. 1(b) show histograms of the session densities of the mean command rate in commands per hour (activity level during busy periods). We will refer back to these plots to visually confirm the results obtained later in the paper.

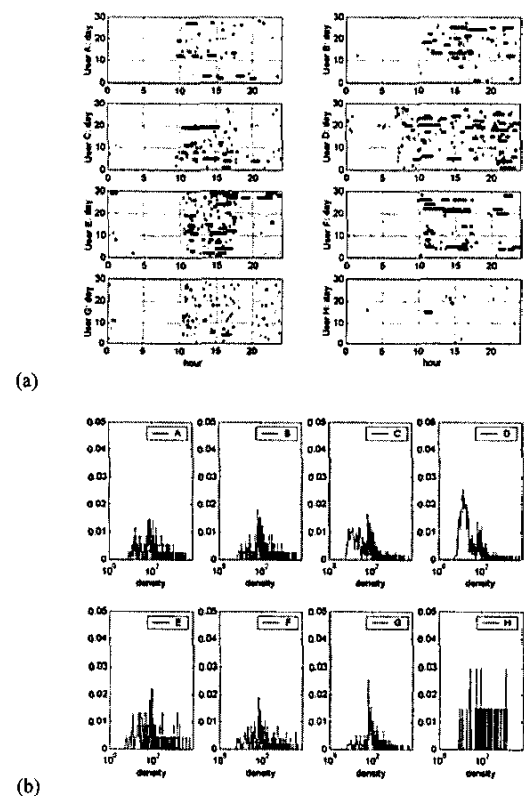


Fig. 1. Raw user data for visual analysis of experimental results.

## IV. MODEL BUILDING

### A. Logistic Regression

Logistic regression is a powerful statistics tool for determining how multiple factors affect a discrete output [18]. In the context of user profiling, the factors are the components of the feature vector of observed user behavior in Eq. (1) and the output is the profile group that most likely generated the feature vector, e.g., either the feature

vector came from profile  $i$  or it came from profile  $j$ . While logistic regression can be extended to analyze multiple choice data (see [18]), let us explain its use here in terms of binary classification. Specifically, let us use logistic regression to build models  $M_{ij}$ , where  $i, j \in \{A, B, \dots, H\}$  with A, B, ..., H the labels assigned to the 8 users in Fig. 1. For a given feature vector  $X$  the output of model  $M_{ij}$  corresponds to one of two hypothesis,

- $H_0$  = feature vector  $X$  came from user  $i$  (null hypothesis),
- $H_1$  = feature vector  $X$  came from user  $j$  (alternative hypothesis).

A binary logistic regression model has the form,

$$\log[p/(1-p)] = c_0 + CX = c_0 + c_1x_1 + c_2x_2 + \dots + c_6x_6 \quad (2)$$

where the input  $X$  is the 6-dimensional feature vector from Eq. (1),  $c_0$  and  $C$  are the model coefficients, and the output  $p$  is the probability that  $H_0$  is true (i.e.,  $p = \Pr[H_0 = \text{true}]$ ). Decisions are made by comparing the output against a threshold parameter  $0 < \theta < 1$  according to the rule,

$$\text{If } p > \theta, \text{ accept } H_0; \text{ otherwise, accept } H_1. \quad (3)$$

The model coefficients  $c_0$ ,  $C$  are obtained using Maximum Likelihood Estimation (MLE).

**Remark:** With a little back of the envelope calculation, we can rewrite Eq. (2) as,

$$p = 1/(1+e^{-y}), y = c_0 + CX = c_0 + c_1x_1 + c_2x_2 + \dots + c_6x_6 \quad (4)$$

Fig. 2 shows this equation in “block diagram” form. The astute reader will immediately recognize that the logistic model has precisely the same form as a “neuron” in an artificial neural network (ANN) [5]. In fact, instead of logistic regression, we could use a feedforward ANN in conjunction with some standard ANN training rule (e.g., error back-propagation). In doing so, however, we would lose the advantages of logistic regression. In particular, logistic regression is statistically rigorous, the results quantifiable, and the coefficients statistically meaningful (for details, see [18] and the references therein). For the ANN, in contrast, it is often very difficult to interpret the meaning of the network’s coefficients.

### B. Training, Testing, and Validation

Recalling Section III, our data set was collected over a 15 week period. The feature vectors collected during the first 5 weeks were used for training, and the feature vectors from the remaining 10 weeks were used for testing and cross validation. The model coefficients were computed using the logistic regression calculation webpage at [12]. The models were validated using the Chi-Square statistic and various other tests of model accuracy. Lacking space, we will not present the testing and validation results here.

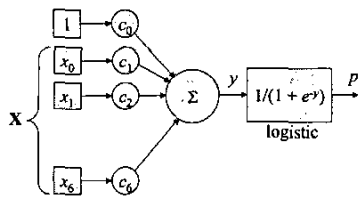


Fig. 2. Block diagram of a logistic regression classifier.

