NISS

A Hybrid High-order Markov Chain Model for Computer Intrusion Detection

W-H Ju and Y. Vardi

Technical Report Number 92
February, 1999

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

A Hybrid High-order Markov Chain Model for Computer Intrusion Detection

Wen-Hua Ju and Yehuda Vardi

Dept. of Statistics, Rutgers University

Abstract

A hybrid model based mostly on a high-order Markov chain and occasionally on an independence model is proposed for profiling the command-sequence of a computer user in order to identify a "signature behavior" for that user. Based on the model, an estimation procedure for such a signature behavior driven by Maximum Likelihood (ML) considerations is devised. The formal ML estimates are numerically intractable, but the ML-optimization problem can be substituted by a linear inverse problem with positivity constraints (LININPOS), for which the EM algorithm can be used as an equation solver to produce an approximate ML-estimate. A user's command-sequence is then compared to his and others' estimated signature-behavior in real time, by means of statistical hypothesis testing. A form of the likelihood-ratio test is used to test if a given sequence of commands is from the proclaimed user, with the alternative hypothesis being masquerader user. Data from a real-life experiment, conducted at a research lab, is used to assess the method.

Key Words: Anomaly Detection; Unix; Mixture Transition Distribution (MTD); LININPOS; EM.

1 Introduction

Computer - and network - intrusions are top priority security issues for nowadays computer systems. An intrusion-detection system flags intrusive behavior by monitoring dynamic usage patterns of the system in search for abnormal patterns. A lot of efforts have been put on developing intrusion

detection systems, including commercial products like NetRanger from CISCO (NetRanger 1998) and NIDES from Computer Science Laboratory, SRI International (Anderson, Frivold and Valdes (1995)), and also academic projects like COAST project at Purdue University (Balasubramaniyan et al. 1998) and Computer Immune Systems at University of New Mexico (Forrest et al. 1996). There are two main approaches to intrusion detection: *misuse detection* and *anomaly detection*. The misuse detection approach is to collect a large database of intrusion signatures and use it as a reference to monitor the current computer (mis)use. This implicitly assume prior knowledge of the nature of the intrusion and consequently runs the risk of failing to detect new types of attacks. The anomaly detection approach is to build profiles of normal usage patterns based on historical records, and flagging newly observed patterns which deviate from the "average" past-profiles possible intrusions. The focus of this paper is on the latter approach.

A Markov chain model is considered for profiling the UNIX command sequence of a computer user in order to identify a "signature behavior" for that user. To increase the model-flexibility, a "high-order" Markov structure is assumed, which takes into account the last few commands (rather than the last single command) in order to determine the next command. The underlying rational behind this approach is that although the command sequence of any specific user is random, it generally follows a probabilistic pattern which might be captured by a sufficiently rich Markov model. A simple Markov chain, where the next command depends solely on the current one, seems too crude for such an application, and so high-order model, in which the next command depends on the recent history, say of last three commands seems more appropriate, as a modeling tool. The problem is that this approach would lead to a very high-dimension parameter-space. In a universe of, say, 500 commands a Markov transition probability between two commands will have 249,500 ($= 500^2 - 500$) parameters. A higher order Markov model, say of order 2, where a pair of consecutive

transitions constitute a single state in the chain, will have 124,750,000 $(= 500^2(500-1))$parameters, and, in general, the parameter-space dimensionality is $K^l(K-1)$ if K is the number of distinct commands and l is the index-order of the Markov chain. Our basic approach to overcoming the dimension problem inherent to the high-order Markov chain, has two key components : (a) restrict attention to a subset of the most used commands (with the remaining commands grouped together under a single "command" labeled other), (b) use a *Mixture Transition Distribution* (MTD), Raftery (1985) and Raftery and Tavaré (1994), approach to model the transition probabilities. The combined two steps reduce the dimensionality of the parameter space to a practical level for which we can apply statistical methods of estimation and hypotheses testing.

A major difficulty of the Markov modeling is that users' command patterns could change drastically in time and consequently, it could be that a user's test data contain new commands which were not observed in user's training data. The transition probabilities involving such new commands could not be estimated from a fixed training data, and any Markov model would be on a shaky ground. In such instances we resort to an alternative approach based on an independence model from cross classifying users vs. commands in a simple contingency table.

The paper proceeds as follows: In section 2 we describe the design of the experiment, and the data for testing the methodology. In section 3 we describe the statistical models and estimation procedure. A hypothesis testing procedure is developed in section 4 in order to flag out "abnormal behavior", which are interpreted as potential masquerader. An algorithm for updating the estimated parameters in real-time, as command-sequence data flows in, is described in section 5. The experiment's results are given in section 6, and a summary-discussion in section 7.

2 Data and Experimental Design

The data are collected from the output of the UNIX acct auditing mechanism on a local machine at a research lab, user names and the associated sequences of commands (without arguments) are used in the experiment. The data contain about seventy users, and for each of the users there are at least 15,000 consecutive commands. Fifty out of the eighty users are randomly selected to form a "user community", each of them has a string of 15,000 consecutive commands. Considering these commands as 150 blocks of 100 commands, the first 50 blocks of commands of each user are treated as this user's training data. Starting after block 50, some masquerading command blocks are randomly inserted to the command sequence for each of the 50 users in the community. These masquerading commands are randomly drawn from the 20 users outside the community.

The inserting rules are as follow: For each command block $B_i (i = 50, 51, \ldots, 150)$ of the 50 users, there is a probability 1% that some masquerading command blocks are inserted after it. The number of the inserted command blocks is randomly chosen according to the geometric distribution with mean 5 (pmf $f(x) = 0.2(0.8)^{x-1}, x = 1, 2, \ldots$). After the length are determined, we randomly choose a user and a start command block from the outside users. If there are not enough successive command blocks or if these command blocks are previously used, we randomly choose a user and a start command block again. After the random insertion is completed, the first 150 command blocks are used, with the first 50 as the training data and the last 100 as the test data. The purpose of our study is to develop a method to determine if a test command block is from the proclaimed user or from an outside masquerader.

