RZ 3003 (# 93049) 03/16/98
Computer Science/Mathematics 11 pages

Research Report

Characterizing Masqueraders for Intrusion Detection

Mehdi Nassehi

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

IBM Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

Characterizing Masqueraders for Intrusion Detection

Mehdi Nassehi

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

A new approach to characterizing masqueraders is presented, which can be used to optimize intrusion-detection systems. With this approach the masquerader characteristics are derived from a sample of user behavior and a parameter called the masquerading factor. The derivation is based on the simplest case of a game-theoretic formulation of the intrusion-detection problem, where the masquerader emulates the user characteristics as closely as its knowledge about the user behavior permits. The extent of this knowledge is represented by the masquerading factor. A rule-based anomaly-detection system is used to demonstrate the proposed characterization approach. Numerical results show that, even though the masquerading factor is not known, the approach allows comparison of system alternatives.

1 Introduction

The optimization of intrusion-detection systems requires not only the user characteristics but also those of the intruder. The intruder masquerades as a user in order to access the system without authorization. In this paper we are concerned with the role of the intruder as a masquerader. In particular, we propose an approach for characterizing masqueraders based on a sample of user behavior and a masquerading factor that represents the extent of knowledge the masquerader has about the user behavior.

In [1], two approaches are presented for obtaining an intruder's characteristics. In the first approach each attribute of the intruder process state is assumed to be uniformly distributed. In the second approach the state attributes are assumed to have the same distribution as those of the user model, but to be independent. Therefore, the intruder model is computed based on the user model. In the approach proposed in this paper, on the other hand, we compute the masquerader characteristics directly from the sample of user behavior and the masquerading factor, which has an immediate intuitive meaning as the extent of knowledge of the masquerader about the user behavior.

The masquerader is not necessarily a human being; it may also be a program [2]. An example is a program that contains a virus but appears to be a virus-free program. The approach presented in this paper can be applied to a masquerader whether it is a human being or a program. The source of uncertainty in the behavior of a program is the uncertainty in the behavior of its user. Furthermore, for a large program, a statistical model may be significantly less complex than the deterministic one.

A case similar to masquerading is an intruder that misuses its own privileges. In a sense, its role as an intruder is to masquerade in its role as a user. Because the intruder can observe its own behavior as long as it wishes, it can adapt its user role closely. For this reason, the approach presented here cannot be applied to the case of misuse of privileges.

Even though in this approach the masquerader is characterized to within a masquerading factor, we will present several examples to show that our approach can provide a valuable means for comparing alternatives in aspects of an intrusion-detection system. These examples are based on the rule-based detection scheme investigated by Chen et al. [3, 4]. In contrast to Chen, who used Prolog, we use tree data structures to extract the rules. We then select a subset of these rules and compute the performance of the detector for different systems as a function of the masquerading factor. Note that these curves allow us to compare the systems, even though we do not know the specific value of the masquerading factor.

This paper is organized as follows. In Section 2 we present the rule-based anomaly detection scheme and observe the need for performance optimization based on user and masquerader characteristics. In Section 3 we present the new approach to characterizing the masquerader based on the sample of user behavior and the masquerading factor. The role of this factor in comparing the performance of different systems is presented in Section 4. The analysis behind the numerical results in Section 4 is contained in the appendix. Section 5 concludes the paper.

2 Motivation

Here the performance optimization problem in intrusion detection is discussed based on the example of a rule-based anomaly-detection scheme. An anomaly-detection scheme can be depicted as in Fig. 1. It consists of a procedure for characterizing the user behavior based on a sample of user behavior obtained through measurement. Based on this characterization a set of detection parameters, which we refer to as the detection function, is generated and

