

Network Anomaly Detection with Incomplete Audit Data [★]

Animesh Patcha and Jung-Min Park

*Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Email: {apatcha, jungmin}@vt.edu*

Abstract

With the ever increasing deployment and usage of gigabit networks, traditional network anomaly detection based Intrusion Detection Systems (IDS) have not scaled accordingly. Most, if not all, intrusion detection systems (IDS) assume the availability of complete and clean audit data. We contend that this assumption is not valid. Factors like noise, mobility of the nodes and the large amount of network traffic make it difficult to build a traffic profile of the network that is complete and immaculate for the purpose of anomaly detection. In this paper, we attempt to address these issues by presenting an anomaly detection scheme, called SCAN (Stochastic Clustering Algorithm for Network anomaly detection), that has the capability to detect intrusions with high accuracy even with incomplete audit data. To address the threats posed by network-based denial-of-service attacks in high speed networks, SCAN consists of two modules: an anomaly detection module that is at the core of the design and an adaptive packet sampling scheme that intelligently samples packets to aid the anomaly detection module. The noteworthy features of SCAN include: (a) it intelligently samples the incoming network traffic to decrease the amount of audit data being sampled while retaining the intrinsic characteristics of the network traffic itself; (b) it computes the missing elements of the sampled audit data by utilizing an improved Expectation-Maximization (EM) algorithm-based clustering algorithm; and (c) it improves the speed of convergence of the clustering process by employing Bloom filters and data summaries.

Key words: Network Anomaly Detection, Expectation–Maximization Algorithm, Sampling

[★] Portions of this material was presented at the Fourteenth International Conference on Computer Communications and Networks, San Diego, October 2005.

1 Introduction

Intrusion detection is an important component of a network's security system. It complements existing security technologies (such as firewalls) by providing crucial information to the network administrators about attacks and intrusions that may be undetected by existing security technologies. They also provide invaluable forensic information that will allow organizations to trace back the origins of attacks and aid in the prosecution of the attackers.

Intrusion detection systems (IDS) have traditionally been classified into two categories — *anomaly detection* and *misuse- or signature-based detection*. Misuse detection systems match incoming network traffic to a database of known attack signatures to detect intrusions. While a misuse detection system enjoys a high rate of success at detecting known attacks, they are ineffective at detecting new or unknown attacks. On the other hand, anomaly detection systems create a normal profile of the network or host under observation and flag deviations from the normal profile as probable intrusions. As these systems predict anomalous behavior, they have the advantage of being able to detect new and novel attacks.

However, IDS's have not kept pace with the rapidly evolving field of computer networking. An example is the domain of high speed networks especially gigabit networks, where the large amount of network data that the network produces is posing new challenges in anomaly detection. Prohibitively large volume of network data makes the tasks of storing, classifying, and labeling the data almost infeasible. We can, of course, obtain labeled data by simulating intrusions in a network. However, then we would be limited to the set of known attacks and we would not be able to detect new attacks. As a result, it has been seen that currently available commercial solutions to detect intrusions in gigabit networks can detect less than half of the attacks directed at them [1] at gigabit speeds.

Another assumption that most deployed IDS's work under is the assumption that they have available for analysis, *clean*¹, *complete* and *labeled* data. Traditionally, both anomaly detection and misuse techniques, have traditionally relied on the availability of *clean* and *complete* data for analysis. This, however, is not a valid assumption any more due to the emergence of new types of networks such as Mobile and Ad hoc Networks (MANETs). In MANETs, mobility, sleep patterns of mobile devices and other characteristics specific to mobile and wireless networks prevent the collection of complete audit data for analysis.

Therefore, the focus of this paper is the detection of intrusions in environments where it is not feasible to analyze complete network traffic as is the case in high

¹ By "*clean*" we mean training data that is free of attacks. The availability of attack free data is very important during the training phase of an anomaly detection system, as during this phase an anomaly detection creates a baseline of normal system usage.

speed IP networks and wireless ad hoc networks. We are particularly concerned with mining network data in such a dynamically changing, high volume environment for the express purpose of network anomaly detection in the presence of missing information. We contend that, network monitoring and data mining in high speed networks operates under a few constraints. For example, as pointed out in [2], capturing per packet information at the network edge and transporting it to a central data warehouse is unrealistic because of the storage and transportation costs. A more realistic approach would be to monitor the network traffic at key locations (network gateways and other egress and ingress points) and thereafter possibly share newly discovered intrusion details amongst the key locations. Sekar et al. [3] describe an approach to perform high performance analysis of network data, but unfortunately they do not provide experimental data based on live traffic analysis. Their claim of being able to perform real-time intrusion detection at 500 Mbps is based on the processing of off-line traffic log files.