## V. PERFORMANCE EVALUATION

### A. The Receiver Operating Characteristic

As explained in Section IV, our models  $M_{ij}$  classify a feature vector  $X$  (Eq. 1) as belonging to profile  $i$  if the model output  $p > \theta$ , and as belonging to profile  $j$  otherwise. Such a binary classifier is evaluated by its Type I and Type II error probabilities, defined in Fig. 3.

	truth	
	$H_0$ is true	$H_1$ is true
decision		
Accept $H_0$	correct	
Accept $H_1$		correct

Fig. 3. Type I and Type II error probabilities.

In general, the Type I and Type II error probabilities depend on the value of the threshold parameter  $\theta$ . In particular, if we set  $\theta = 0$ , then the model accepts  $H_0$  as true for any input  $X$ , in which case the *miss* probability is 100%. Alternatively, if we set  $\theta = 1$ , then the model always accepts  $H_1$  as true, in which case the *false alarm* probability is 100%. Varying  $\theta$  between 0 and 1 traces out the so-called *receiver operating characteristic* (ROC), see Fig. 4. In the figure  $P_f(\theta)$  is the *false alarm probability* and  $P_m(\theta)$  is the *miss probability*.

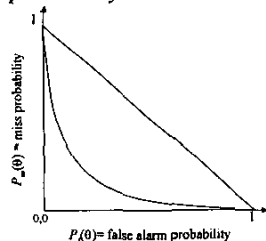


Fig. 4. Hypothetical receiver operating characteristic (ROC).

A classifier with a diagonal ROC gives the same performance as simple random guess; a classifier with an ROC lying entirely *above* the diagonal performs worse than random guess; and a classifier with an ROC that lies entirely *below* the diagonal performs better than random guess. The closer to the origin the ROC is, the better the guess.

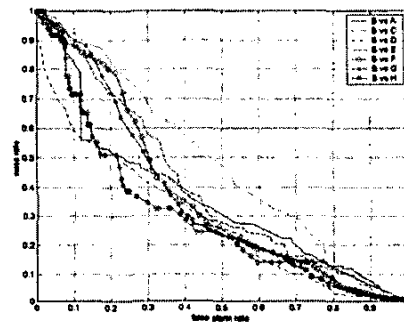
In the context of user profiling,  $M_{ij}$  for two users belonging to the same profile group should have a near diagonal ROC, i.e., the users are indistinguishable. Users belonging to different profile groups should have an “off-diagonal” ROC, i.e., they should be easy to distinguish. Similarly, models of intruders and misbehaving insiders should deviate from their “normal” group profile model.

### B. Summary of Main Results

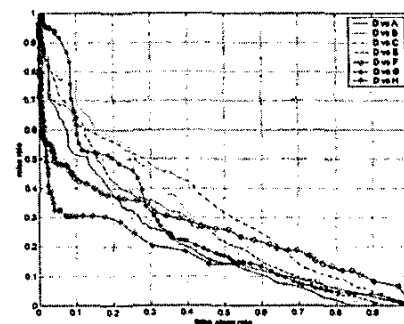
Figs. 5(a)-(b) show the experimentally obtained ROC curves for selected pairs of users. Fig. 5(a), which compares user B with the 7 other users in Fig. 1, shows that the temporal behavior profile of user B is most similar to that of user C (as suggested by a near diagonal ROC), and most different from that of user H (as suggested by an ROC that is much closer to the origin). Visual examination of Fig. 1 bears this out.

Fig. 5(b), which compares user D with the 7 other users in Fig. 1, shows that user D’s profile is not similar to any of the other 7 users, i.e., D’s profile can be relatively reliably identified. A visual inspection of Fig. 1 reveals that this is likely because user D typically starts his work day much earlier than everyone else.

These and our other results provide good evidence that the simple temporal features from Eq. (1) can be useful for building profiles for users like bank tellers whose computer use can be expected to be much more regular than that of graduate engineering students and engineering professors.



(a)



(b)

Fig. 5. ROC curves for selected pairs of users.

### C. Additional Results

To determine if our approach could detect a change in a user’s profile we did a study looking for structural breaks. Structural breaks occur when a user’s behavior changes in some statistically significant way. As we had expected, we observed a major structural break in the behavior of our user population of during the Winter intersession. The significance of our ability to detect this structural break is

an indicator that the temporal features in Eq. (1) may be able to detect the behavioral changes that accompany misuse or intrusion, e.g., a bank teller's account suddenly shows activity at night, on weekends, or at activity levels much different than usual.

## VI. DISCUSSION AND CONCLUSIONS

We believe that the concepts, tools, and techniques developed by the systems and control community has much to offer for computer security. In this paper we demonstrated a proof-of-concept idea on the use of profiling, queuing theory, and logistic regression modeling for intrusion and misuse detection in a very specific target class of computer users, e.g., bank tellers, and insurance and credit processing agents. However, similar temporal regularities also likely exist in government offices like the social security office, IRS, and so forth. Our results, albeit preliminary, were encouraging, suggesting that monitoring even very crude behavioral features might be effective in detecting intrusions and misuse in these specific classes of computer users.