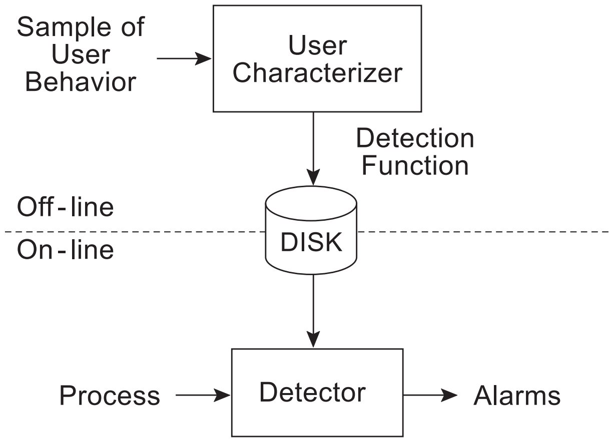

Figure 1: Anomaly-detection scheme.

saved on a disk. Whenever a detector is initiated, this function is loaded into it. The detector monitors a process. Significant deviations in this process from the user characteristics cause the detection function to issue an alarm.

When the anomaly-detection system is rule-based, the process being monitored is a discrete-state discrete-time process and the detection function consists of rules, as shown in Fig. 2. The detection function in the figure consists of only one rule, namely that a symbol A may be followed only by a B or a C. In the process shown, AA violates this rule and results in an alarm.

We use a suffix tree to extract the rules from the sample of user behavior. Suffix trees provide a well-known means for extracting the repeated substrings of a string [5]. Consider the example shown in Fig. 3. The sequence of symbols in the sample of user behavior can be viewed as a string. The suffix tree initially consists of only a root. Then all suffixes of the sample string are added one by one to the tree until the complete suffix tree is obtained. Note that a $, representing an end delimiter, is added to the end of the string in order to ensure that the suffix tree contains one leaf corresponding to each suffix.

The suffix tree is a dictionary of all the substrings of the sample string; any such substring represents a path from the root to a position in the tree. If we remove the branches of the tree that end in a leaf, and necessarily end in a $, then only those substrings are left that occur at least two times in the sample string because they are part of at least two distinct paths leading to two different leaves. Indeed we may use the resulting tree containing these repeated substrings for detection. But we may be able to do better.

It may not be optimal to use all the repeated substrings in the detection function. In fact we shall prune the suffix tree to maximize some measure of the detector performance. To illustrate this, assume that we use only the rules in the tree that are two symbols long. The

Figure 2: Detection function.

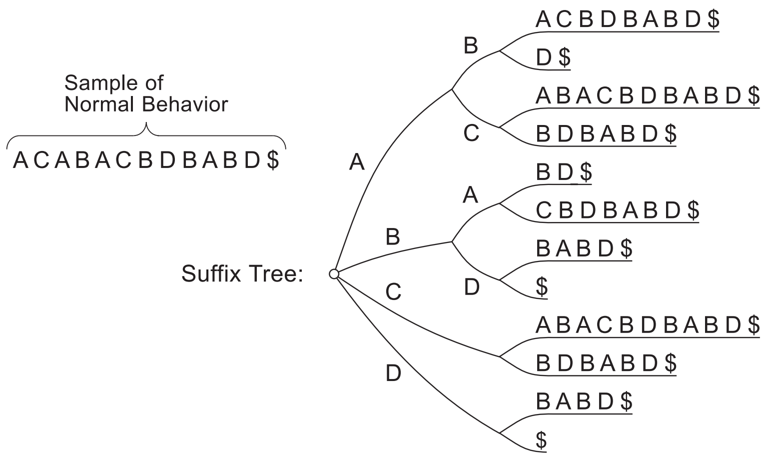

Figure 3: Rule Extraction.

detector is then represented by the finite automaton in Fig. 4, which is similar to the one in [7]. The solid arrows correspond to the rules and the dashed arrows are transitions from a rule to the beginning or middle of the next rule to be checked. For example, when AB is checked, the next rule is either BA or BD, therefore there is a transition from state S_{AB} to state S_B. If the automaton is in state S_A and the next symbol in the process being monitored is an A or a D, an alarm is generated and the detector resets and returns to the initial state S. If the process being monitored has actually been generated by a masquerader, the alarm is a true alarm, otherwise it is a false alarm. The performance of the detector can be defined in terms of true and false alarm rates, where the alarm rate during a time interval is the number of alarms generated divided by the number of symbols monitored. A simple example of the performance measure is the ratio of the true alarm rate to the false alarm rate.