3 Profiling Users Based On Training Data

We first describe the Markov part of our model (§ 2.1) followed by an independence model (§ 2.2) which is needed when the Markov model is not applicable. The latter typically happens in connection with the use of rare commands. Below we outline our parameter-estimation procedure for the two cases.

3.1 MTD Model and Parameter Estimation

We set up the high-order Markov chain model as a *Mixture Transition Distribution* (MTD) model, along the lines proposed in Raftery (1985) and Raftery and Tavaré (1994). With this model, an increase of one dimension in the index-order of the Markov chain amounts to adding a single parameter, so that the dimensionality of the parameter-space is linear with the index-ordering of the Markov chain. For any given user, Let K be the smallest number such that the most frequently used K-1 commands of that user account for at least 99% of his training data, and group all other commands as other. We then combine the most frequently used K-1 commands and other to get the Markov chain's state space, M. For example, suppose that a user's training data consist of 200 commands, including 100 sh, 50 ls, 40 cat, 8 sendmail, 1 rm and 1 date. Then $K = 5$ and $M = \{s_1$=sh, s_2=ls, s_3=cat, s_4=sendmail, s_5=other$\}$ where any command other than {sh, ls, cat, sendmail} is categorized as other.

Let $\{C_t; t = 1, 2, \ldots\}$ be the sequence of commands taking values in the state space $M = \{s_1, s_2, \ldots, s_K\}$. Under the MTD model, the transition probabilities of an l-th order Markov chain can be written as:

$$P(C_t = s_{i_0} | C_{t-1} = s_{i_1}, C_{t-2} = s_{i_2}, \ldots, C_{t-l} = s_{i_l}) = \sum_{j=1}^{l} \lambda_j \, r(s_{i_0} | s_{i_j}), \quad t = l+1, l+2, \ldots \qquad (1)$$

where $\mathbf{R} = \{r(s_i|s_j);\ i,j = 1,2,\ldots,K\}$ and $\mathbf{\Lambda} = \{\lambda_i;\ i = 1,2,\ldots,l\}$ satisfy

$$r(s_i|s_j) \geq 0, \quad i,j = 1,\ldots,K \quad \text{and} \quad \sum_{i=1}^{K} r(s_i|s_j) = 1, \qquad \forall j = 1,\ldots,K. \tag{2}$$

$$\lambda_i \geq 0, \quad i = 1,2,\ldots,l \quad \text{and} \quad \sum_{i=1}^{l} \lambda_i = 1 \tag{3}$$

The advantage of the MTD model is the reduction of the number of parameters, from $K^l(K-1)$ (conventional parameterization) to $K(K-1)+l-1$. For simplicity, we fix $l = 10$ for all users in our experiment, meaning, we consider for each user the history of the last 10 commands as potentially contributing to the transition probability of the next command. In particular, any MTD model of order less then or equal to 10 is covered by this model.

Because the set of commands being categorized as other varies from user to user, it's necessary to fix a universal standard for the transition probabilities involving rare commands, and we use the following rules for all users :

1. $r(\text{other}|s_i) = \epsilon$, $\forall i = 1,2,\ldots,K$, $\quad \epsilon$ is small $\quad$ (we choose $\epsilon = .00001$ in our experiment)

2. $r(s_i|\text{other}) = (1-\epsilon)/(K-1)$, $\forall i = 1,2,\ldots,K-1$.

The rationale behind these rules are (1): To force the probability of transition to other to be small, regardless of the individual definition of other. (2) To compensate for the scarcity of data for estimating transition probabilities from other to other commands. Note that other is a representation of rarely used commands. We choose $\epsilon = .00001$ according to 0.01 (1% chance to get a rare command) times .001 (about 1000 rare commands to choose from).

For each user, the log-likelihood of a command sequence $(c_1, c_2, \ldots, c_T)$ is

$$\log L(c_1,c_2,\ldots,c_T) = \sum_{i_0=1}^{K} \cdots \sum_{i_l=1}^{K} N(s_{i_0}, s_{i_1}, \ldots, s_{i_l}) \log \left(\sum_{j=1}^{l} \lambda_j r(s_{i_0}|s_{i_j}) \right) \tag{4}$$

where $N(s_{i_0}, s_{i_1}, \ldots, s_{i_l})$ is the number of times that the pattern $s_{i_l} \mapsto s_{i_{l-1}} \mapsto \ldots \mapsto s_{i_0}$ is observed in the command sequence and $M = \{s_1, s_2, \ldots, s_K = \mathsf{other}\}$ is the state space of commands. Maximum likelihood estimate (MLE) requires maximizing the expression on the right side of (4) with respect to $\mathbf{\Lambda}$ and $\mathbf{R}$ subject to the constraints in (2) and (3). Raftery and Tavaré (1994) suggest the direct maximization method using sequential quadratic programming algorithm to solve the optimization problem. In many cases, especially the one at hand, this method is too computationally demanding to be practical or implementable, because of the huge number of the variables to be included in the optimization.

To overcome this computational difficulty, We propose a method which approximately solve the ML-optimization problem and significantly reduces the computational effort. A key step is an iterative alternating maximization of the log-likelihood with respect to $\mathbf{\Lambda}$ and $\mathbf{R}$. This leads to global maximization because $\log L$ is concave in $\mathbf{\Lambda}$ and $\mathbf{R}$. For the part when $\mathbf{R}$ is fixed, we maximize $\log L$ with respect to $\mathbf{\Lambda}$. Since the number of elements in $\mathbf{\Lambda}$ is l (the order of the Markov chain) and typically small, the sequential quadratic programming algorithm can be applied to solve it (we use CFSQP v.2.5 as described in Lawrence, Zhou and Tits(1997)).

