Sequence Matching and Learning in Anomaly Detection for Computer Security

Terran Lane and **Carla E. Brodley**
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285
Phone: +1 765 494-3462
email: {terran,brodley}@ecn.purdue.edu

Abstract

Two problems of importance in computer security are to 1) detect the presence of an intruder masquerading as the valid user and 2) detect the perpetration of abusive actions on the part of an otherwise innocuous user. We have developed an approach to these problems that examines sequences of user actions (UNIX commands) to classify behavior as normal or anomalous. In this paper we explore the matching function needed to compare a current behavioral sequence to a historical profile. We discuss the difficulties of performing matching in human-generated data and show that exact string matching is insufficient to this domain. We demonstrate a number of partial matching functions and examine their behavior on user command data. In particular, we explore two methods for weighting scores by adjacency of matches as well as two growth functions (polynomial and exponential) for scoring similarities. We find, empirically, that the optimal similarity measure is user dependant but that measures based on the assumption of causal linkage between user commands are superior for this domain.

Keywords: Application, Sequence learning, Partial matching, Classification, Recognition, Computer security, Anomaly detection, Behavioral modeling.

Introduction

A long-standing problem in the field of computer security is that of *intrusion detection.* The goal is to detect violations of security policy for a computer site by an outsider. Of the many possible approaches to intrusion detection, one that has received considerable attention is *anomaly detection* (Anderson, 1980; Lunt, 1990; Heberlein, Dias, Levitt, Mukherjee, Wood & Wolber, 1990). According to Kumar (1995), "Anomaly detection attempts to quantify the usual or acceptable behavior and flags other irregular behavior as potentially intrusive." Under this definition, the scope of anomaly detection encompasses not only violations by an outsider but also anomalies arising from violations on the part of an authorized user. It is important to note that anomaly detection omits the class of security policy violations that occur within the bounds of normal behavior for a system or site. Detecting anomalous behavior can be viewed as a binary valued classification problem in which measurements of system activity such as system log files, resource usage, command traces, and audit trails are used to produce a classification of the state of the system as normal or abnormal.

In other work (Lane & Brodley, 1997) we have presented a system intended to address the anomaly detection domain. Our system learns a *user profile* and subsequently employs it to detect anomalous behavior. Based on sequences of actions (UNIX commands) of the current user's input stream, the system classifies current behavior as consistent or anomalous with past behavior. The anomaly detection domain presents a number of interesting problems including learning from positive examples only, handling concept drift, and measuring similarity between sequences of human actions. We demonstrated that the closed world assumption can be used to address the positive training examples difficulty, and that user behaviors can be distinguished under a particular definition of behavioral similarity (Lane & Brodley, 1997).

In this paper we explore the issue of similarity measurement — matching a current behavioral pattern with historical behavior. We demonstrate a number of possible matching functions and examine some of the behavioral differences among them.

Capturing the Causal Nature of User Actions

Traditionally, in computer security, user profiles have been built based on characteristics such as resources consumed, typing rate, command issue rate, and counts of particular commands employed (Denning, 1987; Smaha, 1988; Lunt, 1990; Frank, 1994). These approaches do not use the observation that human/computer interaction is essentially a *causal process.* Typically, a user has a goal to achieve when using the computer, which causes the person to issue certain commands, causing the computer to act in a certain manner. The computer's response, in turn, keys further actions on the part of the human.

User profiling is a widely studied problem in the literatures of computer security and machine learning.

In the area of security, the applications of behavioral modeling are in anomaly detection (as described previously) and in software forensics (Spafford & Weeber, 1992). Krsul and Spafford (1996) have examined the topic of authorship analysis and used a set of software metrics such as mean line length, placement of syntactic structures, ratio of global to local variable counts, and ratio of white lines to code lines to identify program authors. Such metrics focus on global statistics but do not address causal sequences of actions. Furthermore, they allow an encoding of domain knowledge (in terms of the higher-level features employed) but may overlook other characteristics present in the data. Machine learning researchers have studied modeling of human behavior in the contexts of, for example, automatic tutoring (Baffes & Mooney, 1996). This work models a student's understanding of a subject as a set of Horn-clauses which defines deviations from the correct theory of the subject. Such a system can work well when the domain theory is sufficiently understood to induce such rules from behavior, but this is often not the case in the context of arbitrary command processing systems (a sufficiently thorough domain theory for this context might well include a complete model of the system state — a computationally infeasible proposition).