The detection function can be selected to maximize the detector performance. For example, if we shorten the rules by pruning the suffix tree, the alarm rates may either stay the same or drop, and hence may change the performance. Therefore an optimal detection function exists

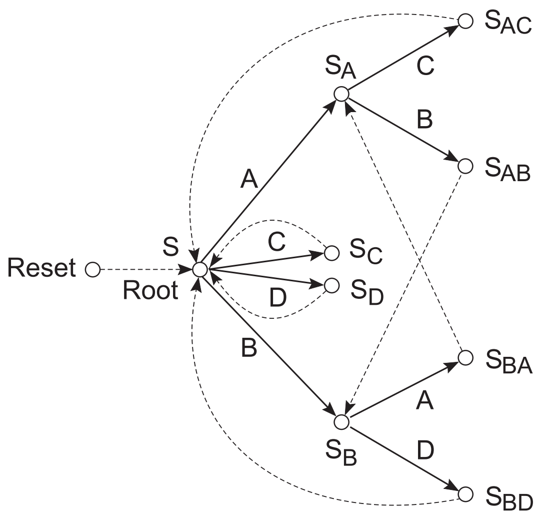

Figure 4: Detector automaton.

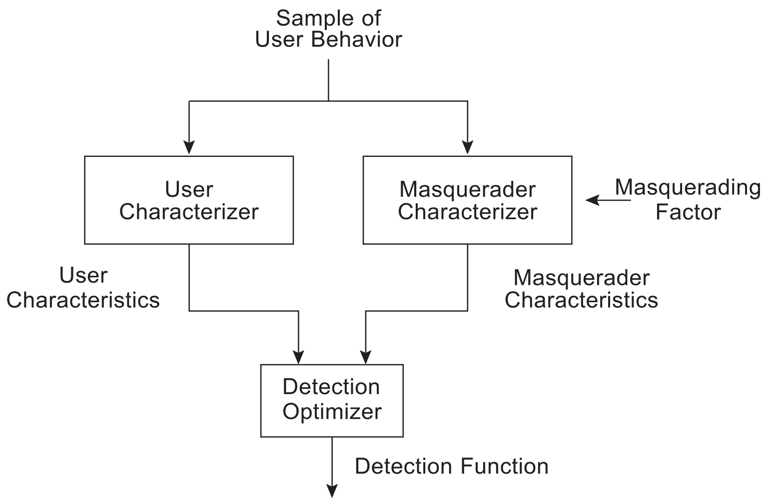

Figure 5: Behavior characterizer.

that maximizes the performance.

To maximize the performance we must compute the true and false alarm rates in terms of the detection function. Let M_U denote a statistical model of the user behavior, where a model is, for example, simply a rule tree with a set of branch probabilities at each node. Let $f(M_U|S_U)$ denote the conditional probability density function of M_U given a sample of user behavior S_U. We also refer to this density function as the characteristics of the process given S_U. This function can be computed based on Bayes' theorem:

$$f(M_U|S_U) = \frac{P(S_U|M_U, l)}{\int P(S_U|M_U, l) dM_U} \quad , \tag{1}$$

where l denotes the length of the given S_U, $P(S_U|M_U, l)$ is the probability of sequence S_U of length l given model M_U, and *a priori* characteristics of the process, $f(M_N|S_N)$, as usual are assumed to be uniform and thus equal to 1 (see the example in the appendix). The value of $P(S_U|M_U, l)$ can be computed by simply walking through the finite automaton of Fig. 4.

In the next section we propose a method to compute the masquerader characteristics from the sample of user behavior and the masquerading factor, which represents the extent of knowledge of the masquerader about the user behavior. The complete behavior characterizer based on this approach is shown Fig. 5.

