NISS

Computer Intrusion Detection Based on Bayes Factors for Comparing Command Transition Probabilities

William DuMouchel

Technical Report Number 91
February, 1999

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

Computer Intrusion Detection Based on Bayes Factors for Comparing Command Transition Probabilities

William DuMouchel
AT&T Labs—Research

Summary

This statistical method can compare in real time the sequence of commands given by each user to a profile of that user's past behavior. We use a Bayes Factor statistic to test the null hypothesis that the observed command transition probabilities come from a profiled transition matrix. The alternative hypothesis is formed as a Dirichlet mixture of multinomial command probabilities. Based on a population of research users on a single computer, data from some users are inserted into the histories of other users to simulate intrusions. The Bayes factor based on the observation of a block of 100 commands had a false alarm rate of about 6.6% while detecting about 78% of blocks from simulated intrusions. We integrate the test into a detection scheme using control charts.

Description of Statistical Methodology

Introduction. In computer intrusion detection one attempts to identify unauthorized accesses to computer accounts. There are two main approaches to intrusion detection: pattern recognition and anomaly detection. Pattern recognition is the attempt to recognize general patterns in command usage that stem from known attacks such as exploiting a software bug. The approach has the disadvantage that it cannot defend against previously unknown software bugs, or any unauthorized user with the knowledge of the account password. Anomaly detection, on the other hand, attempts to identify an unauthorized user by identifying unusual, for the account holder, usage of the computer. Usually, for each user a historical profile is compiled and large deviations from the profile indicate a possible intruder. Therefore it is also referred to as the profile based approach. Intrusion detection systems like IDES (Lunt et al. 1992), NIDES and Emerald (Porras and Neumann 1997) use both approaches, presumably because neither one is uniformly superior to the other. In this paper we only consider the anomaly detection approach. This approach lends itself to a statistical treatment. Ryan et. al. (1998) suggested that each user on a computer system leaves a "print" that could be captured by training a neural network with historical data. When for new data from any user the neural network predicts that the data is more likely to stem from another user in the historical data, then an alarm for a possible intrusion is raised. Forrest et al. (1996) consider anomalies for unix processes (such as ftp, or root) rather than for users. In this paper we propose a test for anomaly detection based on Bayesian hypothesis testing. We are able to test anomalies for unix processes and users using the same methodology.

Command Transition Probabilities. Our method is based on comparing the sequence of each user's commands to a stored profile describing the probability distribution of that user's command sequences. Each command is one of K possible commands, where in our application K may be several hundred. Sparse matrix techniques are used to minimize the computer storage requirements. We represent each user's historical data in terms of a transition matrix of command probabilities

$$\mathrm{p}_{jku} = \mathrm{P}(\text{Next Command} = k \mid \text{Previous Command} = j, \text{User} = u) \qquad (1)$$

The commands are arbitrarily numbered as $j, k = 1, \ldots, K$ and we assume that historical data are available for U users, arbitrarily numbered $u = 1, \ldots, U$. We also assume that each user's transition matrix may evolve over time, and the methodology adjusts for such random evolution by means of an exponentially weighted discount mechanism. As discussed in some of the above references, other data such as the number, duration or timings of logins, the rate of command generation, file or cpu usage, etc., might also form part of a user profile to be checked as part of an anomaly detection effort. This paper continues work begun in DuMouchel and Schonlau (1998a, 1998b) on building models for detecting changes in command frequencies, which could be used in conjunction with other data to form a more complete user profile.

Control Chart Framework. The probabilities (1) must be reliably estimated and updated over time. We measure time not by the clock but in units of T observed commands for each user being monitored. Examples in this paper use $T = 100$ as a group size for computing our Bayes factor test statistic regularly, in the manner of a statistical process control chart (see, e.g., Montgomery, 199x). It is also important to allow for more gradual changes in legitimate user behavior–that is, we must update the user profile on a regular basis. We propose a monitoring process consisting of the following steps:

a) Decide on a block size, T, for computing a measure of discrepancy, x, between current and profiled behavior after every T user commands. Also decide on a run size, B, the number of consecutive blocks (B x-values, BT commands) that must be declared intrusion-free between possible updates of the user profile. Use pilot data (training data) to specify an initial center line μ and an offset, Δ, for determining out of control points. The initial upper control limit of the chart will be $UCL = \mu + \Delta$, and an alarm is raised whenever $x > UCL$. In our implementation, the values of μ vary for each user account to be protected, while the value of Δ is the same for every user. Also use the pilot data (assumed to be intrusion-free) to form estimates, denoted P_u, $u = 1, \ldots, U$, of each user's transition probabilities. Set the run counter to $b = 0$ and run the following process independently for each user:

b) Observe a block of T commands and evaluate them using P to form the test statistic x.

c) If $x \geq UCL = \mu + \Delta$, raise an alarm for a possible intruder and set the run counter to $b = -1$. If $x < UCL$, augment the run counter $b = b + 1$.

d) If $b = B + 1$, use the previous BT commands *before* the most recent block of T commands to update P and μ, and then reset $b = 1$. The run of B alarm-free blocks is not integrated into the profile unless there is an extra alarm-free block on each side of the run.

e) Return to step b) and repeat indefinitely.

This paper focuses on a Bayesian scheme for estimation of P_u and computation of a test statistic x. We lay out the steps a)–e) above to put our methods in context and to show their potential utility. We have not tried to optimally adapt control chart methodology for this application, but rather merely strive for a simple procedure. Of course computer intrusion detection methods that avoid control charts altogether can be and have been devised. We introduce control charts here both because they are a standard method in industrial statistical practice for controlling false alarm rates, and as a way to encourage control chart use for computer intrusion detection.

The Bayesian Dirichlet-Multinomial Model

The most general and commonly used model for categorical data is the multinomial distribution, and Bayesian inference involving multinomial probabilities often uses a Dirichlet prior distribution (O'Hagan 1994, Ch. 10; Johnson and Kotz, 1969, Ch. 11.8.1), because they form a conjugate family of distributions. We use two separate applications of the Dirichlet-multinomial family, once to estimate the distribution of marginal command probability vectors from different users, and once to allow estimates of transition probabilities to "borrow strength" from marginal probabilities. These two applications are combined in the computation of our Bayes factor test statistic.

