Information-Theoretic Measures for Anomaly Detection

Wenke Lee Dong Xiang
Computer Science Department
North Carolina State University
Raleigh, NC 27695-7534
wenke@csc.ncsu.edu, dxiang@unity.ncsu.edu

Abstract

Anomaly detection is an essential component of the protection mechanisms against novel attacks. In this paper, we propose to use several information-theoretic measures, namely, entropy, conditional entropy, relative conditional entropy, information gain, and information cost, for anomaly detection. These measures can be used to describe the characteristics of an audit data set, suggest the appropriate anomaly detection model(s) to be built, and explain the performance of the model(s). We use case studies on Unix system call data, BSM data, and network *tcpdump* data to illustrate the utilities of these measures.

1 Introduction

Intrusion detection systems (IDSs) is an important component of the defense-in-depth or layered network security mechanisms. An IDS collects system and network activity data, e.g., BSM [27] and tcpdump[9] data, and analyzes the information to determine whether there is an attack occurring. The two main techniques for intrusion detection (ID) are *misuse detection* and *anomaly detection*. Misuse detection (sub)systems, for example, IDIOT [10] and STAT [8], use the “signatures” of known attacks, i.e., the patterns of attack behavior or effects, to identify a matched activity as an attack instance. Misuse detection are not effective against *new* attacks, i.e., those that don’t have known signatures. Anomaly detection (sub)systems, for example, the anomaly detector of IDES [20], use established normal profiles, i.e., the expected behavior, to identify any unacceptable deviation as possibly the result of an attack. Anomaly detection can be effective against new attacks. However, new legitimate behavior can also be falsely identified as an attack, resulting a false alarm. In practice, reports of attacks are often sent to security staff for investigation and appropriate actions.

In 1998, DARPA conducted an evaluation to assess the state-of-the-art of ID research. The results showed that the best research systems had detection rates (i.e., the percentages of attack incidents correctly identified) below 70% [18]. Most of the missed intrusions were new attacks that can lead to unauthorized user or root access to the mocked military network used in the evaluation. The results of the 1999 DARPA evaluation are even more troubling. With improved capabilities, e.g., the added modules for detecting the attacks missed in the previous evaluation, the research IDSs still had detection rates below 70% because many new attacks (that is, new in the 1999 evaluation) were missed. These evaluations showed that even the cutting-edge ID technology is not very effective against new attacks, and the improvement is often too slow and too little to keep up with the “innovation” by sophisticated attackers.

The research systems in the DARPA evaluations, like most of the leading commercial products, employ mainly misuse detection techniques. The main reason against deploying anomaly detection (sub)systems is that they tend to generate many false alarms and hence compromise the effectiveness of intrusion detection. Given that our adversaries will always develop and launch new types of attacks in an attempt to defeat our deployed intrusion prevention and detection systems, and that anomaly detection is the key to the defense against novel attacks, we must develop significantly better anomaly detection techniques.

In most computing environments, the behavior of a subject (e.g., a user, a program, or a network element, etc.) is observed via the available audit data logs. The basic premise for anomaly detection is that there is intrinsic characteristic or regularity in audit data that is consistent with the normal behavior and thus distinct from the abnormal behavior. The process of building an anomaly detection model should therefore involve first studying the characteristic of the data and then selecting a model that best utilizes the characteristic. However, due to the lack of theoretical understandings and useful tools for characterizing audit data, most anomaly detection models are built based solely on "expert" knowledge or intuition [19], which is often imprecise and incomplete given the complexities of today's network environments. As a result, the effectiveness of the models is limited. More seriously, a lot of research in anomaly detection (and intrusion detection in general) has been focusing on a specific (and ad hoc) method for a specific environment. The research results often do not contribute to the fundamental understanding of the field nor lend themselves to the broader problem domain.

Our research aims to provide theoretical foundations as well as useful tools that can facilitate the IDS development process and improve the effectiveness of ID technologies. In this paper, we propose to use several information-theoretic measures, namely, entropy, conditional entropy, relative conditional entropy, information gain, and information cost, for anomaly detection. These measures can be used to describe the characteristics of an audit data set, suggest the appropriate anomaly detection model(s) to be built, and explain the performance of the model(s). We use case studies on *sendmail* system call data, *sendmail* BSM data, and network *tcpdump* data to illustrate the utilities of these measures.

The rest of the paper is organized as follows. Section 2 describes the information-theoretic measures. Section 3 presents several case studies in using these measures to build anomaly detection models. Section 4 discusses the limitations and possible extensions of our current approach. Section 5 compares our research with related efforts. Section 6 outlines our future work.

2 Information-Theoretic Measures

In this section, we discuss several information-theoretic measures (these concepts are covered in many texts on information theory, e.g. [4]). We explain how these measures characterize the regularity embedded in audit data and influence the performance of anomaly detection models. We also outline the procedure of using these measures to build anomaly detection models.

2.1 Entropy

Entropy, or Shannon-Wiener Index [25], is an important concept in information theory and communication theory. It measures the uncertainty (or impurity) of a collection of data items.

Definition 1 *For a dataset X where each data item belongs to a class $x \in C_X$, the entropy of X relative to this $|C_X|$-wise classification is defined as*

$$H(X) = \sum_{x \in C_X} P(x) \log \frac{1}{P(x)}$$

where $P(x)$ is the probability of x in X.

The typical interpretation of entropy is that it specifies the number of bits required to encode (and transmit) the classification of a data item. The entropy value is smaller when the class distribution is skewer, i.e., when the data is more “pure”. For example, if all data items belong to one class, then entropy is 0, and 0 bit needs to be transmitted because the receiver knows that there is only one outcome. The entropy value is larger when the class distribution is more even, i.e., when the data is more “impure”. For example, if the data items are evenly distributed in $|C_X|$ classes, then $log|C_X|$ bits are required to encode a classification.

For anomaly detection, we can use entropy as a measure of the regularity of audit data. Each *unique* record in an audit dataset represents a class. The smaller the entropy, the fewer the number of different records (i.e., the higher the redundancies), and we say that the more regular the audit dataset. High-regularity data contains redundancies that help predicting future events because the fact that many events are repeated (or redundant) in the current dataset suggests that they will appear in the future. Therefore, anomaly detection model constructed using dataset with smaller entropy will likely be simpler and have better detection performance. For example, if the audit data contains a single event class, e.g., a user command dataset where all commands are *mail*, then the entropy is 0 and a single rule can identify any other event, e.g., *ftp*, as an anomaly. If the audit data contains many event types, then the entropy is greater than 0 and a more complex model is needed.