For the part when $\mathbf{\Lambda}$ is fixed, we approximate the optimization problem with a linear inverse problem subject to positivity constraints (LININPOS problem), and apply the EM algorithm to solve it (Vardi and Lee (1993)). Toward this goal, re-index the log-likelihood (4) as follows: Defines the map

$$\varphi : (i_0, i_1, \ldots, i_l) \longmapsto k, \quad \text{where } k = \varphi(i_0, \ldots, i_l) = 1 + \sum_{j=0}^{l} (i_j - 1) K^{l-j}.$$

which leads to

$$N(s_{i_0}, s_{i_1}, \ldots, s_{i_l}) \longrightarrow a_k \quad \text{and} \quad \sum_{j=1}^{l} \lambda_j r(s_{i_0} | s_{i_j}) \longmapsto b_k.$$

and the log-likelihood being of the form

$$\sum_k a_k \log b_k, \tag{5}$$

where

$$\sum_k a_k = \sum_{i_0=1}^{K} \cdots \sum_{i_l=1}^{K} N(s_{i_0}, s_{i_1}, \ldots, s_{i_l}) = T - l,$$

and

$$\begin{aligned}
\sum_k b_k &= \sum_{i_0=1}^{K} \cdots \sum_{i_l=1}^{K} \left(\sum_{j=1}^{l} \lambda_j r(s_{i_0}|s_{i_j}) \right) \\
&= \sum_{i_1=1}^{K} \cdots \sum_{i_l=1}^{K} \left(\sum_{j=1}^{l} \lambda_j \sum_{i_0=1}^{K} r(s_{i_0}|s_{i_j}) \right) \\
&= \sum_{i_1=1}^{K} \cdots \sum_{i_l=1}^{K} (\sum_{j=1}^{l} \lambda_j) \\
&= K^l.
\end{aligned}$$

Note that φ defines an one-to-one correspondence between $(i_0, \ldots, i_l)$ and k which allows us to change the multi-dimensional index to one-dimensional.

Now $\{a_k\}$ are given (data) and a simple Lagrange method argument shows that the log-likelihood (5) will be maximized when

$$\hat{b}_k = \frac{a_k}{\sum_k a_k} \sum_k b_k = \frac{a_k}{T-l} K^l, \quad \forall k$$

i.e.

$$\sum_{j=1}^{l} \lambda_j \, \hat{r}(s_{i_0}|s_{i_j}) = \frac{K^l}{T-l} N(s_{i_0}, s_{i_1}, \ldots, s_{i_l}), \quad \forall (i_0, \ldots, i_l). \tag{6}$$

Treating (6) as a linear system subject to the constraints (2) results in a LININPOS problem in $\{r(\cdot|\cdot)\}$ that can be "solved" (in the sense of a minimum Kullback-Leibler distance between the

left- and right-side of (6)) using the EM algorithm. Specifically, we have the linear system

$$\begin{pmatrix} \mathbf{A} \\ \\ \mathbf{B} \end{pmatrix} \mathbf{R} = \frac{K^l}{T-l} \begin{pmatrix} \mathbf{N} \\ \\ \mathbf{1} \end{pmatrix}$$

where

$$\begin{aligned} \mathbf{R}^T &= (r(s_1|s_1), r(s_2|s_1), \ldots, r(s_K|s_1), \ldots, r(s_1|s_K), \ldots, r(s_K|s_K)) \\ &= (r_1, r_2, \ldots, r_{K^2}). \end{aligned}$$

are the unknowns, $r(s_i|s_j) = r_{i+K(j-1)}$, and

$$\mathbf{N} = \begin{pmatrix} N(s_1, s_1, \ldots, s_1) \\ N(s_1, s_1, \ldots, s_2) \\ \vdots \\ N(s_1, s_1, \ldots, s_K) \\ \vdots \\ N(s_K, s_K, \ldots, s_1) \\ \vdots \\ N(s_K, s_K, \ldots, s_K) \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \\ \vdots \\ N_K \\ \vdots \\ N_{1-K+K^{l+1}} \\ \vdots \\ N_{K^{l+1}} \end{pmatrix}$$

where $N(s_{i_0}, s_{i_1}, \ldots, s_{i_l}) = N_i$, $i = \varphi(i_0, i_1, \ldots, i_l)$.

$$\mathbf{A} = \{a_{ij}\}_{K^{l+1} \times K^2}, \quad \text{where } a_{ij} = \sum_{k=0}^{l} \lambda_k I[j = i_0 + K(i_k - 1)], \ (i_0, \ldots, i_l) = \varphi^{-1}(i)$$

and looks like:

$$\mathbf{A} = \begin{pmatrix} \lambda_1 + \ldots + \lambda_l, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & 0 \\ \lambda_1 + \ldots + \lambda_{l-1}, & 0, \ldots, 0, & \lambda_l, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & 0 \\ \lambda_1 + \ldots + \lambda_{l-1}, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & \lambda_l, & 0, \ldots, 0, & 0 \\ & & \vdots & & & & \\ 0, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & 0, & 0, \ldots, 0, & \lambda_1 + \ldots + \lambda_l \end{pmatrix}_{K^{l+1} \times K^2}$$

$$\mathbf{B} = \begin{pmatrix} 1, \ldots, 1, & 0, \ldots, 0, & 0, \ldots, 0, & 0, \ldots, 0, \\ 0, \ldots, 0, & 1, \ldots, 1, & 0, \ldots, 0, & 0, \ldots, 0, \\ 0, \ldots, 0, & 0, \ldots, 0, & \ddots & 0, \ldots, 0, \\ 0, \ldots, 0, & 0, \ldots, 0, & 0, \ldots, 0, & 1, \ldots, 1, \end{pmatrix} = \{b_{ij}\}_{K \times K^2} \qquad \mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{K \times 1}$$

This can be solved iteratively using the following EM iteration step, with initial values for $\{r_j\}$ being strictly positive.

$$r_j \leftarrow \frac{a_{\cdot j}}{a_{\cdot j} + b_{\cdot j}} \hat{r_j}(\mathbf{A}, \mathbf{N}, \mathbf{R}) + \frac{b_{\cdot j}}{a_{\cdot j} + b_{\cdot j}} \hat{r_j}(\mathbf{B}, \mathbf{1}, \mathbf{R}) \tag{7}$$

where

$$\hat{r_j}(\mathbf{W}, \mathbf{u}, \mathbf{v}) \equiv \frac{v_j}{w_{\cdot j}} \sum_i \frac{w_{ij} u_i}{\sum_k w_{ik} v_k} \quad j = 1, 2, \ldots, K^2$$

for matrix $\mathbf{W} = \{w_{ij}\}$ and vectors $\mathbf{u} = \{u_i\}, \mathbf{v} = \{v_i\}$, and

$$\begin{aligned} a_{\cdot j} = \sum_i a_{ij} &= \sum_{(i_0, \ldots, i_l) = \varphi^{-1}(i)} \sum_{k=0}^{l} \lambda_k I[j = i_0 + K(i_k - 1)] \\ &= \sum_{k=0}^{l} \lambda_k \sum_{i_0} \ldots \sum_{i_l} I[j = i_0 + K(i_k - 1)] \\ &= \sum_{k=0}^{l} \lambda_k \cdot K^{l-1} \\ &= K^{l-1}. \end{aligned}$$

and $b_{\cdot j} = \sum_i b_{ij} = 1$.