Let $p = (p_1, \ldots, p_K)$ be a random probability vector constrained to the simplex $p_k > 0$, $\Sigma_k p_k = 1$. The vector p has a Dirichlet distribution if its probability density is given by

$$f(p) = \Gamma(\Sigma_k \alpha_k)\ \Pi_k\, p_k^{\alpha_k - 1}/\Gamma(\alpha_k) \qquad [\alpha_1 > 0, \ldots, \alpha_K > 0] \tag{2}$$

For a fixed sample size $n_.$, suppose one of the K commands is drawn independently at random $n_.$ times with respective probabilities $(p_1, \ldots, p_K)$, and let n_k be the number of times command k is drawn, where $\Sigma_k n_k = n$. Then the multinomial probability of the count vector $n = (n_1, \ldots, n_K)$ is

$$\text{Prob}(n \mid p) = n_.!\ \Pi_k\, p_k^{n_k}/n_k! \tag{3}$$

If one first draws p from (2), and then uses the result to draw n from (3), the marginal distribution of n is the Dirichlet compound multinomial distribution

$$\text{Prob}(n) = [n_.!/\alpha_.(\alpha_. + 1)\cdots(\alpha_. + n_. - 1)]\ \Pi_k\, [\alpha_k(\alpha_k + 1)\cdots(\alpha_k + n_k - 1)/n_k!] \tag{4}$$

where $\alpha_. = \Sigma_k\, \alpha_k$. After observation of n, the posterior distribution of p is also Dirichlet with the parameter α_k replaced by $\alpha_k + n_k$. If a very large sample $n_.$ has been observed, all the values of $\alpha_k + n_k$ will be large and the Dirichlet posterior distribution will be highly concentrated around the posterior mean values, $E[p_k \mid n] = (\alpha_k + n_k)/(\alpha_. + n_.)$, which are sometimes called *shrinkage estimates* of p, since they move the observed proportions $n_k/n_.$ from different users toward the common prior means $\alpha_k/\alpha_.$.

Empirical Bayes Model for Marginal Probabilities

Before discussing the estimation of command transition probabilities, we discuss the estimation of marginal probabilities as an intermediate step. We assume that data are available on command frequencies for each of U users, and that it is desired to estimate q_{uk}, the proportion of times that user u uses command k. The data consist of a count matrix M_{uk}, the number of times user u was observed to use command k. For this estimation of q, we ignore dependencies within command sequences and use methods based on an independence model. Although the per-user sample sizes $M_{u.}$ are fairly large, the natural estimates $q_{uk} = M_{uk}/M_{u.}$ will contain many 0 values because K is large and many commands are infrequent. To avoid this we use empirical Bayes shrinkage estimates of the form

$$q_{uk} = (M_{uk} + \alpha_{uk})/(M_{u.} + \alpha_{u.}) \tag{5}$$

where the Dirichlet parameters α_{uk} must be estimated from the matrix M_{uk}. The standard Dirichlet model assumes that each user's vector of probabilities was drawn from a common Dirichlet distribution, which would require that α_{uk} not depend on u. We are using an extension of this model, in which we first estimate a common parameter α_{0k}, and then modify that estimate differently for each user.

Estimation of the common Dirichlet Parameter. We use a method of moments estimate of α_{0k}. Let $\pi_k = \alpha_{0k}/\alpha_{0.}$ and let α_0 be a preliminary estimate of $\alpha_{0.} = \Sigma_k\, \alpha_{0k}$. Then, assuming that the variance of each $M_{uk}/M_{u.}$ is approximately proportional to $1/\alpha_0 + 1/M_{u.}$, a natural estimate of π_k is to weight inversely proportional to variance, namely

$$\pi_k = \Sigma_u(1/\alpha_0 + 1/M_{u.})^{-1}(M_{uk}/M_{u.}) \,/\, \Sigma_u(1/\alpha_0 + 1/M_{u.})^{-1} \qquad k = 1, \ldots, K \tag{6}$$

The estimate α_0 can be updated given π_k by computing the chi-squared statistic

$$X^2 = \Sigma_{u,k}(M_{uk} - M_{u.}p_k)^2/M_{u.}p_k$$

The excess of X^2 over its degrees of freedom is expected to be proportional to $1/\alpha_0$, leading to the estimate

$$a_0 = \max[.5,\ 1/\max(.0002, [X^2 - (U\text{-}1)(K\text{-}1)]/(K\text{-}1)M_{..})] \tag{7}$$

In (7) the estimate α_0 has been artificially restricted to the interval $0.5 < \alpha_0 < 5000$. The empirical Bayes estimates are produced by iterating (6)-(7) until convergence, and defining our final estimate of α_{0k} to be

$$\alpha_{0k} = \alpha_0\, \pi_k, \qquad k = 1, \ldots, K. \tag{8}$$

Extended Model with Estimation of User-Specific Commands. Schonlau and Theus (1998) report on the implications of the fact that very many of the commands given by unix users are user-specific. That is, many commands are commonly used by a small subset of the U users and seemingly never used by the other users. This may happen because the user has actually created a private command definition, or just because very few users are aware of that command or ever find it useful, even though others use it frequently. In such a case, the Dirichlet model for marginal command frequencies will not fit well. We extend the model by letting each of the commands k be associated with two separate hyperparameters, α_k and β_k, and a set of unobserved variables δ_{uk}, $u = 1, \ldots, U$, where

α_k = Dirichlet parameter for command k, applicable to users that ever use k

β_k = Prob(a randomly chosen user will ever use command k)

$\delta_{uk} = 1$ if user u ever uses command k, $\qquad = 0$ if user u never uses command k