3 Characterizing Masqueraders

In this section we show how the masquerader characteristics can be obtained from a sample of user behavior and a masquerading factor. Consider the case of anomaly detection. The objective of the masquerader is twofold: to carry an unauthorized action and to avoid or postpone being detected. The objective of the detector is to maximize the performance, which is expressed in terms of true and false alarm rates. Therefore anomaly detection may be viewed as a game with two players: the masquerader and the detector. Each player seeks the strategy for optimizing its objective, which depends on the strategy of the other player. In such a competitive game, one or more stable strategy pairs may exist [6]. However, except for very simple models of masquerader and detector, finding these strategies is analytically complex. Here, we consider only the simplest case, in which the strategy of the masquerader is decoupled from that of the detector. In this case the masquerader subordinates his objective of taking an unauthorized action to that of masquerading. It therefore behaves to the best of its knowledge in such a way that the characteristics of its behavior matches those of the user behavior. We represent the masquerader's knowledge by a fictitious sample of user behavior that it possesses. In particular, we refer to the size of this sample as the masquerading factor m. The masquerader and user characteristics from the detector's point of view are then used to optimize the detector function, which defines the optimal detector's strategy.

Therefore, to obtain the optimal detection function, we need to obtain the masquerader characteristics from the detector's point of view. But first we have to obtain the masquerader's characteristics from its own point of view. Let S'_U denote the fictitious sample of user behavior that the masquerader possesses, and M'_U denote the user model from the masquerader's point of view. Then, as in equation (1), the user characteristics from the masquerader's point of view are given by

$$f(M_M|S'_U) = f(M'_U|S'_U) = \frac{P(S'_U|M'_U, m)}{\int P(S'_U|M'_U, m) dM'_U} \quad , \tag{2}$$

where the first equality indicates that the masquerader emulates the user characteristics from its point of view. Now, we observe that the distribution of the fictitious sample of user behavior, S'_U, can be expressed in terms of its length, m, and a given user model from the user's point of view, M_U. This dependency is denoted by $P(S'_U|M_U, m)$. Combining this with $f(M_U|S_U)$ and $f(M'_M|S'_U)$ from equations (1) and (2), respectively, and unconditioning on S'_U and M_U, we obtain

$$f(M_M|S_U, m) = \int \sum_{S'_U} f(M_M|S'_U) P(S'_U|M_U, m) f(M_U|S_U) dM_U \quad . \tag{3}$$

The above equation expresses the characteristics of the masquerader behavior in terms of the sample of user behavior and the masquerading factor. This equation can be used for every process in which both state and time are discrete. If one or both of them are continuous, we have to replace the distribution $P(S'_U|M_U, m)$ with the density function $f(S'_U|M_U, m)$.

4 Application

In the previous section we characterize the masquerader's behavior to within a masquerading factor. We now show that even though the masquerading factor is not known, this characterization can assist us in comparing different alternatives for aspects of systems. The numerical results are based on the analysis contained in the appendix. The examples are for the rule-based anomaly-detection system described in Section 2.

In the first example we examine the effect of the optimization of the rule tree on the performance of the detector. We use the sample of user behavior given in Fig. 6(a). This sample is generated based on the probability-transition matrix in Fig. 6(b). We consider two cases. In the first case we extract from this sample a rule tree truncated to a depth of three symbols. We then compute the user characteristics and the masquerader characteristics for a given masquerading factor. Using these characteristics we compute the normalized performance defined as $(R_T/R_F)/(P_M/P_U)$, where R_T (R_F) is the true (false) alarm rate, and P_M (P_U) is the probability that the process is generated by the masquerader (user). In the second case we optimize the rule tree by selecting among all those trees with a depth not exceeding three symbols the one that results in the highest detector performance. The

DCBADCADCABCDACB
DABDABCADCBADCAB
CADBCADBCADBADCB
DCBDABCDABCABDCA
DCABDCBACDBCDACD
ABDCABCADCADCABD
ABCABCDACBADBADC
ADCBACDACBDACBAD
CBDABDABDACDACBA
DCBDCADCBACBACBA
CBDACBDACDBADCAB
CADCBADBACDABDAC
BDCABD$

(a) Sample.

State	Next Symbol A	B	C	D
AB	0	0	0.5	0.5
AC	0	0.5	0	0.5
AD	0	0.5	0.5	0
BA	0	0	0.5	0.5
BC	0.5	0	0	0.5
BD	0.5	0	0.5	0
CA	0	0.5	0	0.5
CB	0.5	0	0	0.5
CD	0.5	0.5	0	0
DA	0	0.5	0.5	0
DB	0.5	0	0.5	0
DC	0.5	0.5	0	0

(b) Probability Transition Matrix.