The linear system is given in a block form, which was first utilized by Iusem and Svaiter (1994). See also Vardi (1996) section 3.4 for more detail. Note that if $K^{l-1} \gg 1$, one may consider use only the first term in the right side of (7).

We set the initial values of the parameters as follows: Let all λ_i are equal, $\lambda_1 = \lambda_2 = \ldots = \lambda_l = 1/l$. $\mathbf{R}$ is estimated by the first-order transition counts of the data, except that: (a) If the transition $s_j \mapsto s_i$ is not observed, we arbitrarily set the count equal to one; (b) Fix $r(\mathsf{other}|\cdot) = \epsilon$ and $r(s_i|\mathsf{other}) = (1-\epsilon)/(K-1)$, $i = 1, \ldots, K-1$ as described earlier. We continue the alternating maximization steps until the proportion of the increase of the log-likelihood is less than a pre-specified number (we use 10^{-5}). The EM iteration (7) is applied once in each maximization step, and after that $r(\cdot|\cdot)$ are linearly normalized to satisfy the constraint (2). Based on our experimental results, the estimates obtained from the proposed procedure are very close, if not the same, to the ones from the direct maximization. By using this approach, we significantly reduce the computational effort and are able to derive the MLE of $\mathbf{\Lambda}$ and $\mathbf{R}$ to get an estimated "user signature" or "user profile".

3.2 Independence Model

As explained above when many rare commands are present in the test data, any Markov model is on shaky grounds and we need to resort to methods based on an independence model. This happens approximately 8% of the time in our data set. In these cases we assume that user u's commands are independently generated from a multinomial distribution. That is,

$$P(C_1 = c_1, \ldots, C_T = c_T | \text{user } u) = \prod_{t=1}^{T} P(C_t = c_t | \text{user } u) = q_{uc_1} \ldots q_{uc_T}.$$

Let N_{uk} be the number of times user u used command k in the training data, q_{uk} can be estimated by $\hat{q}_{uk} = N_{uk}/N_{u\cdot}$. Here "$\cdot$" denoting summation over the subscript that it replaces (e.g. $N_{u\cdot} = \sum_k N_{uk}$). Such simple statistics could be poor estimates of the probability that user u uses command k, q_{uk}, because not all commands have the same power to distinguish different users. For $\hat{q}_{u,k}$ to be a proper user-identifier, the following behavior would be desirable:

$$\hat{q}_{u_1k}/\hat{q}_{u_2k} \begin{cases} \approx 1 & \text{if command } k \text{ is common. (command } k \text{ is used by a lot of users)} \\ \propto \frac{N_{u_1k}/N_{u_1\cdot}}{N_{u_2k}/N_{u_2\cdot}} & \text{if command } k \text{ is rare. (command } k \text{ is used by few users)} \end{cases}$$

That is, for a given sequence of commands, the likelihood ratio statistic for testing H_0: command sequence is from user u_1 vs. H_1: command sequence is from user u_2 depends on the users' usage of rare commands. For example, suppose user John didn't use command `cat` in the training data, while every other users in the same community used it. Then the fact that we observe `cat` in John's test data is not a big surprise, as it's a fairly common command for the community. On the other hand, if that command is `xev` and was only used by one user in the community (but not John) during the training period, then it's very unusual if we see command `xev` in John's test data.

To achieve this, we transform N_{uk} as follows: Let π_k be the proportion of users who have used command k in the training data and let the weight $w_k = 1 + \frac{1}{U} - \pi_k$ (U = total number of users).

1. Let

$$N'_{uk} = w_k N_{uk} + (1 - w_k)(N_{\cdot k}\frac{N_{u\cdot}}{N_{\cdot\cdot}})$$

And define

$$d_{u+} = \sum_k \max(0, N_{uk} - N'_{uk}) \qquad d_{u-} = \sum_k \max(0, N'_{uk} - N_{uk})$$

$$E_{uk+} = \begin{cases} \frac{w_k}{\sum_{\{i:N_{ui}>0\}} w_i} d_{u+} & \text{if } N_{uk} > 0 \\ 0 & \text{if } N_{uk} = 0 \end{cases}$$

2. Let

$$N''_{uk} = N'_{uk} + E_{uk+}$$

And define

$$B_{uk} = \begin{cases} w_k N''_{uk} & \text{if } N_{uk} = 0 \\ 0 & \text{if } N_{uk} > 0 \end{cases}$$

$$E_{uk-} = \begin{cases} B_{uk} & \text{if } N_{uk} = 0,\ B_{u\cdot} \leq d_{u-} \\ \frac{d_{u-}}{B_{u\cdot}} B_{uk} & \text{if } N_{uk} = 0,\ B_{u\cdot} > d_{u-} \\ \frac{N''_{uk}}{\sum_{\{i:N_{ui}>0\}} N''_{ui}} (d_{u-} - B_{u\cdot}) & \text{if } N_{uk} > 0,\ B_{u\cdot} \leq d_{u-} \\ 0 & \text{if } N_{uk} > 0,\ B_{u\cdot} > d_{u-} \end{cases}$$

3. Let

$$N'''_{uk} = N''_{uk} - E_{uk-}$$

And define

$$\hat{q}_{uk} = \begin{cases} \frac{N'''_{uk}}{N'''_{u\cdot}} & \text{if } N'''_{uk} > 0 \\ \epsilon & \text{if } N'''_{uk} = 0 \end{cases}$$

The rational behind this is to get the weighted sum of the original user/command contingency table (N_{uk}) and the average one ($N_{\cdot k}\frac{N_{u\cdot}}{N_{\cdot\cdot}}$), where the weight w_k is determined by the proportion of users who used command k in the training data. Step 2 and 3 make sure the new table is properly transformed such that it has the same row (user) marginal as the original N_{uk} has. By applying this, the estimates $\hat{q}_{uk}$ have the desired behavior.