2.2 Conditional Entropy

Definition 2 *The conditional entropy of X given Y is the entropy of the probability distribution $P(x|y)$, that is,*

$$H(X|Y) = \sum_{x,y \in C_X, C_Y} P(x,y) \log \frac{1}{P(x|y)}$$

$P(x,y)$ is the joint probability of x and y and $P(x|y)$ is the conditional probability of x given y.

Because of the temporal nature of user, program, and network activities, we need to measure the temporal or sequential characteristic of audit data. Using the definition above, let X be a collection of sequences where each is denoted as $(e_1, e_2, \ldots, e_{n-1}, e_n)$, and each e_i is an audit event; and let Y be the collection of subsequences where each is $(e_1, e_2, \ldots, e_k)$, and $k < n$, then the conditional entropy $H(X|Y)$ tells us how much uncertainty remains for the rest of audit events in a sequence x after we have seen y, i.e., the first k events of x (note that since y is always a subsequence of x here, we have $P(x,y) = P(x)$). For anomaly detection, we can use conditional entropy as a measure of regularity of sequential dependencies. And as the case of entropy, the smaller the conditional entropy, the better. For example, if each audit trail is a sequence of events of the same type, e.g., $X = \{aaaaa, bbbbb, \ldots\}$, then the conditional entropy is 0 and the event sequences are very deterministic. Conversely, a large conditional entropy indicates that the sequences are not as deterministic and hence much harder to model.

2.3 Relative Conditional Entropy

Definition 3 *The relative entropy between two probability distributions $p(x)$ and $q(x)$ that are defined over the same $x \in C_X$ is*

$$relEntropy(p|q) = \sum_{x \in C_X} p(x) \log \frac{p(x)}{q(x)}$$

For anomaly detection, we often build a model using a *training* dataset and apply the model to the *test* dataset. These two datasets must have the same (or very similar) regularity for the anomaly detection model to attain high performance. Relative entropy measures the *distance* of the regularities between two datasets. It is obvious that the smaller the relative entropy, the better. For example, if $p = q$, then the relative entropy is 0, indicating that the two datasets have the same regularity.

When we use conditional entropy to measure the regularity of sequential dependencies, we can use relative conditional entropy to measure the distance between two audit datasets.

Definition 4 *The relative conditional entropy between two probability distributions* $p(x|y)$ *and* $q(x|y)$ *that are defined over the same* $x \in C_X$ *and* $y \in C_Y$ *is*

$$relCondEntropy(p|q) = \sum_{x,y \in C_X, C_Y} p(x,y) \log \frac{p(x|y)}{q(x|y)}$$

Again, for anomaly detection, the smaller the relative conditional entropy, the better.

2.4 Information Gain and Classification

Intrusion detection can be cast as a classification problem: we wish to classify an audit event as belonging to the normal class, the abnormal class (in the case of anomaly detection) or a particular class of intrusion (in the case of misuse detection). Here assuming that classifiers are used as anomaly detection models, we discuss how the regularity of audit data influences the performance of anomaly detection models.

Given a training dataset where the records are defined by a set of features and each record belongs to a class, the goal of constructing a classifier is that after (selectively) applying a sequence of feature value tests, the dataset can be partitioned into "pure" subsets, i.e., each in a target class, so that the sequence of feature value tests can be used as the conditions in the classifier to determine the class of a new record (when its class is not yet known). In this process, *all* records in each final subset are considered as belonging to the *majority* class of the subset because for each record, there can be only one classification outcome. It is obvious that the purer the final subsets, the more accurate the classifier. Therefore when constructing a classifier, a classification algorithm needs to search for features with high *information gain* [22], which is the reduction of entropy when the dataset is partitioned according to the feature values.

Definition 5 *The information gain of attribute (i.e., feature) A on dataset X is*

$$Gain(X, A) = H(X) - \sum_{v \in Values(A)} \frac{|X_v|}{|X|} H(X_v)$$

where $Values(A)$ *is the set of possible values of* A*, and* X_v *is the subset of* X *where* A *has value* v*.*

If all the features have low information gain, then the classifier will have poor performance because after the original dataset is partitioned, the subsets still have large entropy, i.e., they are still "impure". Therefore, for anomaly detection (and intrusion detection in general), the higher the information gain of the features, the better.

When the regularity of sequential dependencies is used directly in the anomaly detection model, there is a direct connection between conditional entropy and information gain. For example, suppose we have a classifier that uses the first n-1 audit events to classify (i.e., predict) what normally the nth event should be. In this case, the first n-1 events are used as the features and the nth event as the class. Since all n-1 features can be used in the classifier, for simplicity in our discussion, we can "collapse" all of them into a single feature A. Then for $Gain(X, A)$, the second term in the formula is essentially the conditional entropy of the length n sequence given the length n-1 subsequence (prefix). Therefore, when we model a sequence, the smaller the conditional entropy, the higher the information gain, and hence the better detection performance of the model.

When we model a complex subject, e.g., network traffics, we often need not only information pertaining to the current event but also sequential (or temporal) information on the previous events. Conditional entropy can be used in the *feature construction* process to suggest what features can be added so that the feature set contains the information on *both* the current and the previous events. For example, suppose that in a timestamped audit dataset, each record inintially is defined as $< t, f_1, f_2, \ldots, f_n, c >$, where t is the timestamp, each f_i is a feature, e.g., the duration of the current connection, number of bytes sent, etc., and c is the class label. Suppose that we use the service of a connection as its class, that is, we want to model how each service normally behaves. If there is strong regularity, i.e., low conditional entropy, on the sequence of services (or the combination of service and other features), we can add features that express this regularity. One way is to add features that act as place holders for the services of previous connections (that fall within a time window), i.e., each connection record includes the names of some previous services, s_{i-1}, s_{i-2}, etc. Alternatively, to reduce the total number of features (and hence the complexities of the model), we can use some statistical feature(s), e.g., within the past n seconds, the percentage of the services that are the same as the current one, and the percentage of those that are different, etc., to approximate the regularity information. In [13], we showed that these temporal and statistical features usually have high information gain, and hence a better model can be built when these features are added to the audit data.

2.5 Information Cost

Intuitively, the more information we have, the better the detection performance. However, there is always a cost for any gain. For intrusion detection, we define information cost as the average time for processing an audit record and checking against the detection model. When we include more information, we not only increase the data processing time, we often increase the model complexities as well. There needs to be a trade-off between detection performance and cost. For example, the measure Accuracy/Cost may be used to determine the "optimal" amount of information to be used in the model.