Figure 6: Symmetric user behavior.

performance of these two cases as a function of the masquerading factor is shown in Fig. 7. As expected, the detector performance is higher for smaller masquerading factors. Moreover, the improved performance achieved by an optimal detector becomes more substantial for smaller masquerading factors. Overall, we observe that the optimization of the detector can provide significant performance benefits.

In the second example, we examine the effect of the sample of user behavior on the detector performance. We compare the performance resulting from the sample in Fig. 6(a) with that resulting from the sample in Fig. 8(a).

Note the probability-transition matrix of the sample in Fig. 6 is symmetric, whereas that in

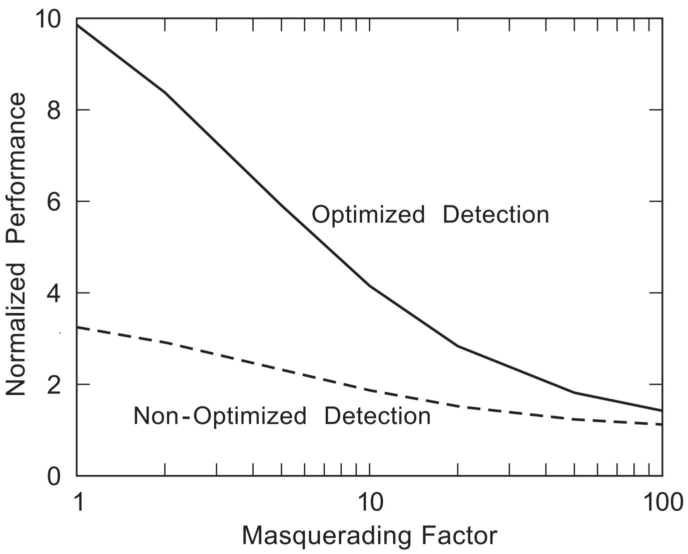

Figure 7: Effect of optimization on performance.

DABCADCBACBACBDA
BCADBADCABCABDAB
CABCABCABDCABDAC
DABCABCADBACBADC
ADCBACBACDABCABC
ADCABCABCDABDACB
DABCABDABCABCABC
ABCADBACBADCBDAC
BADBACBACBADCABC
ABDABDABCABDACDA
BDABCABDABCABDAC
BADBACBACBACBACB
ADBACB$

(a) Sample

	Next Symbol			
State	A	B	C	D
AB	0	0	0.7	0.3
AC	0	0.8	0	0.2
AD	0	0.7	0.3	0
BA	0	0	0.7	0.3
BC	0.9	0	0	0.1
BD	0.8	0	0.2	0
CA	0	0.8	0	0.2
CB	0.9	0	0	0.1
CD	0.7	0.3	0	0
DA	0	0.7	0.3	0
DB	0.8	0	0.2	0
DC	0.7	0.3	0	0

(b) Probability Transition Matrix.

Figure 8: Asymmetric user behavior.

Fig. 8 is asymmetric. As in the previous example we compute the optimal detection function for each of these two samples. The detector performance for each of these samples is shown in Fig. 9. We observe that the symmetric process provides better performance, especially for small masquerading factors. Such a result is beneficial in an intrusion-detection system in which, because of limited amount of resources, we have to choose from a number of processes for use in the anomaly-detection subsystem.

Comparison of different degrees of detector optimization and the comparison of different samples of user behavior are but two examples of how the proposed approach of characterizing masquerader behavior can be beneficial even though the masquerading factor is not known.