To form a user profile our approach learns *characteristic sequences* of actions generated by users. The underlying hypothesis is that a user responds in a similar manner to similar situations, leading to repeated sequences of actions. Indeed, the existence of command alias mechanisms in many UNIX command interpreters supports the idea that users tend to perform many repeated sets of actions, and that *these sequences differ on a per-user basis.* It is the differences in characteristic sequences that our approach uses to differentiate a valid user from an intruder masquerading as that user. Note that the detection of anomalous behavior is made more difficult because an informed, malicious intruder may attempt to emulate the valid user's behavior, including alias and command usage.

To represent characteristic patterns of actions, our system uses the *sequence* (an ordered, fixed-length set of temporally adjacent actions) as the fundamental unit of comparison. For this research, actions were taken to be UNIX shell commands with their arguments, although the approach developed here is general and can be extended to any stream of discrete events such as operating system calls or graphical user interface events.

A set of behavioral sequences (gathered from a, presumably, intruder-free history) is collected into a *dictionary* of sequences. This dictionary, along with system parameters such as sequence length, instance selection policy, noise-suppression function, and classification threshold, forms a user profile. Any new sequence under scrutiny can be compared to the dictionary (as described below) to yield a measure of 'familiarity' or 'similarity' to past behavior. This similarity measure is then used as the basis for classification of the sequence as anomalous or normal.

We envision our anomaly detection system as part of a personal software assistant that helps monitor a user's account for penetrations. We hope that, in conjunction with techniques that employ more domain-specific knowledge (such as rule-base systems or systems employing user-selected higher-level features), our techniques can provide a basis for greater security and privacy for users and sites. Because of privacy issues, and the fact that it is impossible to characterize the full space of user behaviors, only positive examples of the account owner's behavior are available for training.

Norton has explored sequence learning for DNA sequences (Norton, 1994), but his data had both positive and negative training examples. The anomaly detection domain differs from traditional concept formation tasks in that one must characterize user behavior from "positive" examples only. To resolve this difficulty we invoked the *closed world* assumption — that anything not seen in the historical data represents a different user. Intuitively, it seems likely that this is a reasonable assumption — the very terms anomaly, abnormal, and unusual imply that divergence from past behavior is an important indication of trouble.

It is interesting to note that a separate research group has independently developed an anomaly detection technique based on examination of sequences (Forrest, Hofmeyr, Somayaji & Longstaff, 1996a; Forrest, Hofmeyr & Somayaji, 1996b). Their work, while similar in intent, focuses on monitoring the behavior of privileged programs (UNIX system daemons) to detect exploitation of services by malicious users. Monitoring processes entails a different set of problems and goals than does monitoring users. Their system examines the sequences of system calls emitted by running processes. Both the set of possible system calls and the possible orderings of those calls are from a more restricted set than the space of user-level commands and command sequences, so strict equality matching has a much higher probability of yielding useful results. Furthermore, system calls are emitted by system services much more quickly than are commands by users (potentially 10^7 times as quickly). Thus, many fewer resources can be devoted to processing each sequence of system calls than can be devoted to processing each sequence of user commands. It is particularly intriguing that this research group arrived at a sequence based detection system through a model of the human immunological system, while our system has its origins in a behavioral model. The degree of convergence in the systems suggests the possibility of congruences in the underlying models and, perhaps, even in the original human systems.

Detecting Anomalous Behavior

Once a user profile is formed, the basic action of the detection system is to compare incoming input sequences to the historical data and form an opinion as to whether or not they both represent the same user. The fundamental unit of comparison in the anomaly detector system is the command sequence. Dietterich and Michalski (1986) have studied the problem of learning to predict sequences by fitting sequence data to a model from a space of possible models. Their goal was to create a system that could predict subsequent actions in the sequence, whereas our goal is to classify sequences of new actions as consistent or inconsistent with sequence history. To this end, all input token streams are segmented into overlapping sequences of tokens (where the *length* of each sequence is a parameter to the system, but is fixed for a single run). Two sequences can then be compared using a similarity measure.

Computing Sequence Similarity

One approach to learning from sequence data is to convert the data into feature vectors by accumulating measures of the individual sequences (Hirsh & Japkowicz, 1994; Salzberg, 1995). Then one can apply any off the shelf classifier construction algorithm such as a neural network or a decision tree to the feature vectors that describe the sequence data. By contrast, our approach uses a measure of similarity between sequences to compare current input to historical data. Direct analysis of the user's command stream avoids introduction of the domain knowledge required to construct higher-level features. Our hypothesis is that these two classes of approaches are complimentary and that a system incorporating may well be more effective than either individually.