4 Detecting Masqueraders : Hypothesis Testing applied to the Test Data

Suppose that the recorded command sequence $\{c_1, \ldots, c_T\}$ in the test data is proclaimed to be from user u. We then test the hypothesis:

$$\begin{aligned} H_0 : &\quad \text{commands are generated by user } u \\ H_1 : &\quad \text{commands are NOT generated by user } u. \end{aligned} \tag{8}$$

We use the likelihood ratio test in the following way:

Define the statistics

$$X_1 = \log\left(\frac{\max_{v \neq u} L(c_1, \ldots, c_T | \hat{\mathbf{\Lambda}}_v, \hat{\mathbf{R}}_v)}{L(c_1, \ldots, c_T | \hat{\mathbf{\Lambda}}_u, \hat{\mathbf{R}}_u)}\right) \quad \text{and} \quad X_2 = \log\left(\frac{\max_{v \neq u} \prod_i q_{vc_i}}{\prod_i q_{uc_i}}\right) \tag{9}$$

where $\mathbf{\Lambda_u}$ and $\mathbf{R_u}$ are the parameter estimates of user U. Note that in the computation of X_1, the likelihood $L(c_1, \ldots, c_T)$ (defined in (4)) could be zero if there exists pattern $i_l \mapsto i_{l-1} \mapsto \ldots \mapsto i_0$ in the command sequence such that $p = \sum_{j=1}^{l} \hat{\lambda}_j \hat{r}(i_0 | i_j) = 0$. In such case we set $p = \epsilon$.

To bring the two statistics X_1 and X_2 into the same scale, we regress X_2 over X_1 with no intercept term, $X_1 = \rho X_2$, based on the command sequences in the training data. We use a robust regression procedure based on least median of squares to estimate ρ. The estimated $\hat{\rho}$ is then the scaling factor used to equalize the scale of X_1 and X_2.

Let s be the number of commands that are categorized as other, according to user u's definition, in $\{c_1, \ldots, c_T\}$. We define the test statistic (score) to be

$$X = \begin{cases} X_1 & \text{if } s/T \leq \xi, \quad \text{(we choose } \xi = 0.2 \text{ in our experiment)} \\ \hat{\rho} X_2 & \text{if } s/T > \xi. \end{cases} \tag{10}$$

The likelihood ratio test rejects H_0 if X is greater than the threshold w. In our experiment, we let every users have the same threshold $w = \mu + 3\sigma$ and estimate μ and σ by sample mean and

sample standard deviation of the X's obtained from the training data. Specifically, let x_{ub} be the score for user u and commands block b, $A = \{x_{ub}; u = 1, \ldots, 50, b = 1, \ldots, 50\}$. Then $w_1 = \hat{\mu} + 3\hat{\sigma}$, where $\hat{\mu} = \text{mean}(A)$ and $\hat{\sigma} = \sqrt{\text{var}(A)}$. This leads to the following hybrid method: In the case of "usual" test data (the rare proportion is less than ξ), we use the Markov chain model; in the case of "unusual" test data (the rare proportion is greater than ξ), we use an independence model. Namely, we set up a rule base to decide how to process the test command sequences, which is one of the artificial intelligence techniques have been used in many intrusion detection system (Frank (1994)). Since it's impossible to derive the probability distribution of the test statistic (10), we don't know the power of this test. However, from the experiment we get a type I vs. type II error curve based on different critical values, which can be used to evaluate the test procedure.

5 Updating User Profiles and Alarm Threshold

In real life applications it's desirable to be able to update users' training data and profiles in real time, but this involves the risk of including an intruder (masquerader) data in the updated training data set. To reduce the risk of contaminating the training data, we need a much more conservative acceptance-rule than the one which determine the alarm threshold w_1 in the previous section. Note that based on the definitions in (9) and (10), the test statistic X is the log of likelihood ratio. Thus, we set the updating threshold $w_2 = 0$, and any command block with score less than zero has the interpretation that, among all users in the community, this command sequence is most likely from the proclaimed users. All such command blocks are then incorporated into the training data, along with the original one (commands blocks 1 to 50), to form a new updated training data set. Due to the computational demand of the parameters estimation procedure in the Markov chain model, we

can't afford to update the user profiles very often. In our experiment, we only update them once, after commands 10,000.

To update the alarm threshold, we include all the scores of the alarms-free command blocks to the scores of the original training data, and compute their sample mean and sample standard deviation as before. Specifically, let $A = \{x_{ub}; u = 1, \ldots, 50, b = 1, \ldots, 50\}$ and $B = \{x_{ub}; u = 1, \ldots, 50, b = 51, \ldots, 100, x_{ub} \leq w_1\}$. Then $w_{1,new} = \hat{\mu}_{new} + 3\hat{\sigma}_{new}$ where $\hat{\mu}_{new} = \text{mean}(A, B)$ and $\hat{\sigma}_{new} = \sqrt{\text{var}(A, B)}$.

6 Results

Based on the hypothesis (8), type I error indicates a false alarm and type II error indicates a missing alarm. Figure 1 visually displays the inference results for the given alarm threshold. The dark shaded background indicates masquerader data and the light shaded background indicates true user data. The alarms raised by the method with and without user profile updating are represented by $\bullet$ and $+$. In the case of no profile updating, there are 221 false alarms, 103 missing alarms and the percentage of correct inferences is 93.5%. In the case of profile updating, there are 163 false alarms, 115 missing alarms and the percentage of correct inferences is 94.4%.