Our key innovation is simplicity. Instead of digging deep into the OS commands users generate, we instead simply look at the *busy period structure* of the sample path they generate, i.e., *when they are busy, how long they are busy, and roughly how active they are while busy*. This leads to a very fast and simple implementation, requiring only simple time-stamping and counting. Being so simple and fast, another potential application, outside of user profiling, is for making a quick first pass over audit data, either throughout the day, or to aid forensic investigation after an attack by locating traces for further more detailed examination. We are also the first (as far as we know) to demonstrate the usefulness of the powerful statistical logistic regression technique in the IDS-MDS setting. Logistic regression is very similar to ANNs, which are sometimes used for IDS-MDS, but is statistically more rigorous and amenable to detailed statistical analysis of model quality and sensitivity to various features.

The approach presented has certain limitations. First, it only explores a very limited portion of the space of possible behavior features. Thus, it is not designed to be used standalone to detect computer abuse, but as another "sensor" in an integrated, comprehensive defense-in-depth security system. For example, the behavior of a bank teller can be correlated with customer arrivals to the bank's lobby. Another limitation is that the approach does not provide real-time detection, since it only makes its decisions when a session ends—although by simple modification one can imagine a system that dynamically comes to a decision as features reveal themselves, e.g., while the length of the session is not known until it ends, the starting time, day, and interval since the last session are

known immediately at the session start.

The main open question to be addressed is to evaluate the usefulness of the approach in building profiles for groups of users who because of the nature of their work are expected to have similar behaviors. To do these experiments we are seeking user data from industries such as banking and insurance and from government offices.

## REFERENCES

- [1] D. Anderson, T.G. Lunt, H. Javitz, A. Tamaru, and A. Valdes, "Detecting Unusual Program Behavior using the Statistical Components of the Next-Generation Intrusion Detection Expert System (NIDES)," *Computer Science Laboratory*, SRI-CSL-95-06, May 1995.
- [2] R. Clandos, "Eye on Cybercrime," *IEEE Security & Privacy Magazine*, Vol. 1, No. 4, July/August 2003.
- [3] G. Cybenko, "Boiling Frogs," *IEEE Security & Privacy Magazine*, Vol. 1, No. 4, pg. 5, July/August 2003.
- [4] D. Denning, "An Intrusion Detection Model," *IEEE Trans. on Software Engineering*, Vol. 13, No. 2, pp. 222-232, 1987.
- [5] L. Fausett, *Fundamentals of Neural Networks*: Prentice Hall, 1994.
- [6] S. Greenberg, "Using UNIX: Collected Traces of 168 Users," Research Report 88-333-45, Dept. of Computer Science, University of Calgary, Calgary, Canada, 1988.
- [7] A. Jones and S. Li, "Temporal Signatures for Intrusion Detection," 17<sup>th</sup> Annual Computer Security Applications Conference (ACSAC'01), New Orleans, LA, December 2001.
- [8] V.C.S. Lee, J.A. Stankovic, and S.H. Son, "Intrusion Detection in Real-Time Database Systems via Time Signatures," *Proc. of the 6<sup>th</sup> IEEE Real-Time Technology and Applications Symposium (RTAS'01)*, 2001.
- [9] W. Lee and S.J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Trans. on Information and System Security*, Vol. 3, No. 4, 2000.
- [10] Y. Li, N. Wu, X. Wang, and S. Jajodia, "Enhancing Profiles for Anomaly Detection using Time Regularities," *Proc. of the 1<sup>st</sup> ACM Workshop on Intrusion Detection Systems*, Athens, Greece, 2000.
- [11] D. Pepyne, W. Gong, and Y. Ho, "Modeling and Simulation for Network Vulnerability Assessment," presented at the 40th U.S. Army Ops. Research Symposium (AORS), Fort Lee, VA, October 9-11, 2001.
- [12] J.C. Pezzullo, *Logistic Regression Calculating Page*, <http://members.aol.com/johnp71/logistic.html>.
- [13] M. Schonlau, "Masquerading User Data," <http://www.schonlau.net/intrusion.html>, 1998.
- [14] A. Seleznyov, O. Mázhelis, and S. Puuronen, "Learning Temporal Regularities of User Behavior for Anomaly Detection," *International Workshop MMM-ACNS*, St. Petersburg, Russia, 2001.
- [15] A. Seleznyov and S. Puuronen, "Anomaly Intrusion Detection Systems: Handling Temporal Relations between Events," *Recent Advances in Intrusion Detection (RAID'99)*, 1999.
- [16] A. Seleznyov, T. Terziyam, and S. Puuronen, "Temporal Probabilistic Network Approach for Anomaly Intrusion Detection," *1st Conf. on Comp. Security Incident Handling and Response*, 2000.
- [17] "Stop Thieves from Stealing You," *Consumer Reports*, pp. 12-17, October 2003.
- [18] J. Whitehead, "An Introduction to Logistic Regression," <http://personal.ecu.edu/whiteheadj/data/logit/>.
- [19] C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and Early Warning for Internet Worms," *University of Massachusetts, Elec. and Comp. Eng. Tech. Rpt TR-CSE-03-01*, 2003.
- [20] C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proceedings of CCS'02*, Washington D.C., November, 2002.