Using these definitions, we modify the assumption about each user's prior distribution for the marginal command probabilities to be

$$(q_{u1}, \ldots, q_{uK}) \mid \delta \sim \text{Dirichlet}(\alpha_1\delta_{u1}, \ldots, \alpha_K\delta_{uK}) \tag{9}$$

Thus our extended procedure is to define α_{uk} in (5) to be estimates of $\alpha_k\delta_{uk}$ in (9). Let α_{0k} be the estimate of α_k defined iteratively using equations (6)-(8) as described above. Next set

β_k = Proportion of users who have ever used command k in the historical data

$\alpha_k = \alpha_{0k}/\, \beta_k$

In order to use (9), we need estimates of the δ_{uk}. If $M_{uk} > 0$, then we set $\delta_{uk} = 1$. But suppose after observing $M_{u.}$ commands of user u, we still have $M_{uk} = 0$. Then we use the Bayes rule estimate

$$E[\delta_{uk} \mid M_{u.}] = \beta_k\, L(M_{u.})/[1 - \beta_k + \beta_k\, L(M_{u.})]$$

$$L(M_{u.}) = [1 - \alpha_k/\alpha_.][1 - \alpha_k/(\alpha_.+1)]\cdots[1 - \alpha_k/(\alpha_.+M_{u.}-1)] \approx (1 + M_{u.}/\alpha_.)^{-\alpha_k} \tag{10}$$

The quantity $L(M_{u.})$ is the likelihood of observing $M_{u.}$ successive commands in a row, none of which are command k, if the commands have been generated by the compound multinomial distribution given by (4). The right-most expression in (10) is an approximation based on a Taylor's series. [Approximate $1 - \alpha_k/(\alpha_.+m)$ by $\exp\{-\alpha_k/(\alpha_.+m)\}$ and approximate the resulting summation in the exponent by $-\alpha_k\log(1 + M_{u.}/\alpha_.)$.] In this extended model, all users having $M_{uk} > 0$ share the same α_{uk}, but users that have not used command k have different Dirichlet parameters for that command, which approach 0 as $M_{u.}$ becomes large:

$$\alpha_{uk} = \alpha_{0k}/\beta_k \qquad (M_{uk} > 0) \qquad (11a)$$

$$\alpha_{uk} = \alpha_{0k}/[\beta_k + (1 - \beta_k)(1 + M_{u.}/\alpha_.)^{\alpha_{0k}/\beta_k}] \qquad (M_{uk} = 0) \qquad (11b)$$

Shrinkage Estimates of Transition Probabilities

Let $P_u = \{p_{jku}\}$ be (1), the desired estimates of transition probabilities from command j to command k for user u, and, as before, let q_{uk} be the estimated marginal probability that user u uses command k. Also let N_{jku} be the frequency counts of command transitions based on historical data. In what follows, we shall drop the subscript u when there is no danger of confusion and just refer to p_{jk}, q_k, and N_{jk}, respectively. For many (j, k) pairs, there will be little or no dependency and we might expect p_{jk} to be close to q_k. Our shrinkage estimate represents p_{jk} as an average of the observed transition proportions and the marginal estimate, namely

$$p_{jk} = (N_{jk} + n_j\, q_k)/(N_j + n_j) \qquad N_j = \Sigma_k N_{jk} \qquad (12)$$

$$n_j = \max\{.5,\ \ 1/\max[.0002, (X_j^2 - K')/N_jK']\} \qquad X_j^2 = \Sigma_k(N_{jk} - N_j\, q'_k)^2/N_j.q'_k$$

In the above equations, $K'+1$ is the number of k-values for which there are any $N_{jk} > 0$, and q'_k is a renormalization of q_k so that the corresponding q'_k values sum to 1. (Recall that $N_{jk} = 0$ for all j only if the corresponding $M_{uk} = 0$ and from (5) $q_k = q_{uk}$ will be close to 0. We want to leave such k-values out of the chi-squared calculation, but *not* out of the general formula (12).) The n_j will be large if the raw transition frequencies are not significantly different from the marginal frequencies, but small if the raw transition frequencies are reliably different than the marginal ones. Note also that for any j such that every $N_{jk} = 0$ and thus $N_j = 0$, the value ν_j cancels out of (12) and $p_{jk} = q_k$.

In this second use of Dirichlet estimation, we are in effect assuming that each transition probability vector has been drawn from a Dirichlet population with parameters $\alpha_{jk} = \nu_j q_k$. In this case we are taking the q_k as given and only need to estimate ν_j, which corresponds to $\alpha_{0.}$ in the earlier Dirichlet problem. The empirical Bayes hierarchical models and the corresponding shrinkage estimates enable us to compute reliable estimates of these large transition matrices (a total of UK^2 parameters) with a limited amount of historical data. We are finally ready to define the test for intrusion detection.

Bayesian Hypothesis Testing Framework

Suppose that user u is online and being monitored, and has generated a sequence of T+1 commands C_0, C_1, …, C_T. We consider the two hypotheses

$$H_0\text{: } P(C_t = k \mid C_{t-1} = j) = p_{jku} \qquad (13)$$

$$H_1\text{: } P(C_t = k \mid C_{t-1} = j) = Q_k \qquad (Q_1, \ldots, Q_K) \sim \text{Dirichlet}(\alpha_{01}, \ldots, \alpha_{0K})$$

The null hypothesis H_0 assumes that the legitimate user has generated the data from the profiled transition probabilities. The alternative hypothesis H_1 assumes that the T commands have been drawn independently using an arbitrary probability vector, which was drawn from a Dirichlet distribution with specified hyperparameters. These hyperparameters could be estimated from a database of intrusion records, if one were available, but in our examples we use the estimated hyperparameters (8) from the U users used above to estimate the q_{uk}. Although the alternative H_1 is more general than H_0 in that Q is not specified while $P_u = \{p_{jku}\}$ is completely specified, it is less general in that it only involves marginal probabilities and not transition probabilities. Thus H_0 is not nested within H_1. Since the number of commands T available to compute the intrusion detection statistic x is necessarily small compared to the much larger amount of historical data available for estimation of P, it is not feasible to have a more general comparison of two sets of transition matrices. In our formulation, there is no carry-over of command frequency information from one block of T commands to the next, except through the updating of the profile parameters P and μ. In our examples, we take $T = 50$ or 100.