Based on Figure 1, several interesting points are observed:

1. The false alarms tend to appear in long sequences (e.g. user 16, 20, 23). The interpretation could be that when a user changes his usage pattern, he typically stays at the new usage pattern for several command blocks.

2. After block 100, there are fewer false alarms given by the method which includes updating of training data than those given by the method without updating. This indicates that the newly

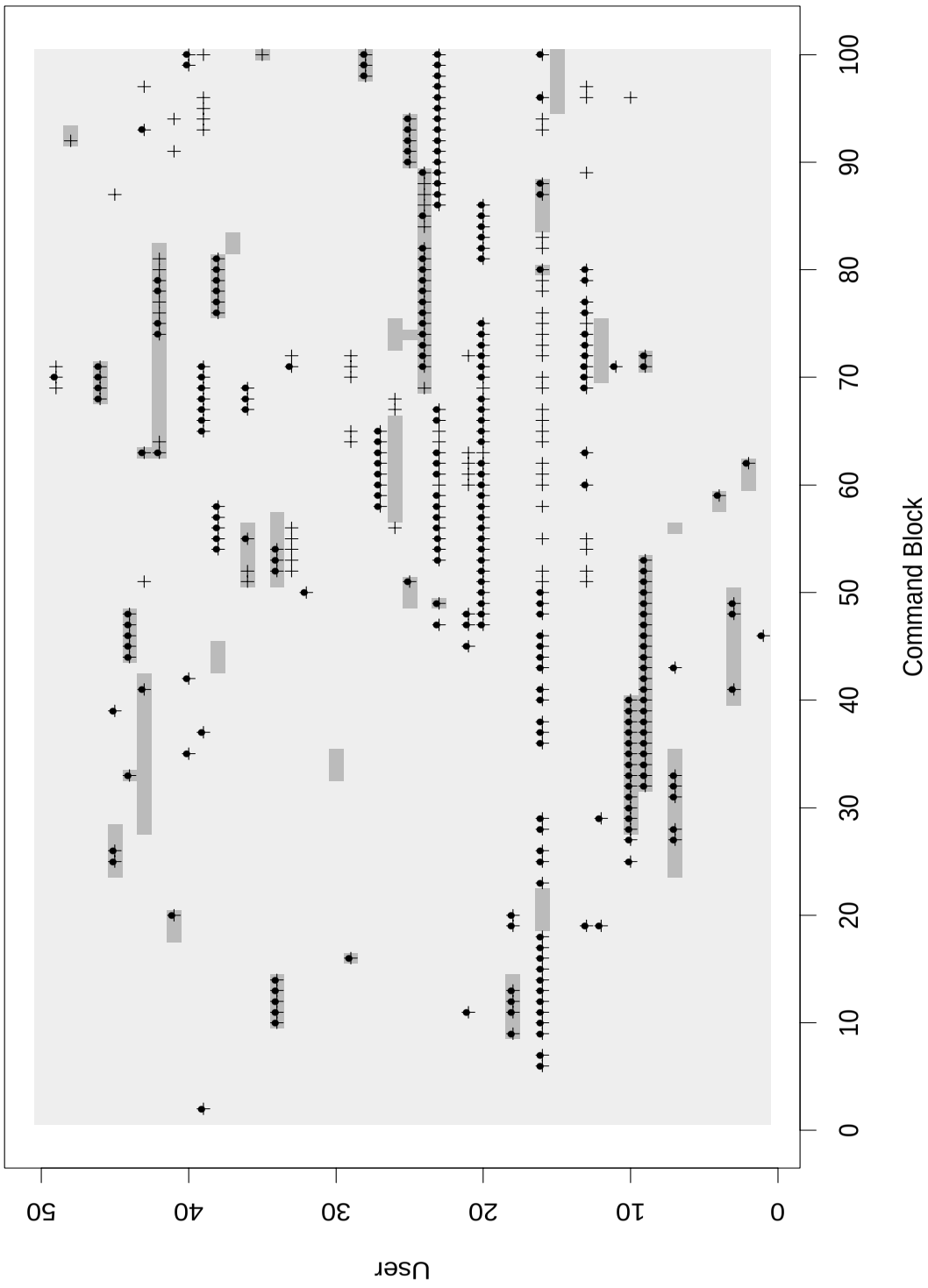

Figure 1: Plot of alarms. • : with updating. + : no updating

included training command sequences, even though determined on the basis of a conservative rule, could be really helpful in profiling the users.

Focusing on individual users, figure 2 shows the plot of the test scores, based on the method with profile updating, for 6 particular users. The horizontal axis represents the command blocks and the vertical one represents the test score. The points above the threshold line indicate alarms and the shaded background indicates the presence of masquerader data.

The results for user 9 and 5 are examples of correct inferences. All the simulated masqerader command blocks are flagged for user 9, and while no such blocks present for user 5, the test correctly doesn't generate any false alarms.

The result for user 30 is an example of the missing alarms. The test scores of the command sequences from the masqerader fail to exceed the alarm threshold. However, it's easy to see from the plot that they are very suspicious, in the sense that those scores are much higher then the average scores.

After about block 100, there are a lot of false alarms for user 20. This could indicate the change of computer usage for this user. The result for user 42 is quite interesting. While the test successfully detects the beginning of the masquerader data, the test scores go down and up dramatically afterwards.

The result for user 16 is an example of the performance improvement due to updating. From block 51 to 100, the test gives a lot of wrong inferences. After block 100, however, the updated training data improves the accuracy of the estimated user profiles, and the procedure's error rates are much smaller.

It's also interesting to see the tradeoff between false alarms and missing alarms. By varying the thresholds, adding different constants to w_1, the false and missing alarm rates change accordingly.