2.6 Application in Anomaly Detection

The information-theoretic measures we define here can be used for anomaly detection in the following general approach:

- Measure the regularity of the audit data and perform the appropriate data transformation. Iterate this step if necessary so that the dataset used for modeling has high regularity.
- Determine how the model should be built, i.e., how to achieve the best performance or the optimal performance/cost trade-off, according to the regularity measure.
- Use the relative entropy measure to determine whether a model is suitable for a new dataset.

In Section 3, we present several case studies to illustrate this approach in details.

3 Case Studies

In this section, we describe our experiments on the University of New Mexico (UNM) *sendmail* system call data, MIT Lincoln Lab (DARPA Evaluation) *sendmail* BSM data, and MIT Lincoln Lab *tcpdump* data, to show how to use the information-theoretic measures defined earlier to build anomaly detection models. These case studies are presented in the order of simpler to more complex in terms of the audit data used. With the UNM system call data, we demonstrate how to use conditional entropy to determine the appropriate length used for sequencing the system calls to construct anomaly detection models. With the Lincoln Lab BSM data, we show how to use conditional entropy to determine whether including additional information, e.g., *obname*, will likely to improve detection performance. With the Lincoln Lab *tcpdump* data, we show how to use entropy to partition the network data into more regular subsets, and how to use conditional entropy to determine the time window size by which temporal and statistical features can be computed and included in the anomaly detection models.

3.1 UNM *sendmail* System Call Data

In a ground-breaking study, Forrest et al. [6] discovered that the short sequences of (consecutive) system calls made by a program during its normal executions are very consistent. More importantly, the sequences are different from the sequences of its abnormal (exploited) executions as well as the executions of other programs. Therefore a very concise database containing these normal sequences can be used as the "self" definition of the normal behavior of a program, and as the basis to detect anomalies. A number of follow-on studies, for example, [5, 7, 15, 28], attempted alternative models, e.g., variable-length patterns, classification rules, Neural Nets, Hidden Markov Model, etc., instead of the original simplistic model of database look-up of fixed-length sequences. These alternative and more sophisticated models do not have significant performance improvement over the original model. It is thus believed that the *sendmail* system call data is highly regular and hence a simple model would suffice. However, we have seen no attempt to study how to measure the regularity and exploit it in the model building process. Most noticeably, the original study by Forrest et al. did not suggest a means to determine the appropriate sequence length, rather, an ad hoc trial-and-error approach was used. The follow-on studies simply used the sequence length given by Forrest et al.

In this case study, we did not attempt to suggest yet another model. Rather, we studied how to measure the data regularity, and use it to determine the sequence length, i.e., how should the model be built, and explain the performance of the anomaly detection model, i.e., why it works. We obtained a set of *sendmail* system call traces from UNM. The details of their data gathering process, experiments on the data, and results are in [6]. Each trace contains the (entire sequence(s) of) consecutive system calls made by the run-time process(es). Using a sliding window of size n, we can process a system call trace into a set of length-n sequences. This set is used as our dataset, i.e., each sequence is a data point. We can then compute the conditional entropy, a measure of regulatory, of the dataset. Let X represent the set of length-n sequences, and Y be the set of (prefix) subsequences of the length n-1, the conditional entropy $H(X|Y)$ then measures the regularity of how the first n-1 system calls determines the nth system call. In more details, for each unique $x \in X$, $|x|$ is the number of occurrences of x in X, and $y(x)$ be the length=n-1 subsequence of x, i.e., if $x = (s_1 s_2 \ldots s_{n-1} s_n)$, then $y(x) = (s_1 s_2 \ldots s_{n-2} s_{n-1})$, then $H(X|Y) = \sum_{x \in X} \frac{|x|}{|X|} \log \frac{|x|}{|y(x)|}$.

Figure 1 shows the conditional entropy for each normal trace when window size varies from 3 to 19 with an increment of 1. Each trace here (e.g., "plus", "queue", etc.) models a different kind (or configuration) of normal *sendmail* runs [6], hence we model the traces separately. We can also put all traces together to form the "total" dataset and compute a model. "mean" is simply the average of results from all the individual traces. We can see that the conditional entropy drops as the sequence length increases, intuitively, because the more information is included, the more deterministic (i.e., regular) the dataset becomes. We can also see that the conditional entropy drops to very small values after sequence length reaches 6 or 7 (Forrest et al. used length 6 in their original study).

The small conditional entropy values suggest that for the *sendmail* system call data, the nth