A number of possible methods exist for measuring the similarity of two sequences. The most straightforward is the equality function, which yields `TRUE` when both sequences match in every position and `FALSE` otherwise. This is the similarity function employed by string matching algorithms and has the advantage of being widely studied and highly optimizeable. For example, the UNIX `diff` program employs this form of matching. Srikant and Agrawal (1996) use a modified equality matching function to detect frequently occurring sequences in large data sets; they allow gaps, or intervening non-matching elements, in their sequence detection. In spite of the (presumed) causal nature of human-generated command sequences, we do not expect exact matches for reasonably long sequences because of unpredictable and asynchronous events (arrival of mail, telephone calls, etc.), which can be viewed as noise distorting the causally-driven process. Thus, the equality function is not a viable choice for this particular domain. Equality matching can, however, be a viable technique for other aspects of anomaly detection in computer security, as demonstrated by Forrest, et al. (1996b).

Our system, therefore, computes a numerical *similarity measure* that returns a high value for pairs of sequences that it believes to have close resemblance, and a low value to pairs of sequences that it believes largely differ. The individual elements of the sequences are from an unordered set, which creates a matching problem identical to that of symbolic features for IBL. However, unlike IBL, our similarity measure is judging the similarity between two *sequences* rather than two feature vectors.

Initially, we examined a similarity measure that simply assigns a score equal to the number of identical tokens found in the same locations of the two sequences. Upon consideration, however, we hypothesized that a measure that assigns a higher score to adjacent identical tokens than to separated identical tokens might be preferable. The intuition is that token matches separated by interleaving non-matching tokens are more likely to have occurred by chance, while adjacent matches are more likely to have occurred due to a causal process. Therefore, if sequence Seq_1 has k tokens in common with each of Seq_2 and Seq_3, but the common tokens are adjacent in Seq_1 and Seq_2 then we would like the similarity measure to have the property that $\mathrm{Sim}(Seq_1, Seq_2) > \mathrm{Sim}(Seq_1, Seq_3)$. Under this requirement, the pair of sequences shown below on the left would have a higher similarity value than would the pair on the right.

```
ls foo ; vi                    ls -l foo ;
ls foo cat bar                 ls -F foo cat
```

Thus, one axis of differentiation between similarity measures is 'does not detect match adjacency' vs. 'detects match adjacency'. A second axis is the bound of the maximum similarity measure as a function of sequence length. The similarity measure that scores sequences from a count of matches, regardless of adjacency, has an upper bound that is polynomial in the length of the sequences. Specifically, for sequences of length n, this measure is $\leq n$. We denote this similarity measure as MC-P (for match count with polynomial bound). To examine the hypothesis that detecting match adjacency is useful for this task, we modified MC-P to bias the similarity score in favor of adjacent matches (as described below). This measure is denoted MCA-P (for match count with adjacency and polynomial bound) and is bounded by $n(n+1)/2$. A polynomial bound seems appropriate, considering that the central hypothesis is that adjacent tokens are produced by a (mostly) causal process, and, therefore, should display a high degree of correlation.

In a stream of independently generated tokens, it seems likely that an exponentially bounded function would be more appropriate. Our intuition is that the causal linkage of user-generated tokens will evidence itself as a deviation from the characeristics of the independence assumption. To test our hypothesis (i.e. reject the independence hypothesis), we examined the

	$f(\mathrm{Sim}, c)$	$u(c)$
MC-P	$\mathrm{Sim} + 1$	1
MCA-P	$\mathrm{Sim} + c$	$c + 1$
MC-E	$2 \cdot \mathrm{Sim}$	1
MCA-E	$\mathrm{Sim} + c$	$2 \cdot c$

Table 1: scoring and update functions for the similarity measures

behavior of exponentially bounded measures analogous to the polynomially bounded ones just described. The MC-P measure was modified to score exponentially in the number of matches (still without considering adjacency) and was labeled MC-E. The MC-E measure has upper bound 2^n. The MCA-P measure was adapted in a similar fashion to create MCA-E, whose upper bound is $2^n - 1$. All four similarity measures are encompassed by the following algorithm operating on sequences Seq_1 and Seq_2:

- Set an adjacency counter, $c := 1$ and the value of the measure, $\mathrm{Sim} := 0$.
- For each position, i, in the sequence length, n:
  - If $Seq_1(i) = Seq_2(i)$ then $\mathrm{Sim} := f(\mathrm{Sim}, c)$ and $c := u(c)$
  - Otherwise, $c := 1$.
- After all positions are examined, return the measure value.