Bayes Factor and Weight of Evidence. The Bayes Factor BF is the ratio of the probabilities of the data under the two hypotheses:

$$BF = \text{Prob}(C_1, \ldots, C_T| \text{H}_1) / \text{Prob}(C_1, \ldots, C_T| \text{H}_0) \tag{14}$$

The larger BF is, the more evidence there is against H_0 in favor of H_1. In fact, $x = \log(BF)$ is often called the *weight of evidence*. On the log scale there is the nice property that evidence from two independent data sets is the sum of their individual evidence. The subjective Bayesian interpretation of BF is that the prior odds in favor of H_1 are multiplied by BF in order to produce the posterior odds in favor of H_1. Rather than focus on this subjective probability interpretation, which depends on the model being exactly correct, we will just treat $x = \log(BF)$ as a test statistic with solid Bayesian credentials for being quite informative about the hypotheses in question.

Let n_{jk} be the transition count matrix from $C_0, \ldots, C_T$. Then, using (3), (4), (13) and (14),

$$BF = \Pi_k[\alpha_{0k}(\alpha_{0k}+1)\cdots(\alpha_{0k}+n_{.k}-1)] \;/\; [\alpha_{0.}(\alpha_{0.}+1)\cdots(\alpha_{0.}+T\text{-}1)\; \Pi_{j,k}\, p_{jku}{}^{n_{jk}}] \tag{15}$$

From (15), BF is the product of T terms, each one of the form $(\alpha_{0k} + i)/(\alpha_{0.} + t)p_{jku}$, amounting to just 5 arithmetic operations per monitored command. In addition, the computation of each p_{jku} using (12) requires at most 4 more operations (fewer if $N_j = 0$) and, to avoid the possibility of underflow or overflow, it is advisable to take the log of each of the T terms and sum, adding one log calculation per monitored command. Thus we have about 9 floating point operations plus a log computation per monitored command, plus the integer operations needed to match command names in order to access the sparse array N_{jk} and the other variables needed to evaluate (15). (To reduce the chance of wild outliers, we bound each of the T terms that sum to $\log(BF)$, so that if $|\log[(\alpha_{0k} + i)/(\alpha_{0.} + t)p_{jku}]| > 10$, it is replaced by ± 10.)

The Control Chart

Returning to the control chart framework discussed earlier in this section, we must set thresholds for the x_{ui}, the log(BF) for the ith block of T commands from user u, to control the false alarm rate and to be sensitive to changes in the distribution of x, especially to increases in the mean of x. A simple procedure is to set $\text{UCL}_{ui} = \mu_{ui} + \Delta$, where Δ is empirically determined and is the same for all blocks and for each user. We define an alarm condition any time that $x_{ui} \geq \text{UCL}$. In order to get initial estimates μ_{u0} and estimates of Δ, we use the same data used to estimate the α_{0k}, assumed to be intrusion free. We first estimate the P_u based on just a subset of the commands from each user in the training data and then calculate x-values for each user based on the remaining commands known to be intrusion-free. Let x_{ui}, $u = 1, \ldots, U$; $i = 1, \ldots, I$, be the resulting values of log(BF). Choose a global value UCL_0 (we take $UCL_0 = 0$) as an average threshold for raising an alarm. However, we want to adjust for the fact that different users may have different average values of x in the absence of intruders. Therefore we define initial values of μ_{u0} and Δ so that the average value of UCL_{u0} is UCL_0. Namely, let

$$\bar{x}_u = \Sigma_i\, x_{ui} / I \qquad \bar{x}. = \Sigma_u\, \bar{x}_u / U$$

$$\Delta = UCL_0 - \bar{x}. \qquad \mu_{u0} = (\bar{x}. + \bar{x}_u) / 2$$

The rationale for $\mu_{u0} = (\bar{x}. + \bar{x}_u)/2$ is that the user-specific means $\bar{x}_u$ are too variable to trust completely, being based on just I blocks, so they are shrunk towards the grand mean. Other choices of UCL_0 besides the value 0 could be based on percentiles or the sample variance of $(x_{ui} - \bar{x}_u)$. Once m_{u0} and Δ are estimated, the initial values of P_u are updated with the commands used to compute the x_{ui}. This completes the initialization of the control chart described in step a) of the subsection "Control Chart Framework."

Decision to update. If, for a run of $B + 2$ consecutive values of i, $x_i <$ UCL, then the data from the middle B such blocks are approved for use in updating the process parameters. The choice of B depends on the computational burden of going through the updating process and on the expected frequency of intrusions and of expected changes in legitimate user behavior. The choice $B = 1$ results in including all non-alarm x_{ui} except those immediately before or after an alarm. Larger values of B result in a procedure that is less likely to contaminate the training data with intrusions and that is also less demanding computationally.

Updating m. The implementation of step d) of the control chart framework listed above, the updating of process parameters P and m, involves the choice of another constant, denoted $n_{1/2}$, called the *command half-life*, that determines how quickly old command counts "age out" of the computations. Suppose we wish to modify the current estimate of μ, based on B_m values of x, to include information from B new values of x having mean $\bar{x}$. Then we replace the pair (B_m, μ) by the pair

$$B_m^{\text{new}} = B + B_m\, 2^{-BT/n_{1/2}}$$

$$\mu^{\text{new}} = (\bar{x}\, B + \mu\, B_m\, 2^{-BT/n_{1/2}}) / B_m^{\text{new}}$$