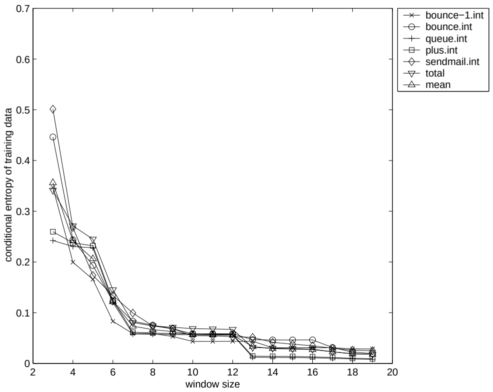

Figure 1: Conditional Entropy of Training Data

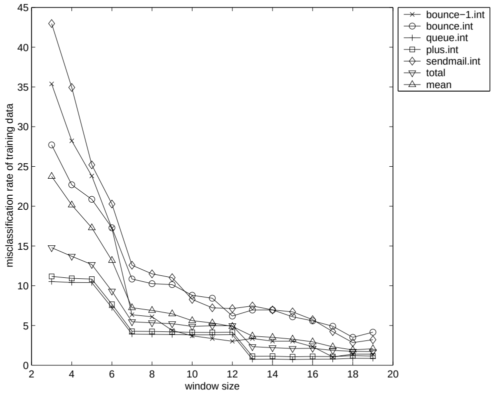

Figure 2: Misclassification Rate of Training Data

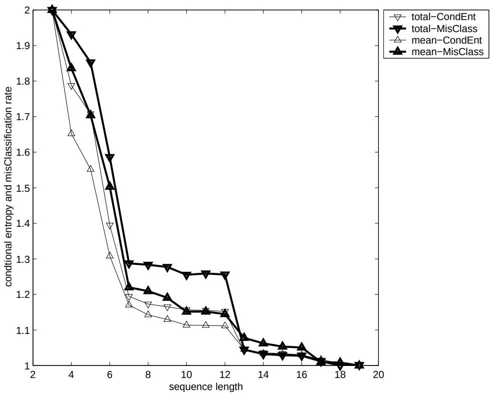

Figure 3: Conditional Entropy vs. Misclassification Rate

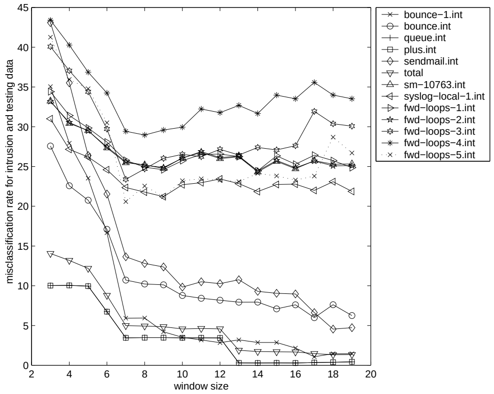

Figure 4: Misclassification Rate of Testing Data

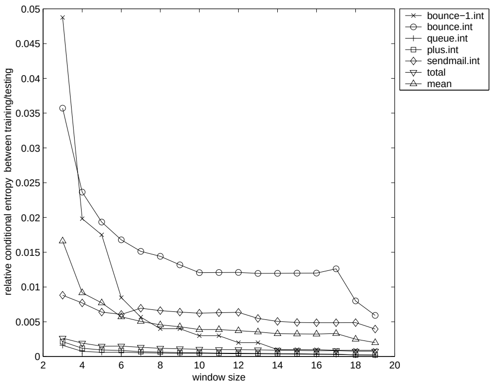

Figure 5: Relative Conditional Entropy btw. Training and Testing Normal Data

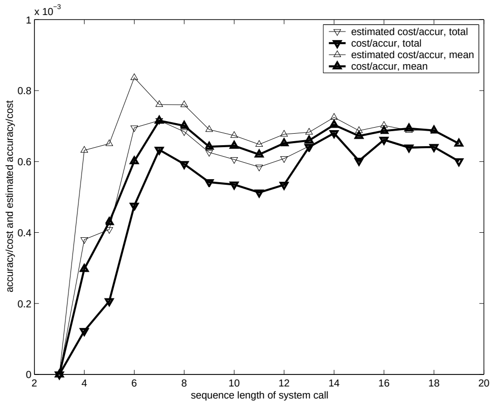

Figure 6: (Real and Estimated) Accuracy/Cost Trade-off

system call is highly deterministic given the first n-1 system calls. According to the discussion in Section 2.4, we can build a classifier where the first n-1 system calls are the features and the nth system call is the class. We can expect this anomaly detection model to have good detection performance. For each normal trace, we used the first 80% as the training data and the last 20% as the testing data. We applied RIPPER [3], a (typical) classification rule induction program, to the training data to compute a classifier, and then tested it on the testing data and intrusion traces. To verify the direct connection between conditional entropy and detection performance (see Section 2.4), we built the classifiers using n from 3 to 19.

Figure 2 shows the misclassification rate on training data. Figure 3 shows the comparison of misclassification rate on the training data and conditional entropy when the values are all scaled into 1 to 2 range. A misclassification is the situation where the classifier predicts an item to be in class i while the actual class is j. The misclassification rate, computed as the percentage of misclassification in the whole dataset, thus measures the detection performance. We see in Figure 3 that the trend of the misclassification rate coincides with the trend of the conditional entropy. This is what we we expected according to the discussion in Section 2.4. Because of this phenomenon, we can use the conditional entropy plot, which can be considered as the *estimated* trend of misclassification rate, to select a sequence length for the detection model. For example, if detection performance is all we care about, then we know that length 6 is better than 4, and 14 is better than 6.

Figure 4 shows the misclassification rate on testing data. We can see that the misclassification rates for the intrusion traces are much higher, in fact, beyond sequence length 6, they are in different ranges. This suggests that we can use the range of the misclassification rate as the indicator of whether a given trace is normal or abnormal (intrusion). That is, in practice, the IDS reports an anomaly only when the misclassification rate (for the whole trace) is high, not when a system call is misclassified. Figure 5 shows the relative conditional entropy between training and testing normal data. We can see that when the relative entropy is larger, i.e., when the training and testing normal datasets differs more (see discussion in Section 2.3), then the misclassification rate on testing normal data is also gets higher. This phenomenon suggests that we should use relative conditional entropy (or simply relative entropy) between the training and the testing sets to either understand why the detection performance is satisfactory or discover that the testing set has different regularity and is not suitable for the model.

From Figures 2 and 4, we can see that the longer the sequence length, the better the detection performance. However, as discussed in Section 2.5, we need to consider the information cost. We define information cost as the average time required for processing an audit record and checking against the detection model. The results from our time measurement experiments verified the paper-and-pencil analysis that the cost is a linear function of the sequence length. That is, we can normally estimate the cost, without building and running a model, if we know the data and the algorithm used in the model. Suppose we wish to select the sequence length to build a model that has the optimal accuracy per cost unit. We can study the ratio between the *estimated* accuracy, i.e., one minus the conditional entropy, and the cost for a given sequence length. Figure 6 shows the ratios between real and estimated accuracy and cost. The plots on estimated accuracy/cost versus sequence length match the trend of the real accuracy/cost, and can thus be used to select the best sequence length if we want to optimize accuracy per unit cost.

3.2 MIT Lincoln Lab *sendmail* BSM Data

The UNM *sendmail* data only contains the system call names. An interesting question in building anomaly detection for *sendmail* (or other programs) is: can there be detection performance gain by including additional information, i.e., arguments, object names, etc? Here we studied whether we can use the regularity of data to find the answer instead of the expensive trial-and-error process of building and testing many models.