The differences between the measures are determined by the nature of the scoring function, $f(\mathrm{Sim}, c)$, and the adjacency update function, $u(c)$, as summarized in Table 1.

Finally, we define the similarity of a single sequence Seq_i to a set of sequences, D, as:

$$\mathrm{Sim}(Seq_i, D) = \max_{Seq_j \in D} \{\mathrm{Sim}(Seq_i, Seq_j)\}$$

Thus, the similarity of a sequence to the user dictionary is the measure of that sequence compared to the *most* similar sequence in the dictionary.

Classifying User Behavior

Given an input stream of command tokens parsed by the data collection module (as described in (Lane & Brodley, 1997)), the detection module classifies the current user as normal or anomalous after each token. The output of the detection module is a stream of binary decisions indicating, at each point in the input command data, whether or not it believes that the input stream at that point was generated by the profiled user.

To make these decisions, the detection module first calculates the similarity of each input sequence to the user's dictionary, yielding a stream of similarity measures. In an intuitive sense, this stream represents the familiarity of the input commands at each time step, given knowledge about the previous behavior of the user. The similarity stream is smoothed to reduce noise (Lane & Brodley, 1997) and classification is performed for each time step. In the current implementation the classification is made with a threshold decision: if the current smoothed similarity measure is between maximum and minimum allowable bounds, then classify the current time step as normal, otherwise classify it as abnormal. The thresholds are system parameters and were selected empirically by examining an independent 'parameter selection' data set drawn from the command history of the profiled user. After computing sequence similarity to the user profile (under the currently selected similarity measure) for each item in the parameter selection data set, the thresholds were selected such that a proportion of *err_rate* samples fell outside the thresholds (specifically, *err_rate*/2 below the minimum threshold and *err_rate*/2 above the maximum threshold). A smaller allowed error rate, therefore, corresponds to a wider range of acceptable similarities. We acknowledge that this classification scheme is relatively unsophisticated, but it turns out to perform surprisingly well in many cases. We are currently investigating employing other non-parametric classification techniques, such as clustering with Parzen windows (Fukunaga, 1990), to this task.

Examination of Similarity Measures

To examine the degree of class separation produced by each of these similarity measures, we used each to classify user command history traces. The data were command histories collected from five members of the Purdue MILLENNIUM lab over the course of slightly more than an academic semester and two sets donated by other students. From each user's data set, two thirds of the tokens were devoted to training (i.e. initial dictionary construction) and the remaining one third was divided into 1000 testing instances and the rest into instances used for dictionary instance selection and parameter selection. The user profiles were initialized with all available training data and were pruned down to the desired testing sizes of 200, 500, or 1000 sequences via the LRU instance selection algorithm as described in (Lane & Brodley, 1997). The LRU (Least Recently Used, by analogy with the page replacement algorithm of the same name) prunes instances from the dictionary by examining the timestamps of their last reference (the last point at which they were selected as 'most similar' to an instance under examination) and removing those with the oldest timestamps.

All experiments employed a sequence length of ten tokens and a smoothing window length of eighty sequences. These values were selected based on previous experimentation (Lane & Brodley, 1997). Profiles were created for four of the users and then the test data from all seven users were classified against the profiles according to each similarity measure. The classification thresholds were selected to achieve a false-negative error rate of 1% on the parameter selection data (as detailed above). Results of the experiments are displayed in Tables 2 and 3. The results given here are typical and are generally reflective of the trends we found in the data.

Each value in this table reports the percentage of instances for which the tested user was identified as the profiled user. The goal, therefore, is to minimize all rows other than the profiled user and to maximize the row corresponding to the profiled user (USER0, in Table 2 and USER2, in Table 3). In Table 3, the symbol '†' indicates the position in which the best value occurs for each user. As above, MC and MCA denote match count and match count with adjacency bias similarity measures, respectively, while the suffixes P and E indicate that the similarity measure's upper bound is polynomial or exponential in the length of the sequence.