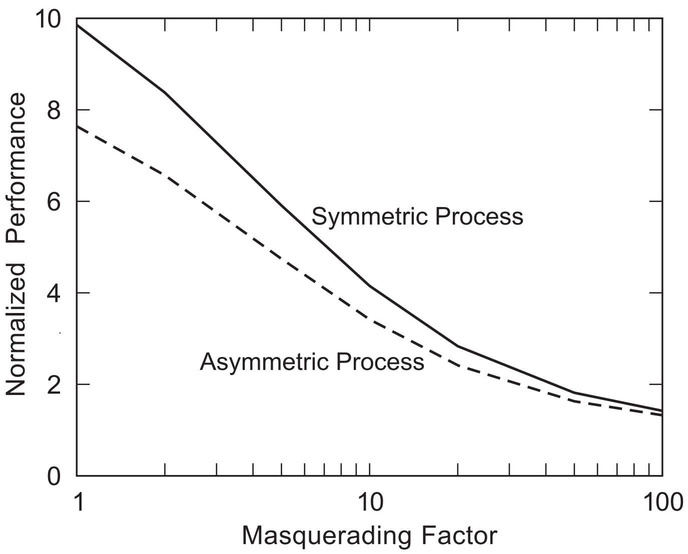

Figure 9: Effect of process symmetry on performance.

We can use the same approach to compare two model classes. For example we can compare the tree-based approach discussed here with a model based on fixed blocks of the sample. Or we can use the same approach to compare detectors that use the tree-based model in different ways. For example, as described here, one detector may in every state generate an alarm for every unexpected symbol. Another detector may operate primarily on long-term averages. It may maintain the frequency at which the branches of the tree are traversed by the monitored processes and generate an alarm in case of large deviations from a nominal value. In all these cases the plots of detector performance versus masquerading factor can be of benefit.

5 Conclusion

An approach was proposed here for characterizing a masquerader in terms of a sample user behavior and a masquerading factor. The approach is based on a special case of a game-theoretic formulation of the anomaly-detection problem. In this special case the masquerader adapts its behavior to its best knowledge of the user characteristics. We applied this approach to a rule-based anomaly-detection system and obtained numerical results for the detector performance. We observed that even though the masquerading factor is not known, such numerical results can be valuable for comparing various alternatives in the system.

6 Appendix

Here we present the analysis behind the numerical results in Section 4. There we chose the ratio of the true alarm rate to the false alarm rate as the measure of performance. Let R_T (R_F) denote the true (false) alarm rate. Let P_I (P_N) denote the probability that the process under observation is generated by a masquerader (user). As was described in Section 2, after the detector automaton generates an alarm, it returns to the root state. Let r (s) denote the mean number of symbols to the first true (false) alarm rate starting from the root state, S. We then have $R_T = P_I/r$ and $R_F = P_N/s$.

Now we compute r and s. We denote a node in the finite automaton by the sequence of the symbols on the path from the root to that node, as shown in Fig. 4. Let s_a denote the mean number of symbols to the first false alarm starting from the state S_a. If we let p_{ab} denote probability, given that the current state of the automaton is S_a and the process being monitored is the user process, the next symbol corresponds to the outgoing branch b. First consider how s_A is computed. The automaton is in state S_A. The next symbol may be A, B, C, or D. If it is B, no alarm is generated and the automaton proceeds to S_B. In that case s_A would be equal to $1 + s_B$. Similarly, if the next symbol is C, s_A would be equal to $1 + s$. If the next symbol is A or D, however, there would be a false alarm and s_A would be zero. Therefore,

$$s_A = p_{AA}.0 + p_{AB}(1 + s_B) + p_{AC}(1 + s) + p_{AD}.0 \quad .$$

Computing s_a for other states, we obtain

$$\left\{ \begin{array}{lcl} s & = & p_A(1 + s_A) + p_B(1 + s_A) + (p_C + p_D)(1 + s) \\ s_A & = & p_{AB}(1 + s_B) + p_{AC}(1 + s) \\ s_B & = & p_{BA}(1 + s_A) + p_{BD}(1 + s) \end{array} \right. \quad .$$

Solving this set of equations we obtain the value of s. Similarly, we can compute r based on q_{ab}, defined like p_{ab} but for the masquerader process. The set of equations will look identical. Note that the above set of equations represents an approximation by which we take the mean of branch probabilities before computing s and r. This approximation becomes accurate as the size of the sample of user behavior increases. The expected value is taken with respect to the distribution of the branch probabilities.