We used the BSM data developed and distributed by MIT Lincoln Lab for the 1999 DARPA evaluation in our experiments. We processed a week's BSM data and extracted audit records of all *sendmail* sessions. Each audit record corresponds to a system call made by *sendmail*. In addition to the system call name, each audit record contains additional information such as the (real and

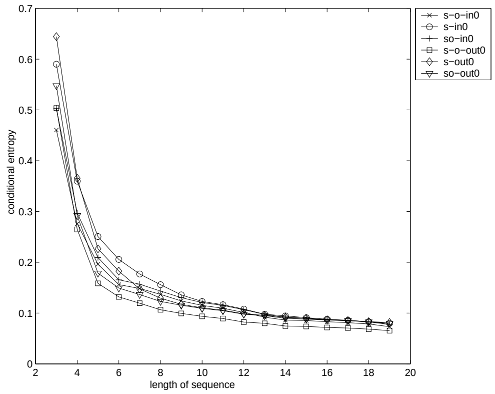

Figure 7: Conditional Entropy of In- and Out-bound Email

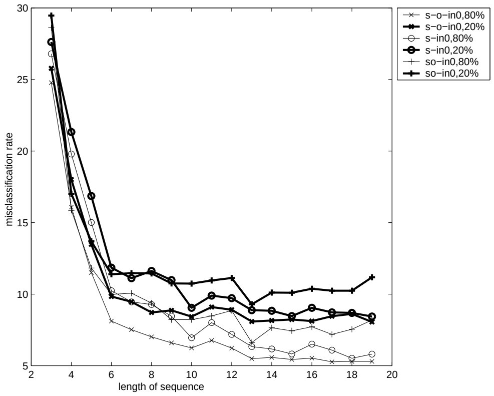

Figure 8: Misclassification Rate of In-bound Email

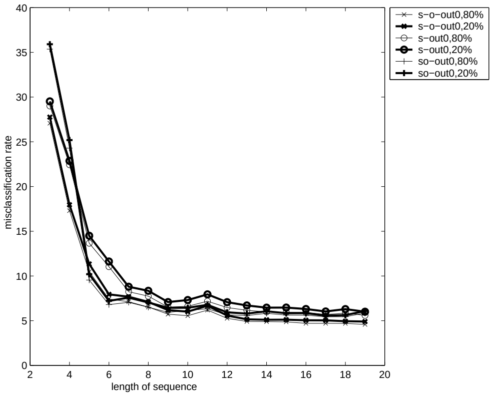

Figure 9: Misclassification Rate of Out-bound Email

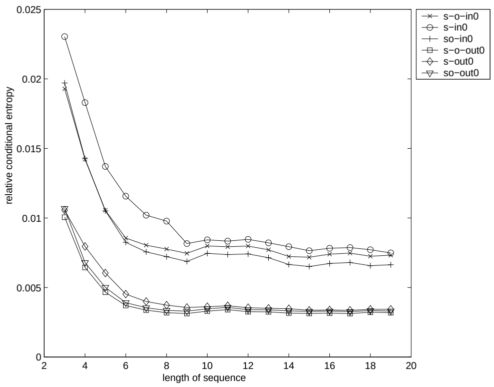

Figure 10: Relative Conditional Entropy

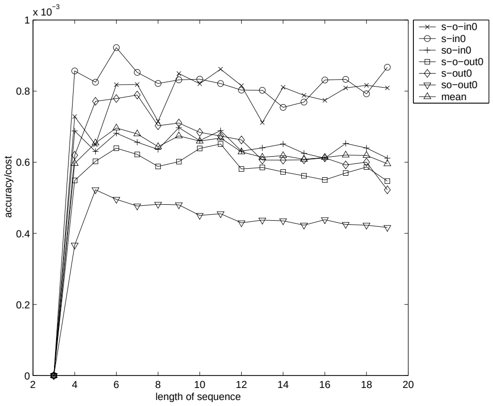

Figure 11: Accuracy/cost Trade-off

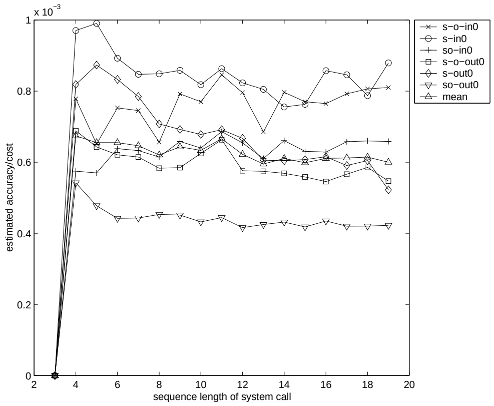

Figure 12: Estimated Accuracy/cost Trade-off

effective) user and group IDs, the obname (i.e., the name of the object accessed by the system call), and arguments, etc. That is, a *sendmail* BSM trace from a session is $(< s_1, o_1, a_1, \ldots >, < s_2, o_2, a_2, \ldots >, \ldots, < s_l, o_l, a_l, \ldots >)$, instead of $(s_1, s_2, \ldots, s_l)$. Here, s_i, o_i, and a_i represent a system call name, obname, and argument, respectively.

From the experiments on UNM data, we know that for *sendmail*, conditional entropy directly influences the detection performance. Thus, to find out whether including additional information will help improve the detection performance, we just need to test whether it results in smaller conditional entropy. We tested two alternative methods of including obname: in the first, denoted as **so**, the trace now becomes $(s_1\_o_1, s_2\_o_2, \ldots, s_l\_o_l)$, that is, the obname is simply appended to the system call name; in the second, denoted as **s-o**, the trace now becomes $(s_1, o_1, s_2, o_2, \ldots, s_l, o_l)$, that is, the obname is treated as equally important as the system call name in the sequence. We also changed the value of an obname to either "system" (indicating that the object is in a system directory), "user" (indicating that the object is in a user's directory) , or "other". This transformation is necessary because if we use the full obname, which is often a temporary file in a system or user directory, the data will be very irregular.

In our experiments, we used the first 80% of all the *sendmail* traces for computing conditional entropy and training classifiers, and the remaining 20% for testing. Since there are two directions for the *sendmail* runs, i.e., in-bound and out-bound, we used the data from the two directions in separate experiments. There is no exploit against *sendmail* in the data, thus, we only compare the detection models on the normal testing data. In Figures 7, 8, 9, 10, 11, the legends "s-in/out" denote system call only data, "so-in/out" refer to datasets with system call combined with obname in **so** mode, and "s-o-in/out" denote data sets with system call followed by obname in **s-o** mode. A "0,80%" appendix refers to training datasets and a "0,20%" refers to testing datasets.

From Figure 7, we can see that conditional entropy decreases as the sequence length increases, as the case of with experiments on UNM data. In addition, datasets with system call only have slightly larger conditional entropy than those with the added obname, and that **s-o** datasets have slightly smaller conditional entropy than **so** datasets. Figures 8 and 9 show that the detection models computed using datasets with added obname have slightly lower misclassification rate (hence better detection performance) and that the detection models from **s-o** datasets slightly outperform the models from **so** datasets. This again confirms that there is direct connection between conditional entropy and detection performance. Comparing Figures 8 and 9, we can see that for in-bound mails the testing data have clearly higher misclassification rates than the training data, whereas out-bound mails do not have such phenomenon. Figure 10 shows the relative conditional entropy between training and testing datasets. We can see that the out-bound mails have much smaller relative conditional entropy than the in-bound mails. This again confirms that relative conditional entropy is indicative of detection performance on test data sets.