Table 2 indicates a definite superiority of MC-P for the task of identifying the profiled user (USER0) but somewhat inferior behavior on the task of distinguishing other users from USER0. By contrast, all other algorithms perform spectacularly when distinguishing other users from the profiled user but have what is likely to be an unacceptably high false negative rate. Unfortunately, the behaviors of many of the similarity measures are nearly identical for large parts of the tested space, so it is difficult to identify their relative merits from this data. Nonetheless, for USER0's profile, the MC-P similarity measure seems to be preferable. This result provides evidence against the hypothesis that detecting adjacent matches is desirable.

The results in Table 3 indicate that, overall, MCA-P is the preferable similarity measure for USER2's profile, followed by MCA-E. A strong preference for the adjacency-based measures is supportive of the hypothesis of the causality of command sequences. And, as with USER0, preference for the polynomial bounded similarity measure further supports this hypothesis. It's also noteworthy that the MC-P algorithm has the poorest performance for this user (often dramatically so). This indicates that there are cases in which equality matching (with the addition of 'don't care' positions) is insufficient to the task of user modeling in this context.

Together the results for USER0 and USER2 demonstrate that different similarity measures are appropriate for different users. This indicates that there is a need for a method to detect which similarity measure is appropriate for a particular user and that future research should take this into account. This issue is further complicated by the possibility that the optimal similarity measure may be time variant within the context of a single user. Finally, the possibility exists (and is supported to a certain degree by the data in Table 3) that the optimal similarity measure depends also upon the nature of the anomaly (i.e. the identity of the masquerading user). This presents a difficulty because, as mentioned in previously, the only data available for training is that of the valid user.

Finally, note that there is a significant disparity between optimal dictionary sizes for the users (1000 vs. 200 sequences). This seems to indicate that USER2's behavior is characterized by a smaller set of actions than is USER0's. The possibility remains, however, that none of the similarity measures investigated here are really appropriate for measuring USER0's behavior, and that under a different measure fewer characteristic sequences would be required. Upon examination, however, we note that instance selection (via the LRU algorithm, as described above) was performed with only 1667 tokens of data for USER0's profile while over 5000 tokens were available for dictionary initialization. When a small number of sequences are rated as highly characteristic by the instance selection algorithm, so few of the other dictionary instances will be touched that selection becomes effectively random for sequences other than the most strongly characteristic ones. This turns out to be the case for USER0 as approximately 260 of the instance selection sequences were devoted to selecting only two of the final dictionary sequences. Thus, we hypothesize that for USER0 the LRU instance selection algorithm concentrates undue attention on sequences that are not necessarily reflective of true behavior while selecting the majority of the instances effectively randomly. Therefore a large number of sequences are required in the final profile to obtain reasonable accuracy. By contrast, LRU seems to select important instances much more successfully for USER2 so fewer are needed in the final profile. Any sequences in the profile beyond those most characteristic of behavior represent noise and lead to decreased performance (and, indeed, degraded performance was seen in the case of USER2's profile for larger final dictionary sizes). The behavior demonstrated here highlights an interaction between the similarity measure and the instance selection algorithm, which makes it likely that choice of similarity measure is affected not only by the identity of the profiled user and intruder (and possibly time/concept drift) but also by the choice of instance selection technique.

Conclusions and Future Work

This paper has examined some of the issues involved in similarity-based matching of user behavior sequences. We find that the best measure is user-dependent and there are indications that it may also depend on choice of instance selection technique and even the identity of the anomalous/intrusive user. There do, however, seem to be indications that polynomially bounded measures and measures biased in support of adjacent matches are, overall, preferable. Both the adjacency and the polynomial bound are evidence in support of the hypothesis that user behavioral sequences are characterized by strong correlations among temporally close command tokens.

	MC-P	MCA-P	MC-E	MCA-E
USER0	93.96	67.07	73.44	79.36
USER1	9.11	0.00	0.00	8.23
USER2	0.00	0.00	0.00	0.00
USER3	15.37	0.00	0.00	0.00
USER4	5.49	0.00	0.00	0.00
USER5	10.54	0.00	0.00	0.00
USER6	10.87	0.00	0.00	0.00

Table 2: Detections over all users and similarity measures for USER0's profile, dictionary of 1000 sequences