We now compute p_{ab} and q_{ab} for $S_a \in R$ and $b \in B$, where R denotes the set of states and B the set of symbols observed in the sample of user behavior. For this computation we assumed that the suffix tree is not a compact digital tree in that, if for example there is a branch XY, we assume a quasistate between X and Y with a single outgoing branch Y. The probability of this outgoing branch is close to but not exactly one, even though in the sample of the user behavior this branch X was always followed by Y.

We assume at every state or quasistate the *a priori* probabilities of the next symbol is b is $1/|B|$ for b belonging to B. We also assume that the distributions of branch probabilities of every pair of nodes are independent.

In the remainder of this appendix we show that

$$p_{ab} = \frac{n_{ab}+1}{l+|B|} \qquad S_a \in R, \quad b \in B \quad ,$$

and

$$q_{ab} = \frac{mp_{ab}+1}{m+|B|} \qquad S_a \in R, \quad b \in B \quad ,$$

where n_{ab} denotes the number of occurrences of branch b at state S_a in the sample of user behavior, l denotes the length of the sample of user behavior, and m is the masquerading factor.

Consider l trials where each has an outcome belonging to the set $\{1, 2, \ldots, |B|\}$. Let $\underline{n}$ denote the vector $(n_1, n_2, \ldots, n_{|B|})$, where n_i is the number of occurrences of outcome i. Let $\underline{\theta}$ denote the probability vector $(\theta_1, \theta_2, \ldots, \theta_{|B|})$, where θ_i is the probability of outcome i. We define $\underline{n}! = n_1! n_2! \cdots n_{|B|}!$ and $\underline{\theta}^{\underline{n}} = \theta_1^{n_1} \theta_2^{n_1} \cdots \theta_{|B|}^{n_{|B|}}$, and

$$\int f(\underline{\theta})\, d\underline{\theta} = \int_0^1 \int_0^{1-\theta_1} \cdots \int_0^{1-\theta_1-\theta_2-\cdots-\theta_{|B|-2}}$$
$$f(\theta_1, \theta_2, \ldots, \theta_{|B|-1}, 1-\theta_1-\theta_2-\cdots-\theta_{|B|-1})\, d\theta_{|B|-1} \cdots d\theta_2\, d\theta_1 \quad .$$

Using integration by parts and induction, it can be shown that

$$\int \underline{\theta}^{\underline{n}}\, d\underline{\theta} = \frac{\underline{n}!}{(l+|B|-1)!} \quad . \tag{4}$$

The conditional probability of sample $\underline{n}$ is given by

$$P(\underline{n} \mid \underline{\theta}, l) = \begin{pmatrix} l \\ \underline{n} \end{pmatrix} \underline{\theta}^{n} \quad . \tag{5}$$

From (1) we obtain

$$f(\underline{\theta} \mid \underline{n}) = \frac{\underline{\theta}^{\underline{n}}}{\int \underline{\theta}^{\underline{n}} d\underline{\theta}} = \frac{(l+|B|-1)!}{\underline{n}!} \underline{\theta}^{\underline{n}} \quad . \tag{6}$$

The estimate of the branch probabilities for the user is, for $i = 1, 2, \ldots, |B|$

$$p_i = \int \theta_i f(\underline{\theta} \mid \underline{n}) d\underline{\theta} = \frac{n_i+1}{l+|B|} \quad . \tag{7}$$

Now, for $i = 1, 2, \ldots, |B|$, we compute

$$q_i = \int \theta_{M,i} f(\underline{\theta}_I \mid \underline{n}_U, m) d\underline{\theta}_M \quad .$$

Equation (3) adapted to this case is

$$f(\underline{\theta}_M \mid \underline{n}_U, m) = \int \sum_{\underline{n}_M} f(\underline{\theta}_M \mid \underline{n}_M) P(\underline{n}_M \mid \underline{\theta}_U, m) f(\underline{\theta}_U \mid \underline{n}_U) d\underline{\theta}_U \quad .$$

Equation (6) is

$$f(\underline{\theta}_M \mid \underline{n}_M) = \frac{(m+|B|-1)!}{\underline{n}_M!} \underline{\theta}_M^{\underline{n}_M} \tag{8}$$