We computed information cost the same as in the experiments on the UNM data. The estimated accuracy/cost plots in Figure 12 match the trend of the real accuracy/cost plots in Figure 11, and can thus be used to select not only the best sequence length for a particular model but also the best model. The plots suggest that although including the additional obname has shown to improve the detection performance, when the trade-off of accuracy/cost is considered, it is actually better to use system call only.

3.3 MIT Lincoln Lab Network Data

A major challenge in anomaly detection is to determine the granularity of the subject. For example, in modeling user behavior, we need to decide whether to build separate profiles for weekdays and weekends, and for the weekdays, whether finer time segments, e.g., mornings, afternoons, and evenings, are necessary [16]. Likewise, for a network, whether we should build models for each host, service, or some combinations of the two. Without proper guidelines and tools, this remains an ad hoc process. Here we studied whether we can measure the regularity of network data and use it to guide data partitioning, which is equivalent to subject refinement, and to help feature construction (hence model building).

We used the *tcpdump* data developed and distributed by MIT Lincoln Lab for the 1998 DARPA evaluation in our experiments. The data contains traffics in a simulated military network that

Date	Entropy	Entropy per-Host	Entropy (further) per-Service
Monday	5.78633	3.12795	2.60068
Tuesday	6.02534	3.1319	2.69049
Wednesday	6.83504	3.63338	3.31312
Thursday	7.38497	2.97228	2.69542

Table 1: Entropy of Network Connection Data

consists of hundreds of hosts. We processed four days' *tcpdump* data using a modified version of Bro [23], a programmable IDS with a robust packet filtering and re-assembly engine. Each record describes a connection using the following features: timestamp, duration, source port, source host, service (destination port), destination host, source bytes (number of bytes from source to destination), destination bytes, and flag (summarizing the hand-shake behavior). We used the connection data for each day as a separate set for experiments. We also separated out the intrusions to create the pure normal datasets.

We computed the entropy, i.e., the irregularity, for each (normal) dataset. Here, each data point is simply a connection record with the timestamp removed. In order to achieve high detection performance with low false alarm rate, the dataset needs to be as regular as possible, i.e., its entropy as small as possible (see discussion in Section 2.1). If the entropy is large, then we should try to further partition the data set into more regular subsets. Table 1 shows the entropy of the original (unpartitioned) datasets and the subsets. Here the entropy after partitioning is the average of the entropy values of all the subsets. We can see the entropy values of the original datasets are very large. This implies that if we build a model using dataset that contains all hosts and all services, the data may be too irregular for the model to work well. We tried all features to select the one that results in subsets with the smallest entropy values. Destination host was then used for partitioning the data in to per-host subsets. We see that the entropy is significantly decreased, which means that each subset is much more regular. If we further partition the data into per-service subsets, the entropy continues to decrease but not as dramatically. Note that this data partitioning process is equivalent to the classification process (see Section 2.4) since both use reduction in entropy, i.e., information gain, as the guiding principle.

In previous work, we showed that introducing some per-host and per-service temporal and statistical features, e.g., "the count of connections to the same host as the current one in the past 2 seconds", to the connection data can significantly improve the detection performance of network models [16]. However, we did not develop a means to determine the proper time window, e.g., 2 seconds, for computing the features. In the case study here, we next explored if we can use conditional entropy to determine the time window. We created sequences of service, destination host, flag, and the combination of the three, from the connection data as follows: using a sliding time window of n seconds and a step of one connection, scan the connection records that fall within the past n seconds with regard to the current connection and put all the services (or destination hosts, flags, etc.) into a short sequence. Given a set of such sequences, we then compute the conditional entropy of sequence x of length k given its subsequence (prefix) y of length k-1, i.e., the uncertainty of determine the next service (or host, flag, etc.) given the past services. Since the sequences can have different lengths due to the fact that the traffic volume per time window is not constant, we first computed the entropy for each subset of k-length sequences, then the weighted sum of these entropy values is used as the entropy of entire set. We used different time windows, with an increment of 2 seconds, e.g., 2, 4, 6, 8, 10, etc. to create the sequences and compute the conditional entropy. From Figures 13, 14, 15, and 16, we can see that, in general, conditional entropy decrease as window size grows. Intuitively, this is because the more information is included, the smaller the uncertainty. We can also see that the conditional entropy on flag sequences is very low, indicating that, in the normal dataset, connections within a time window are likely to have similar behavior with regard to network protocols, i.e., they all have the normal flags or error flags (e.g., connection failures due to network congestion).