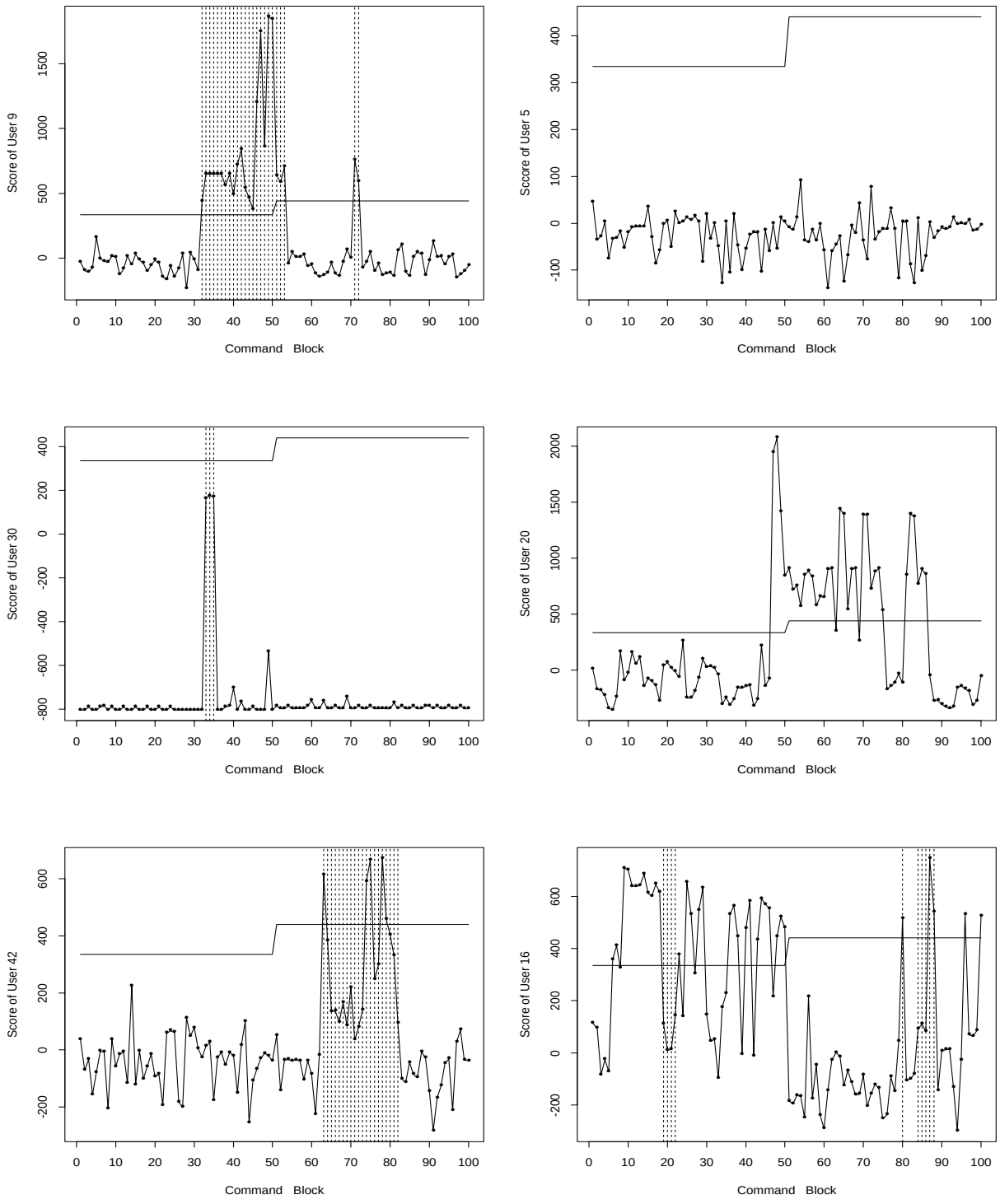

Figure 2: Plot of scores over command blocks for 6 users

The smaller the threshold is, the more the false alarms and the less the missing alarms exist, and vice versa. Figure 3 shows the ROC (Receiver Operating Characteristic) curves of the false vs. missing alarm rates. Each point on the curve represents the error rates given by different thresholds. For 1% false alarm rate, the corresponding missing alarm rates are 78.4% (no updating) and 77.1% (with updating). For 5% false alarm rate, the corresponding missing alarm rates are 40.3% (no updating) and 34.2% (with updating). The highest percentages of the correct inferences are 95.5% (no updating, corresponding to false alarm rate 0.9% and missing alarm rate 79.2%) and 95.8% (with updating, corresponding to false alarm rate 0.6% and missing alarm rate 78.5%)

7 Discussion

We believe that no single criterion can be used to completely defend against computer network intrusions, and our approach can be considered as an example of how to combine various components in a multi-layer defense scheme. A major advantage of our approach is that it uses both probability modeling as well as elementary statistics. Repeated command sequence patterns are modeled by the high-order Markov chain, and information from rarely used commands is utilized through the independent model. Depending on the nature of the command sequences, different strategies can be applied.

While using a high-order Markov chain to model the repeated patterns in the command sequences is successful, there is also a concern about the relatively large computing requirements needed to build the user profile. Based on our experiment, it takes on average 5 minutes to get the parameter estimates for a sequence of 5,000 commands. This can go as short as a few seconds or as long as half an hour, depending on the complexity of the command sequences.