Equation (5) is

$$P(\underline{n}_M \mid \underline{\theta}_U, m) = \begin{pmatrix} m \\ \underline{n}_M \end{pmatrix} \underline{\theta}_U^{\underline{n}_M} \quad . \tag{9}$$

For every function $g(.)$ and $i = 1, 2, \ldots, |B|$, we have

$$\int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_U, m) d\underline{\theta}_M =$$

$$\sum_{\underline{n}_M} \int f(\underline{\theta}_U \mid \underline{n}_U) P(\underline{n}_M \mid K_M, \underline{\theta}_U) d\underline{\theta}_U \int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_M) d\underline{\theta}_M \quad . \tag{10}$$

Inserting (8) and (9) into (10), we obtain

$$\int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_U, m) d\underline{\theta}_M =$$

$$\sum_{\underline{n}_M} \int \frac{(l+|B|-1)!}{\underline{n}_U!} \begin{pmatrix} m \\ \underline{n}_M \end{pmatrix} \underline{\theta}_U^{(\underline{n}_U+\underline{n}_M)} d\underline{\theta}_U \int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_M) d\underline{\theta}_M \quad . \tag{11}$$

Using (4) we obtain

$$\int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_U, m) d\underline{\theta}_M =$$

$$\frac{m!(l+|B|-1)!}{n_U!(m+l+|B|-1)!} \sum_{\underline{n}_M} \frac{(\underline{n}_U+\underline{n}_M)!}{\underline{n}_M!} \int g(\theta_{M,i}) f(\underline{\theta}_M \mid \underline{n}_M) d\underline{\theta}_M \quad .$$

By setting $g(\theta_{M,i})$ to one, we obtain

$$1 = \frac{m!(l+|B|-1)!}{n_D!(m+l+|B|-1)!} \sum_{\underline{n}_M} \frac{(\underline{n}_U+\underline{n}_M)!}{\underline{n}_M!} \quad .$$

We define L as follows:

$$L(m, l, \underline{n}_U) = \frac{n_U!(m+l+|B|-1)!}{m!(l+|B|-1)!} \quad .$$

By rearranging the limits of the following summation we observe that

$$\sum_{\underline{n}_M} \frac{(\underline{n}_U+\underline{n}_M)!}{\underline{n}_M!} n_{M,i} = L(m-1, l+1, \underline{n}_U + I_i) \quad ,$$

where I_i is a vector of $|B|$ elements with the ith element 1 and others 0. By setting $g(\theta_{M,i})$ to $\theta_{M,i}$ in equation (10) we obtain

$$q_i = L^{-1}(m, l, \underline{n}_U) \sum_{\underline{n}_M} \frac{(\underline{n}_U+\underline{n}_M)!}{\underline{n}_M!} \times \frac{n_{M,i}+1}{m+|B|} \quad .$$

Using Eq. (10) above yields for $i = 1, 2, \ldots, |B|$

$$q_i = \frac{L^{-1}(m, l, \underline{n}_U) L(m-1, l+1, \underline{n}_U + I_i) + 1}{m+|B|} = \frac{mp_i + 1}{m+|B|} \quad .$$

References

[1] P. Helman and G. Liepins, "Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Trans. Software Eng.*, vol. 19, no. 9, Sept. 1993, pp. 886–901.

[2] S. Forrest, S. A. Hofmeyr and A. Somayaji, "Computer Immunology," *Communication of the ACM*, October 1997, pp. 88–96.

[3] K. Chen, "An Inductive Engine for the Acquisition of Temporal Knowledge," Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1988.

[4] H. S. Teng, K. Chen and S. C-Y Lu, "Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns," *Proc. IEEE Symp. on Res. in Sec. & Privacy*, pp. 278-84, Oakland, 1990.

[5] G. A. Stephen, *String Searching Algorithms* (World Scientific, Singapore, 1994).

[6] G. Owen, *Game Theory*, second ed. (Academic, New York, 1982).

[7] A. V. Aho and M. J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," *Communications of the ACM*, vol. 18, no. 6, pp. 333-40, June 1975.