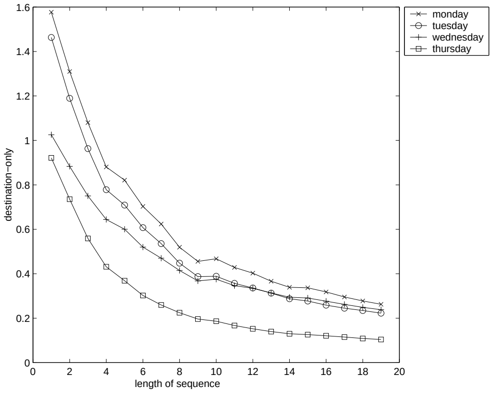

Figure 13: Conditional Entropy: Destination Only

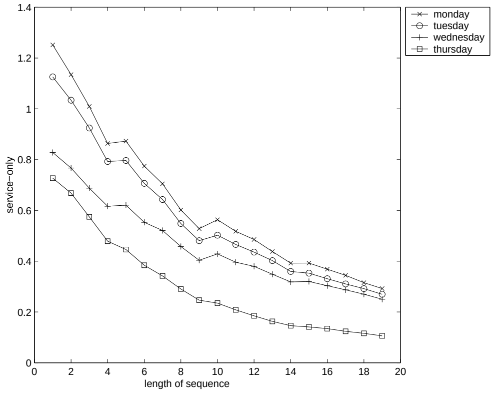

Figure 14: Conditional Entropy: Service Only

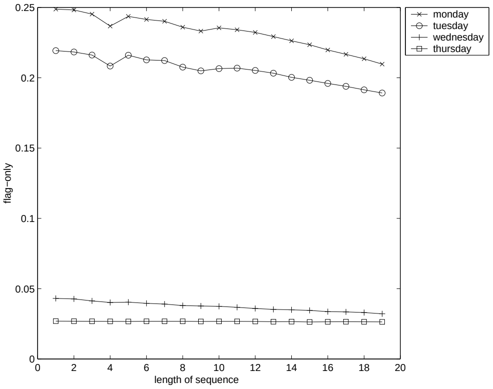

Figure 15: Conditional Entropy: Flag Only

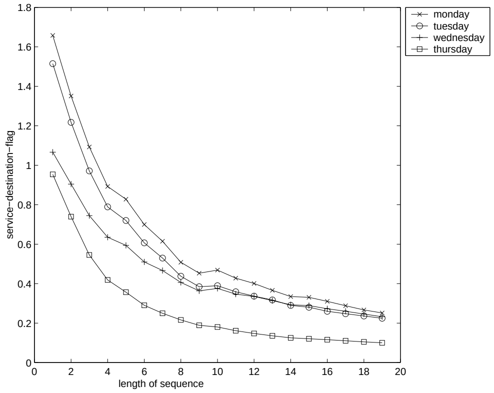

Figure 16: Conditional Entropy: Service, Destination and Flag

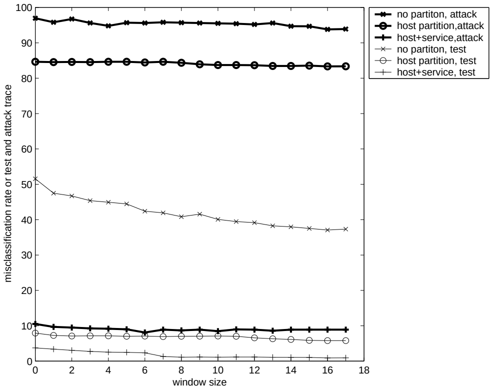

Figure 17: Misclassification Rate: Normal Data and Intrusion Data

As in previous work [16], for each time window, we constructed a set of temporal and statistical features to (approximately) capture the per-host and per-service sequential dependencies, e.g., “for the connections in the past 2 seconds, the percentage that have the same destination host as the current one", "the percentage of different hosts", "the percentage of normal flags", "the percentage of error flags", etc. We added these features to the connection records and applied RIPPER to build classifiers as anomaly detection models. Our goals were to study how the data partitioning scheme based on entropy and the feature construction process based on conditional entropy affect the performance of detection models. We used 80% of the normal data for training and the remaining 20% of the normal data as well as the intrusion data for testing. Two factors determine how a dataset used in the experiments was derived from the original dataset: partitioning - none, by host, or (by host first and further) by service; and temporal and statistical features - none, or using a particular time window to compute and add the features. For the datasets without partitioning, we used the destination host of a connection as its class label, for the per-host datasets, we used service, and for the per-service datasets, we used flag.

Figure 17 shows the misclassification rates of the anomaly detection models constructed from these datasets. A window size 0 means that no temporal and statistical features are added. The misclassification rates of all per-host models, and likewise all the per-service models, of the same time window are averaged. To simply our presentation, we plotted only the 4-day averages here. We can see from the figure that intrusion datasets have much higher misclassification rates, in ranges that clearly separated from the corresponding normal datasets. For the normal data, models from the (more) partitioned datasets have much better performance, and models from the datasets with added temporal and statistical features also have better performance.

Note that compared with the results from experiments on *sendmail* data, here the relationship between conditional entropy and misclassification rate is not as clear because the features we added can only approximate the sequential dependencies. We are experimenting the "place holder" method (see Section 2.4) to construct features that directly represent sequential dependencies [1].

4 Discussion

In this section, we discuss the advantages as well as limitations of our work.

As illustrated in the case studies, we can use the information-theoretic measures to characterize regularity in audit data and use it to guide the model building and evaluation process. In our experiments, we exhaustively computed the models, for example, using different sequence lengths, only for the purpose of showing the relationship between regularity and detection performance. Once we understand this relationship, in practice, we can simply compute the regularity of a given dataset and determine how to build *a* model. Computing regularity, in general, is much more efficient than computing a model. Therefore our approach is much superior than the current ad-hoc and expensive trial-and-error practice where there is no guideline for building a model and explaining its performance.

False alarm rate is a very important performance measure for intrusion detection in general and anomaly detection in particular. We believe that because of the probabilistic nature of anomaly detection, alarms should be post-processed so that sporadic false alarms due to the the inherent uncertainty in data (that is, we cannot assume 100% regular data) can be filtered out. For example, for the *sendmail* data, we use the misclassification rate on the whole trace, instead of the individual misclassification, for detecting anomalies. Likewise, for the network connection data, we can use the misclassification rate on a (time) segment for anomaly detection. We believe that anomalies within a single connection, e.g., a "buffer overflow" attack on a program on the destination within a *telnet* connection, can be best detected using models on lower level data, e.g., system call data

[1] Note to the reviewers: the results of this case study are still preliminary. We are continuing our experiments and will report new results in the final version of this paper.

for the target program. Regardless how alarms are post-processed, the model needs to have high accuracy (e.g., for normal data, low misclassification rate). Therefore, we can say that regularity in audit data (indirectly) influences false alarm rate.

We have not attempted to explain or reason why certain regularity exists in a particular dataset. The motivation is to make our approach independent of the assumptions of the underlying computing environments because after all, we aim to develop general theories and tools for anomaly detection. In practice, our approach can always be complimented by using expert domain knowledge to validate the computed regularity.

We have shown that there is a relationship between regularity and detection performance when the model is a classifier. There are other probabilistic algorithms, e.g., clustering, Bayesian modeling, Hidden Markov Model, etc. that can be used for anomaly detection. Can we use similar information-theoretic measures for these algorithms? And more fundamentally, can we select the best algorithm to build model based on the regularity of data? These questions are for our future work. Our conjecture is that the answers are "Yes" because all these algorithms (including classification) and the information-theoretic measures are all based on some probability theories.

Our experiments with regularity measure on sequential dependencies, i.e., conditional entropy, are all on fixed sequence length or time window models. In [5], it was shown that although a variable-length pattern matching model for *sendmail* data is more difficult to built, it can be more effective. Likewise, using variable time window based on network traffic load may also improve detection performance. Naively, from a conditional entropy plot, we can estimate the performance of various sequence lengths (or time windows), build multiple models with different sequence lengths, and select an appropriate models to use in run-time based on the relative conditional entropy between the sequences in run-time and in training. A better approach is to build an adaptive model that can dynamically adjust to different length based on run-time information. We will extend our approach to facilitate the construction of such models.