Updating the transition matrix *P*. We recommend updating the estimated transition matrix as follows. First, update the estimated marginal probability vector q_k. Recall from (5) that q_{uk} is based on the two matrices M_{uk} and α_{uk}. Denote the command counts in the new data as m_k, where $\Sigma_k m_k = BT$. Then these matrices and the q_k are updated as

$$M_{uk}^{\text{new}} = m_k + M_{uk} 2^{-BT/n_{1/2}} \qquad M_{u.}^{\text{new}} = \Sigma_k\, M_{uk}^{\text{new}}$$

$$\alpha_{uk}^{\text{new}} = [\text{Use (11) with updated values of } M_{uk}]$$

$$q_{uk}^{\text{new}} = [\text{Use (5) with updated values of } M_{uk} \text{ and } \alpha_{uk}]$$

To update the transition probabilities p_{jku}, denote the transition counts in the blocks being newly included as n_{jk}, where $\Sigma_{j,k}\, n_{jk} = BT$. (Note that here we are defining n_{jk} as being the count over the set of B blocks, B times as much data as in the discussion of the Bayes factor computation above.) As before, N_{jk} is defined as the previously incorporated transition counts for user u. Then, in order to be able to use (12) to compute new p_{jku}, we define new values of N_{jk}, N_j, and ν_j as

$N_{jk}^{\text{new}} = n_{jk} + N_{jk}\, 2^{-BT/n_{1/2}}$ $\qquad\qquad$ $N_j^{\text{new}} = \Sigma_k N_{jk}^{\text{new}}$

ν_j^{new} = [Use updated values of N_{jk}, N_j, and q_{uk} in the formulas defining X^2 and ν_j just after (12)]

The discounting of the observed counts decreases the influence of data from the far past, allowing the probabilities and the control chart specifications to keep up with gradual changes in user behavior. Larger values of $n_{1/2}$ produce more gradual changes in the parameters and smaller values produce more current updates that may possibly be unstable from excess sampling variation. (We use $n_{1/2}$ = 5000.) Our algorithms only store nonzero values of n_{jk} and N_{jk} for each user, and compute all required values of p_{jku} on the fly.

Updating a$_{0k}$. We do not propose a regular updating scheme for the in-common prior hyperparameters α_{0k}, but they could be updated or replaced on an ad-hoc basis if it were desired to redefine H_1. If, during an update, entirely new commands are observed, we just take $\alpha_{0,K+k} = .01$, $\beta_{K+k} = 1/U$, $k = 1, 2, \ldots$ and then increase K, but we could re-estimate all the α_{0k} instead. If the Bayes factor computations involve a previously unobserved command, the values $\alpha = .01$, $\beta = 1/U$ are used for these computations too.

Data and Results

To evaluate the method presented in the previous section, we compare test data to training data (profiles) for pairs of users. Ideally, an alarm should always be raised except when a user is tested against his or her own profile. To establish user profiles, we use historical data from usage on our local unix machine. The data (user names and commands) are extracted from output of the unix `acct` auditing mechanism and consist of user names and commands only (without their arguments). Commands recorded by the system consist not only of commands typed but also include implicitly generated commands. For example, for each execution of the .profile file or a make file, all commands contained in these files are also recorded in the data stream.

Experimental Design Data were collected from our local population during a several-month time period. This example uses 50 strings of 15,000 consecutive commands each taken from the records of 50 users. The first 5,000 commands (50 blocks of 100) are taken as originally recorded. However, beginning with block 51, the design builds in the possibility that strings of 500 commands (5 blocks) from other users (not from any of the original 50) are inserted into these sequences. After each block of 100 original commands, with probability .2 a subsequence of 500 "intruder" commands is inserted into the stream. At the end of each set of 5 intruder blocks, one block of original commands is always provided, after which the choice of original vs. intruder continues with the same (.8, .2) probabilities. Each of the 50 constructed sequences stops after 15,000 commands, having a mixture of original and inserted "intruder" commands. The task is to detect the intruder blocks. This design leads to approximately half of the test data arising from a simulated intrusion, which of course is orders of magnitude greater than expected in real life. Having roughly equal numbers of simulated intrusions and controls allows us to estimate the misclassification errors better.

Estimation Results

We estimated the parameters $P_u = \{p_{jku}\}$, α and μ_u based on the initial 5000 commands from each of the 50 users that were known to be intrusion-free. There were K = 636 distinct commands in these data. However, the 50 x 15,000 commands including the simulated intrusions contained 933 distinct commands. Based on the first 50 x 5000 commands, the sum of the α_{0k} was $\alpha_{0.}$ = 106.6, and the largest 20 values of a_{0k} were

```
  sh netscape    ls generic   cat popper sendmail  date    rm  expr
8.88    4.354 4.317   4.137 3.831   3.62    3.179 2.316 2.315 2.154

  sed grep hostname    ln tcpostio   ksh  nawk  tcsh uname true
2.074 1.47    1.469 1.421    1.417 1.415 1.336 1.333  1.32 1.31
```

Values of α_{0k} were somewhat arbitrarily raised to 0.01 if the estimates fell below that value. In addition, as discussed above, previously unseen commands that show up past the initial period are also given the value $\alpha_{0k} = 0.01$. Initially 355/636 commands had $\alpha_{0k} = 0.01$. These rare commands are often used by only one or a few users, and tend to contribute much of the information allowing discrimination between users. See Schonlau and Theus (1998).

There are a potential of $UK^2 = 50 \times 933^2 = 43.5$M different transition probabilities to estimate. However, only about 22,000 of these transitions actually occurred in the initial data, and even after updating each P_u based on seemingly intrusion-free runs of test data, only about 29,000 distinct transitions are ever stored and counted. This shows the utility of Bayesian shrinkage estimation, which allowed us to produce nonzero estimates for all 43.5M p_{jku}. Despite the many empty cells, most users had hundreds or even thousands of identical transitions in their most populated cells.