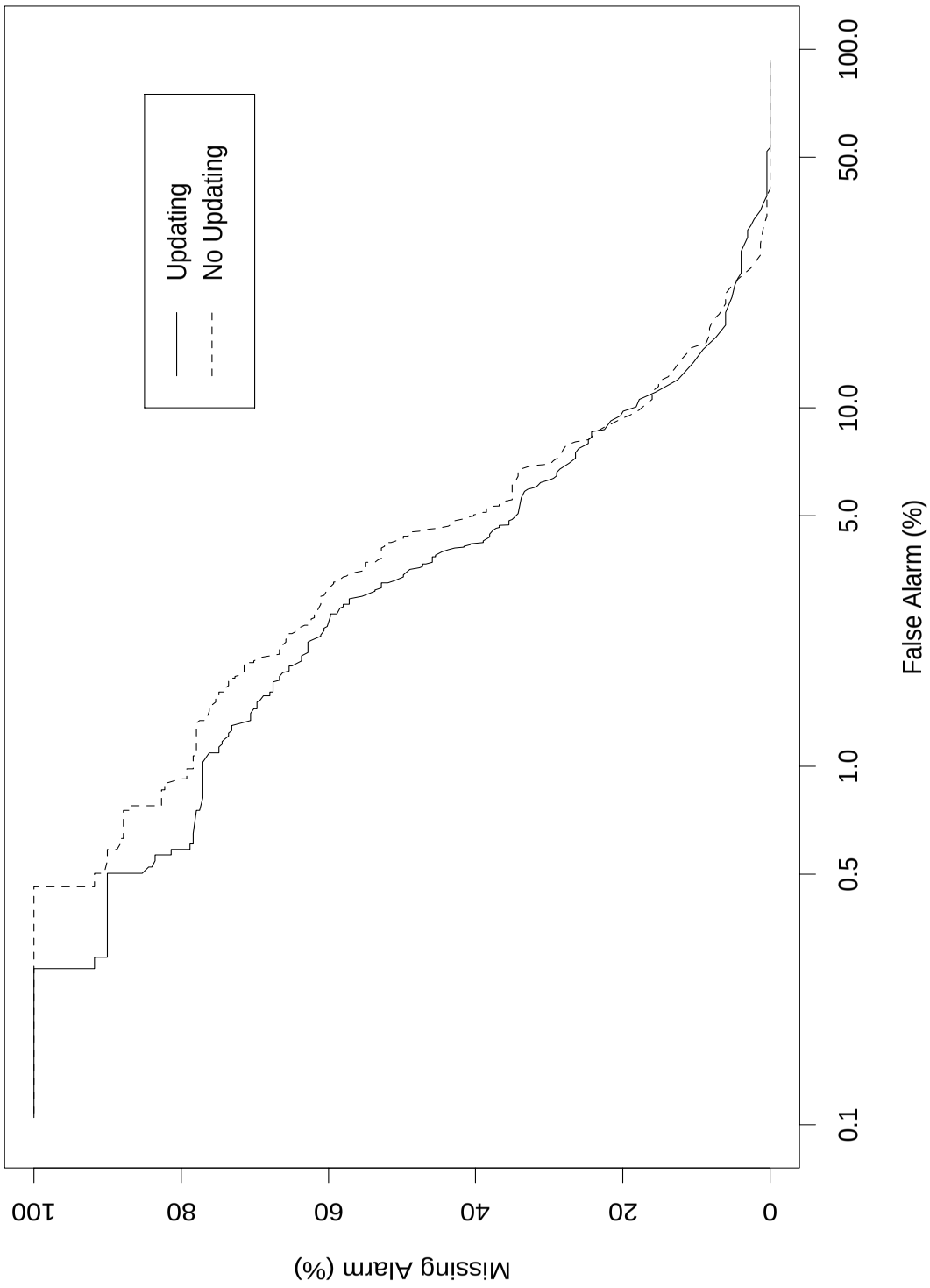

Figure 3: False Alarms vs. Missing Alarms ROC Curve

In the test procedure (9) and (10), the test statistic is defined in a form of the likelihood ratio test. However the numerator of X_1 and X_2 are not really maximized over all alternative hypothesis in (8), but rather over all users in the same community other than the proclaimed user. So it appears that this test can only distinguish users in the same community from one another. However, the experiment's results show that a masquerader's command sequences can be detected quite successfully. Furthermore, an advantage of this test is it can easily accommodate add-on virtual "users" to the community to incorporate misuse detection methods. For example, one can build profiles from the command sequences of a known intrusion, and consider them as the profiles of the intrusive "user" in the community. Then for new command sequences, failure to distinguish them from the virtual (intrusive) "user" indicates a possibly intrusive behavior of the same kind.

Acknowledgements

The authors would like to thank a joint intrusion detection group with members from AT&T Labs, Rutgers University and The National Institute of Statistical Sciences for useful discussion and comments. We are especially grateful to Matthias Schonlau, William DuMouchel and Alan F. Karr for coordinating the study and for some graphical presentation ideas. The authors' work is funded in part by NSF grant DMS-9700867, DMS-9704983 and NSA grant MDA 904-98-1-0027.

References

Anderson, D., Frivold, T. and Valdes, A. (1995), "Next-generation Intrusion Detection Expert System (NIDES): A Summary", Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, May 1995.

Balasubramaniyan, J., Garcia-Fernandez, J. O., Isacoff, D., Spafford, E. H. and Zamboni, D. (1998), "An Architecture for Intrusion Detection using Autonomous Agents", Department of Computer Sciences, Purdue University; Coast TR 98-05; 1998.

Forrest, S., Hofmeyr, S. A., Somayaji, A. and Longstaff, T. A. (1996), "A Sense of Self for UNIX Processes", *Proc. of the 1996 IEEE Symposium on Computer Security and Privacy*, IEEE Press, 1996.

Frank, J. (1994), "Artificial Intelligence and Intrusion Detection: Current and Future Directions", *Proc. of the 17th National Computer Security Conferences*, October 1994.

Iusem, A. N., and Svaiter, B. F. (1994), "A New Smoothing-Regularization Approach for a Maximum-Likelihood Estimation Problem", *Applied Mathematics & Optimization: An International Journal*, 29, 225-241.

Lawrence, C. T., Zhou, J. L. and Tits, A. L. (1997), "User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints", Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland.

NetRanger (1998), "NetRanger Intrusion Detection System Technical Overview", Available at http://www.cisco.com/netranger.

Raftery, A. E. (1985), "A Model for Highorder Markov Chains", *Journal of the Royal Statistical Society*, Ser. B, 47, 528-539.

Raftery, A. E. and Tavaré, S. (1994), "Estimation and Modeling Repeated Patterns in High Order Markov Chains with the Mixture Transition Distribution Model", *Applied Statistics*, 43, 179-199.

Vardi, Y. and Lee, D. (1993), "From Image Deblurring to Optimal Investments: Maximum Likelihood Solutions for Positive Linear Inverse Problems" (with discussion), *Journal of the Royal Statistical Society*, Ser. B, 55, 569-612.

Vardi, Y. (1996), "Network Tomography: Estimating Source-Destination Traffic Intensities From Link Data", *Journal of the American Statistical Association*, 91, 365-377.