5 Related Work

Anomaly detection is an important research area in intrusion detection. In earlier systems [1, 11, 26], a normal profile for a user or program is usually based on statistical measures of the system features, e.g., the CPU usage, the number of shell commands used, etc. In several recent studies, learning-based approaches were applied to build anomaly detection models using system call data of privileged programs [6, 7, 14, 28]. Lane et al. [12] proposed a learning algorithm for analyzing user shell command history to detect anomalies. The algorithm attempts to address the "concept drift" problem, i.e., when the normal user behavior changes. EMERALD [24] uses statistical anomaly detection modules to monitor network traffics and a "resolver" to correlate alarms from misuse and anomaly detectors across an enterprise. While these systems all have some degree of success, they were developed for a particular kind of environment. The fundamental question of "how to build and evaluate anomaly detection model in general" has not been adequately addressed. As a result, the approaches developed in these studies may not be applicable to other environments.

Researchers have begun to develop principles and theories for intrusion detection. Axelsson [2] pointed out that the established field of detection and estimation theory bears similarities with the IDS domain. For example, the subject of an anomaly detection model corresponds to the "signal source" in detection and estimation theory, the auditing mechanism corresponds to "signal transmission", the audit data corresponds to the "observation space", and in both cases, the task is to derive detection rules. Therefore, the results from detection and estimation, which have been found applicable to a wide range of problems, may be used in the IDS domain. One of the key findings by Axelsson is that when building a detection model, *both* anomaly and intrusion data is needed to ensure detection performance. In previous work [16], we showed that using labeled training dataset with normal and intrusion connections, we can build highly effective classifier for intrusion detection. However, in practice, it is difficult to obtain intrusion data. In this work, we therefore focus on how to build anomaly detection models when only normal data is available for training. Another key finding by Axelsson is that a detection model should be optimized for some utility function, not necessary statistical accuracy, and instead could be some definition of cost. We have independently began to address how to build cost-sensitive IDS, i.e., an IDS that provides the best-valued protection [17].

The most related work is by Maxion et al. [21], where the relationship between data regularity and detection performance of anomaly detection model was studied. The study focused on sequence data, and hence regularity is defined as conditional entropy. The key result from experiments on synthetic data is that when an anomaly detection model is tested on data with a range of datasets with varying regularity values, the detection performance also varies. This suggests that the current practice of deploying a particular anomaly detection system across different environments is perhaps flawed and should be reconsidered. Our study here confirmed this finding in that we showed that the expected detection performance can be attained only when the relative conditional entropy between the training and testing datasets is small. Our study is more extensive because we used real system and network audit data in our case studies, but more importantly, we defined more information-theoretic measures and showed how to use them to build anomaly detection models.

6 Conclusion and Future Work

In this paper, we proposed to use some information-theoretic measures for anomaly detection. Entropy can be used to measure the regularity of an audit dataset of unordered records. Conditional entropy can be used to measure the regularity on sequential dependencies of an audit dataset of ordered records. Relative conditional entropy can be used to measure the similarity between the regularity measures of two datasets. Information gain of a feature describes its power in classifying data items. Information cost measures the computation cost of processing audit data by an anomaly detection. We discussed that these measures can be used to guide the model building process and to explain the performance of the model.

In the case studies on *sendmail* system call data, we showed that we can use conditional entropy to determine the appropriate sequence length for accuracy only or for the trade-off between accuracy and cost, a problem that has been posed but not solved by the community. We showed that when relative conditional entropy is low, the detection performance on the testing dataset is comparable to that on the training dataset. In the case study on network data, we showed that entropy can be used to direct the partitioning of a dataset (i.e., refining the subject) and build better models. We also showed evidence that conditional entropy can be used to guide the construction of temporal and statistical features.

Although our work is still preliminary (especially in some of the case studies), we are very encouraged by the results thus far. We have intended to show that despite the need for expert domain knowledge when building an IDS, theoretical understandings and tools are not only necessary, but also possible. Although one may argue that our results are obvious (or not surprising), we feel that it is very important to develop a formal framework, even just for stating the obvious, so that the field of intrusion detection can progress more rapidly and rigorously.

There is much remained to be explored. Besides conducting more comprehensive experiments and evaluations, we will study how to extend our information-theoretic measures to accommodate algorithms other than classification for building anomaly detection models. We will study how to determine the best algorithm to use based on regularity of the data. We will also study how to build model with variable sequence length or time window.

References

[1] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion detection expert system (NIDES): A summary. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.

[2] S. Axelsson. A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical report, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 2000.

[3] W. W. Cohen. Fast effective rule induction. In *Machine Learning: the 12th International Conference*, Lake Taho, CA, 1995. Morgan Kaufmann.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[5] H. Debar, M. Dacier, M. Nassehi, and A. Wespi. Fixed vs. variable-length patterns for detecting suspicious process. In *Proceedings of the 1998 ESORICS (LNCS 1485)*, 1998.

[6] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[7] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *Proceedings of the 8th USENIX Security Symposium*, August 1999.

[8] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[9] V. Jacobson, C. Leres, and S. McCanne. *tcpdump.* available via anonymous ftp to ftp.ee.lbl.gov, June 1989.

[10] S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.

[11] Los Alamos National Laboratory. Wisdom and sense guidebook. Los Alamos National Laboratory.

[12] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. In *Proceedings of 5th ACM Conference on Computer & Communication Security*, 1998.

[13] W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems.* PhD thesis, Columbia University, June 1999.

[14] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[15] W. Lee, S. J. Stolfo, and P. K. Chan. Learning patterns from Unix process execution traces for intrusion detection. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 50–56. AAAI Press, July 1997.

[16] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.

[17] Wenke Lee, Wei Fan, Matt Miller, Sal Stolfo, and Erez Zadok. Toward cost-sensitive modeling for intrusion detection and response. In *1st ACM Workshop on Intrusion Detection Systems*, 2000.

[18] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunninghan, and M. Zissman. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, January 2000.

[19] T. Lunt. Detecting intruders in computer systems. In *Proceedings of the 1993 Conference on Auditing and Computer Technology*, 1993.

[20] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real-time intrusion detection expert system (IDES) - final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[21] R. A. Maxion and K. M. C. Tan. Benchmarking anomaly-based detection systems. In *Proceedings of the 1st International Conference on Dependable Systems & Networks*, 2000.

[22] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[23] V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[24] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *National Information Systems Security Conference*, Baltimore MD, October 1997.

[25] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[26] S. E. Smaha. Haystack: An intrusion detection system. In *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, 1988.

[27] SunSoft. *SunSHIELD Basic Security Module Guide*. SunSoft, Mountain View, CA, 1995.

[28] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.