Control Chart Results

Figures 1 and 2 show typical control charts generated under this simulated scenario. The values of $x = \log(BF)$ are plotted as points connected by lines and the values of μ and UCL are plotted as two parallel nearly horizontal curves. The values $n_{1/2} = 5000$ and either ($T = 50$, $B = 6$) or ($T = 100$, $B = 2$) were used in this example. To estimate μ_{u0} and Δ, P_u based on the first 4000 commands were used to test the next 1000. The top panel in each Figure shows the control chart for $T = 100$, the bottom for $T = 50$. The plotted symbols show the source of the commands in each block; filled circles for simulated intrusions, open circles for commands from the original user. Thus open circles above the line labeled "UCL", or filled circles below the line, represent misclassification errors. Figure 1, describing User 1, has relatively few misclassification errors. From the top panel, when $T = 100$, there was just one false alarm in 5000 legitimate commands, or 0.2 per thousand. Of the ten inserted intrusions in Figure 1, an alarm was sounded after the very first block in nine intrusions, while the remaining intrusion, beginning in block 59, was missed completely. One method of scoring a detection rate is to count the average number of simulated intruder commands before detection. In the case of Figure 1 (top) it would be $.9\times100 + .1\times500 = 140$. The bottom panel of Figure 1 shows what happened with the same input data when the blocksize was $T = 50$. There are twice as many points on the chart, but there is still just one false alarm for the same false alarm rate of .2 per 1000 legitimate commands. (The one open circle seemingly exactly on the UCL line is actually slightly below it.) Of the ten intrusions, the first alarm was at block 1 8 times and at block 2 twice, so that the average number of commands before detection in that panel is $.8\times50 + .2\times100 = 60$.

Figure 1. Control chart for User 1. Top: N = 100 commands per point; Bottom: N = 50.
Simulated intrusions are "•", control blocks are "o." Points above the line marked "UCL" are alarms.

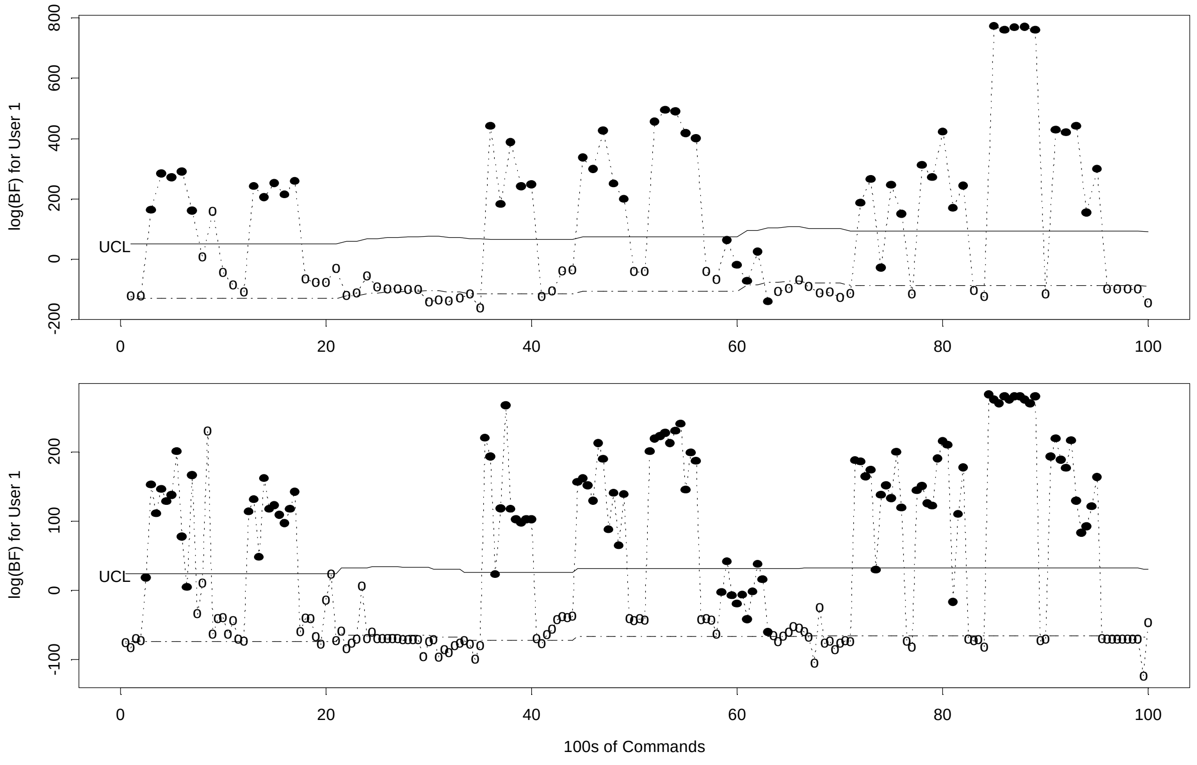


Figure 2. Control chart for User 2. Top: N = 100 commands per point; Bottom: N = 50.
Simulated intrusions are "•", control blocks are "o." Points above the line marked "UCL" are alarms.