	MC-P	MCA-P	MC-E	MCA-E
USER0	60.92	33.48	52.69	29.75 †
USER1	68.72	70.14 †	77.06	73.77
USER2	93.85	99.89	99.89	99.89
USER3	61.25	48.30 †	52.47	54.99
USER4	34.80	33.26 †	39.08	35.13
USER5	79.69	75.74 †	87.16	87.38
USER6	81.78	70.58	83.53	68.06 †

Table 3: Detections over all users and similarity measures for USER2's profile, dictionary of 200 sequences

Future directions for this work include examination of other forms of similarity measures. We are also examining the possibility of replacing the simple threshold classification system employed here with either a non-parametric clustering classification system based on Parzen windows (Fukunaga, 1990) or a classification system based on Hidden Markov Models (Rabiner, 1989) of user behavior. Under any of these classification schemes, the problem of the selection of an optimal similarity measure for each user needs to be examined. Finally, the issue of concept drift must be addressed, including the influence of concept drift on the optimal similarity measure.

One issue that must be kept in mind when dealing with concept drift is that of informed malicious users. Such users are presumed to be aware of all security measures in place, including the anomaly detection system and user profiles. Thus, the anomaly detection system must be resistant to deliberate training — that is, it should, ideally, be able to distinguish genuine concept drift on the part of a legitimate user from distortions introduced by a malicious user to subvert the security measures.

References

Anderson, J. P. (1980). *Computer security threat monitoring and surveillance*, (Technical Report), Washington, PA, James P. Anderson Co.

Baffes, P., & Mooney, R. (1996). A novel application of theory refinement to student modeling. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 403-408). Portland, OR: AAAI Press.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering, 13*, 222-232.

Dietterich, T. G. , & Michalski, R. S. (1986). Learning to predict sequences. In Michalski, Carbonell & Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Forrest, S., Hofmeyr, S. A., Somayaji, A., & Longstaff, T. A. (1996a). A sense of self for Unix processes. *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy.*

Forrest, S., Hofmeyr, S., & Somayaji, A. (1996b). Computer immunology. *Communications of the ACM.*

Frank, J. (1994). Machine learning and intrusion detection: Current and future directions. *Proc. of the 17th National Computer Security Conference.*

Fukunaga, K. (1990). *Statistical Pattern Recognition (second edition).* San Diego, CA: Academic Press.

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. (1990). A network security monitor. *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy* (pp. 296-304).

Hirsh, H., & Japkowicz, N. (1994). Bootstrapping training-data representations for inductive learning: A case study in molecular biology. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 639-644). Seattle, WA.

Krsul, I., & Spafford, E. H. (1996). *Authorship analysis: Identifying the author of a program*, (CSD-TR-

96-052), West Lafayette, IN: Purdue University, Department of Computer Sciences.

Kumar, S. (1995). *Classification and detection of computer intrusions*. Doctoral dissertation, Department of Computer Sciences, Purdue University, W. Lafayette, IN.

Lane, T., & Brodley, C. E. (1997). *Detecting the abnormal: Machine learning in computer security,* (TR-ECE 97-1), West Lafayette, IN: Purdue University.

Lunt, T. F. (1990). IDES: An intelligent system for detecting intruders. *Proceedings of the Symposium: Computer Security, Threat and Countermeasures.* Rome, Italy.

Norton, S. W. (1994). Learning to recognize promoter sequences in E. coli by modelling uncertainty in the training data. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 657-663). Seattle, WA.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE.*

Salzberg, S. (1995). Locating protein coding regions in human DNA using a decision tree algorithm. *Journal of Computational Biology, 2*, 473-485.

Smaha, S. E. (1988). Haystack: An intrusion detection system. *Proceedings of the Fourth Aerospace Computer Security Applications Conference* (pp. 37-44).

Spafford, E. H., & Weeber, S. A. (1992). Software forensics: Can we track code to its authors? 15^{th} *National Computer Security Conference* (pp. 641-650).

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements, . *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT).* Avignon, France.

Acknowledgments: We would like to thank Tim Stough, Gene Spafford, Ronny Kohavi, Paul Utgoff, Craig Codrington, and our anonymous reviewers for their helpful comments. We are also grateful to the members of the Purdue MILLENNIUM Lab and our other data donors for their contributions of data and insight to this work. A portion of this research was funded by the commercial and government sponsors and supporters of the COAST Laboratory: Cisco Systems, HP, Schlumberger, MITRE, Sprint, Sun Microsystems, Hughes Research Laboratories, Thompson Consumer Electronics, and the U.S. Department of Defense.