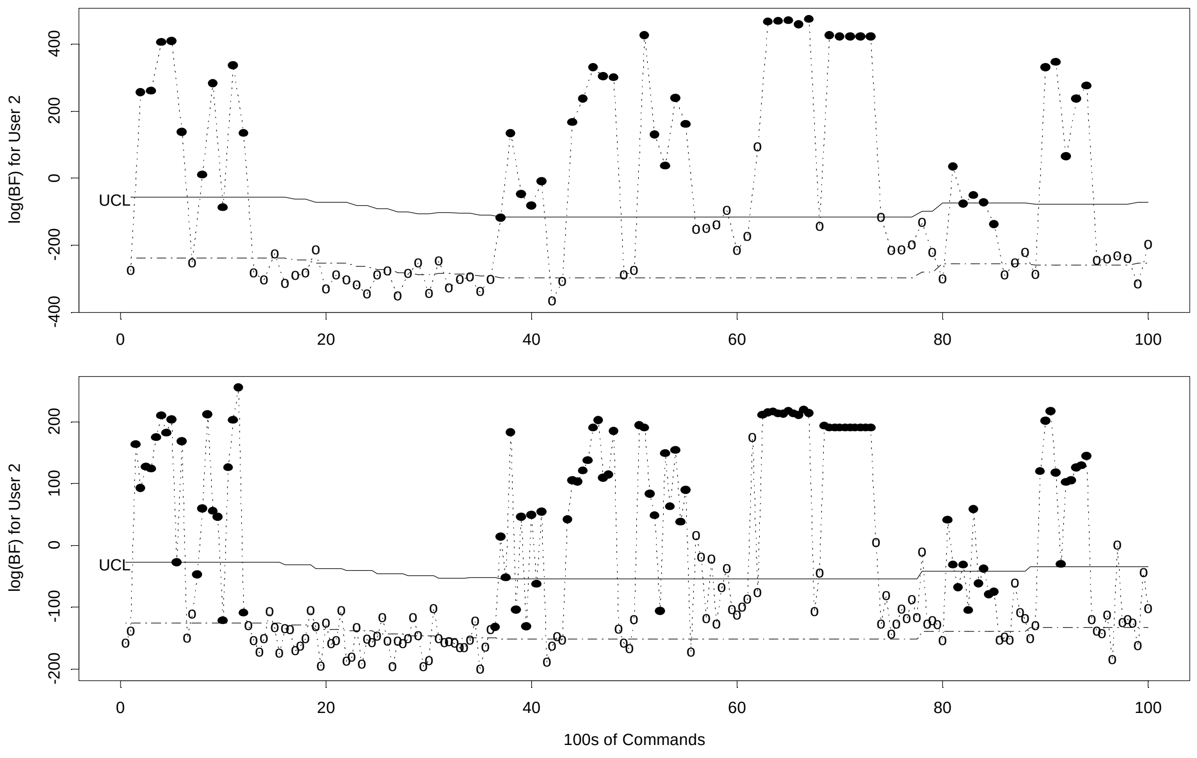


Figure 2 displays the same data for User 2 in our sample. This User suffered just 9 simulated intrusions. The false alarm rate is 2/5500 = .36/1000 when *T* = 100 and 9/5500 = 1.6/1000 when *T* = 50. In the top panel all nine intrusions raise an alarm in the first block, for 100 commands/intrusion before detection. In the bottom panel, the average is (7/9)×50 + (2/9)×100 = 61. These two Figures show the variability inherent in such a simulation, and also that neither of the two studied blocksizes is uniformly better. But both types of error rate are fairly low in these Figures.

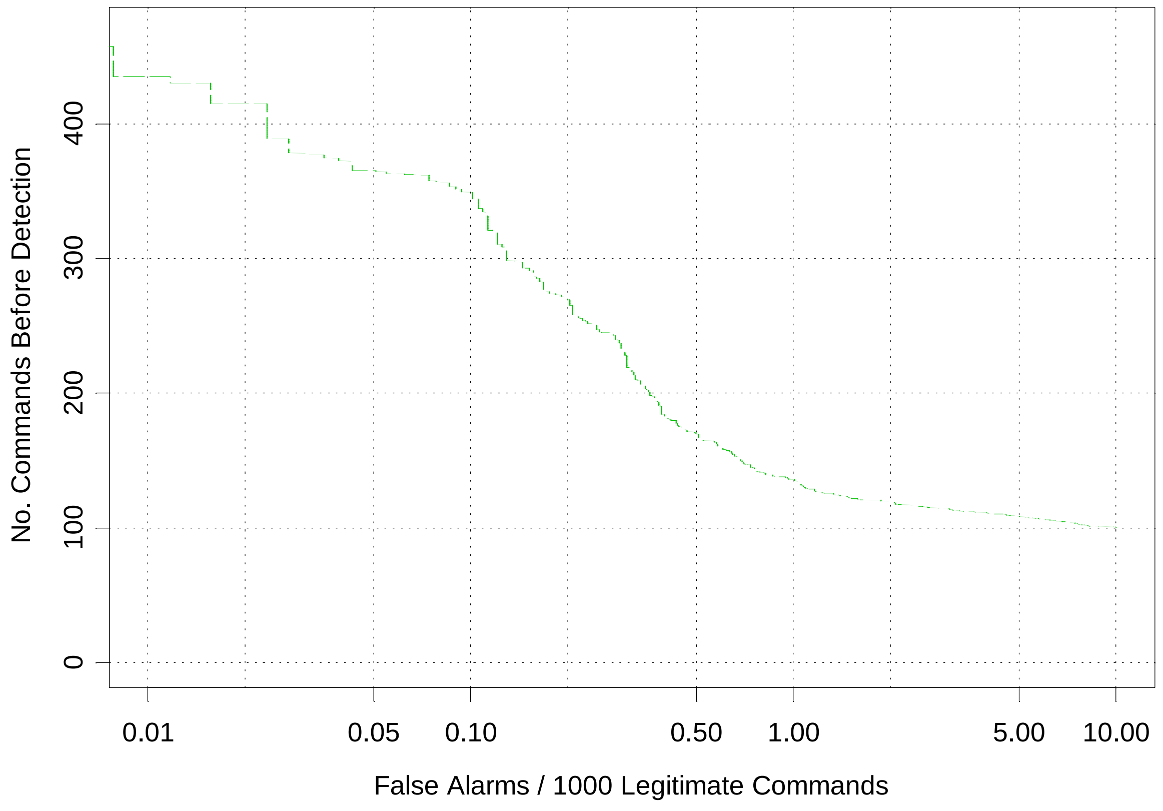
False alarm and detection rates Over all 50 users, the number of false alarms when *T* = 100 was 170 during 256,500 control commands, a rate of .66/1000. The *T* = 50 procedure had 353 false alarms for a rate of 1.38/1000. During the 487 simulated 500-command intrusions, 449 (92.2%) raised an alarm when *T* = 100, while 465 (95.5%) raised an alarm when *T* = 50.

During "intrusions", the distribution of number of commands before detection was

Commands:	50	100	150	200	250	300	350	400	450	500	500(Never)
Count (*T* = 100):		381		30		19		12		7	38
Cum. Detection (%):		78		84		88		91		92	100
Count (*T* = 50):	373	30	14	17	9	5	3	7	2	5	22
Cum. Detection (%):	77	83	86	89	91	92	93	94	94	95	100

The average number of commands/intrusion before detection was 158 for *T* = 100, and 101 for *T* = 50.

Figure 3. Trade-off between false alarms and detection ability, as functions of the offset Δ.
Solid curve: Blocksize = 50 ; Broken curve: Blocksize = 100.



Detection-false alarm tradeoff. It is interesting to compare how the false alarm and detection rates change as Δ, the offset of *UCL* from μ, change. As noted above, the smaller blocksize had a higher false alarm rate but lower average commands/intrusion before detection. It is natural to be curious as to which would have the better detection rate if Δ were adjusted so they had the same false alarm rate. Figure 3, which is similar in spirit to an ROC curve in acceptance sampling (Montgomery, 1991), displays the detection-false alarm tradeoff. For each possible value of Δ, both the number of false alarms and the average commands before detection are computed and then plotted against each other. (The computation merely adds a constant to the observed values of *UCL* in the simulation, and does not take into account how the inclusion of new data into updated profiles might change if *UCL* changed.) Figure 3 shows that if we adjusted the Δs so that both false alarm rates were 1/1000, the commands/intrusion would be about 120 and 140 for $T = 50$ and 100, respectively. However, the preference is reversed at a false alarm rate of .2/1000, raising the commands/intrusion to 293 and 270, respectively. Neither blocksize is uniformly better, and both blocksizes lose quite a bit of detection ability if the false alarm rate must be as low as .1/1000.

Discussion

In order for an intrusion detection tool to be useful, the false alarm rate needs to be low—otherwise alarms tend to be ignored. This can presumably be achieved by setting the control chart parameters as desired and/or by increasing the blocksize. We assume that an operational anomaly detection scheme would also employ other criteria besides command proportions, perhaps based on time-of-day, commands/hour or file access information, and the combined criteria would hopefully have better statistical properties than those based on command proportions alone. In our sample of users, the number of commands generated per day varied quite a bit. The 15000 consecutive commands spanned a period of from 2 days to over 4 months. Most users give between 200 and 2000 commands in a typical working day. This provides some context for our 50- and 100-command blocks. A major strength of the approach presented is its speed. Relatively few operations are needed for computing the test statistic and for updating the control chart parameters; preliminary estimates indicate that it will be easily possible to implement this procedure in real time. We are planning to perform a pilot study of real time monitoring to gather more information on this issue. Currently all software is written in S-PLUS (MathSoft, 1995), an interpreted language programming environment. The pilot project will assess the ability of our S-PLUS implementation to keep up with the accounting flow generated by many users.

Ryan et. al. (1998) use a neural network approach and test classification errors based on 10 users. They have 11 successive days of data, 8 of which are chosen at random and used for training the neural net, which then tried to distinguish users among the other three days. They report a false alarm rate of 7% and 4% missing alarms, based on variable blocksizes equal to a full day of commands for a user. Our test is more challenging in that we test with more users and with smaller blocksizes, and because the test data are collected during a later time period than the training data. On the other hand, unlike them, we excluded users with very low account usage.

See DuMouchel and Schonlau (1998a, 1998b) for descriptions of a comparative study of different test statistics that each use a different approach involving principal components regression and the Fisher score statistic. The current method performs better and requires much less computer storage. Schonlau and Theus (1998) also describe a simple but quite effective test statistic based on a measure of how many users have ever previously used each command–which inspired our extended Dirichlet model.

Control Charts for Intrusion Detection We have provided explicit details of a typical charting procedure in the hopes that some researchers in the field of intrusion detection who are unfamiliar with control charting will be encouraged to integrate their own anomaly detection method into this or a similar control chart scheme. For example, the output of a neural network algorithm could readily take the place of x_i in the control chart, and Figures 1-3 could be constructed for virtually any intrusion detection indicator. If authors of anomaly detection research papers use control charts and ROC plots as a common testing and reporting framework, comparisons of methodology would be greatly facilitated.

Acknowledgement. I would like to thank Matt Schonlau for assembling the data used in the analysis, and the other members of a joint AT&T Labs–Rutgers University–NISS Working Group on Computer Intrusion Detection for valuable advice and discussion.

References

DuMouchel W, Schonlau M (1998a) A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities. *Proceedings of Interface '98*, University of Minnesota, Minneapolis, MN.

DuMouchel W, Schonlau M (1998b) A fast computer intrusion detection algorithm based on hypothesis testing of command transition probabilities. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, Agrawal R, Stolorz P, Piatetsky-Shapiro G, eds., Menlo Park, CA: AAAI Press, pp. 189-193.

Forrest S, Hofmeyr S, Somayaji A, Longstaff T (1996). In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, pp.120-128.

Johnson N, Kotz S (1969) *Distributions in Statistics: Continuous Multivariate Distributions*, New York: John Wiley.

Lunt T, Tamaru A, Gilham F, Jagannathan R, Neumann P, Javitz H, Valdes A, Garvey T (1992) *A Real-Time Intrusion Detection Expert System (IDES)–Final Technical Report*, Computer Science Library, SRI International, Menlo Park, CA.

MathSoft (1995) S-PLUS User's Manual, StatSci Division, MathSoft Inc., Seattle, WA.

Montgomery D (1991) *Introduction to Statistical Quality Control*, 2nd ed., New York: John Wiley.

O'Hagan A (1994) *Kendall's Advanced Theory of Statistics, Vol. 2B: Bayesian Inference*, New York: John Wiley.

Porras P, Neumann P (1997) EMERALD: Event Monitoring Enabling Response to Anomalous Live Disturbances, in *Proceedings of the National Information Systems Security Conference* (to appear).

Ryan J, Lin M, Miikkulainen R (1998) Intrusion detection with neural networks, in Jordan MI, Kearns MJ, Solla SA (eds.) *Advances in Neural Information Processing Systems 10* (NIPS'97, Denver, CO), Cambridge, MA: MIT Press.

Schonlau M, Theus M (1998) Computer intrusion detection based on structural zeros, submitted for publication.