Although there have been recent attempts [4, 5] to train anomaly detection models over noisy data, to the best of our knowledge, the research presented in this document is the first attempt to investigate the effect of incomplete data sets on the anomaly detection process. The anomaly detection approach we present in this paper follows the typical anomaly detection paradigm. We assume attack-free training data, but the outlier detection method we chose, is robust over small amount of missing audit data and noise.

The motivation behind our intrusion detection framework is straightforward: sampling reduces the amount of audit data that needs to be processed, thereby enabling anomaly detection in high-speed networks. In typical cases, sampling would lead to loss of information, leading to inaccurate predictions and/or false alarms. To avoid such a scenario, the proposed intrusion detection scheme, christened SCAN (Stochastic Clustering Algorithm for Network anomaly detection), includes two distinguishing modules:

- An adaptive sampling module that intelligently adapts the sampling rate to sample packets.
- A predictive data mining based anomaly detection module that has the capability to estimate the missing data and reduce the occurrence of false alarms.

At the core of SCAN is a stochastic clustering algorithm which is based on an improved version of the Expectation–Maximization (EM) algorithm [6]. The EM algorithm’s speed of convergence was accelerated by using a combination of Bloom filters, data summaries and associated arrays. The advantage of using the EM algorithm is that unlike other imputation techniques, EM computes the maximum likelihood estimates in parametric models based on prior information. The clustering approach that we propose can be used to process incomplete and unlabeled data. To the best of our knowledge, there have been no prior attempts to investigate the effects of incomplete audit data in anomaly detection.

The salient features of SCAN are:

- Ability to intelligently sample incoming network traffic to reduce the amount of audit data that needs to be processed without losing the inherent characteristics of the network traffic.
- Ability to detect anomalies with a high degree of accuracy in the absence of complete audit data.

The paper is organized as follows. In section 2 we survey the related work. Section 3 presents some background information related to the EM algorithm, the k-means clustering algorithm and the Hurst parameter. Section 4 gives a detailed description of the proposed anomaly detection system, SCAN. Section 5 presents our simulation results. Finally, Section 6 summarizes our findings and concludes the paper.

2 Related Work

In [7], Anderson et al. propose an anomaly detection system called NIDES. NIDES uses a frequency-based model in which the probability of an event is estimated by its average frequency during training. The model is based on the distribution of source and destination IP addresses and ports per transaction. NIDES models ports and addresses, and flagging differences between short and long term behavior.

PHAD (Packet Header Anomaly Detector) [8], LERAD (LEARNING Rules for Anomaly Detection) [9], and ALAD (Application Layer Anomaly Detector) [10] use time-based models in which the probability of an event depends on the time since it last occurred. For each attribute, the aforementioned schemes collect a set of allowed values and flag novel values as anomalous. PHAD, ALAD, and LERAD differ in the attributes that they monitor. PHAD monitors 33 attributes from the Ethernet, IP and transport-layer packet headers. ALAD models incoming server TCP requests such as source and destination IP addresses and ports, opening and closing TCP flags, and the list of commands (the first word on each line) in the application payload. Depending on the attribute, it builds separate models for each target host, port number (service), or host/port combination. LERAD models TCP connections. Although the data set is multivariate network traffic data containing fields extracted out of the packet headers, they break down the multivariate problem into a set of univariate problems and sum the weighted results from range matching along each dimension. Although this paradigm is computationally efficient and effective in detecting some network intrusions, breaking the multivariate data down into univariate data has significant drawbacks in detecting network-based DoS attacks. For example, an indicator for a SYN flood attack is the reception of a high volume of SYN requests along with a lower than normal ACK rates. However, either a higher SYN rate or lower ACK rate by itself is not an appropriate indicator for a SYN Flood attack. The former can occur when the network is busy and the latter can

occur when the network is idle. Thus, it is the combination of the two that should be regarded as the indicator of the SYN Flood attack.

In network intrusion detection, the network traffic that is available is primarily categorical². Many of the clustering approaches that process categorical data, such as ROCK [11], are often resource intensive. For example, ROCK has a high computational cost and requires sampling in order to scale to large datasets. Unsupervised approaches³ for detecting outliers in large dataset have been proposed [12, 13], but these approaches are primarily based on ordered data.

In [14, 15], Lee et al. examined the problems of combining misuse detection and anomaly detection in a supervised learning scenario. The purpose of these methods is to incorporate supervised information in the anomaly paradigm, which typically doesn't include attacks in the training data. Although incorporating supervised information leads to increased detection performance, it is often difficult to obtain.

Lee and Xiang [16] proposed to use several information-theoretic measurements such as entropy and information gain to help evaluate the quality of anomaly detection methods, determine the system parameters and build models for intrusion detection.

More recently, researchers have been looking at hardware based solutions that use FPGA's and network cards equipped with a NPU (Network Process Unit). These hardware solutions are able to compute traffic statistics directly on the card without having to move all the traffic to the computer. Kruegel et al. [17] describe a hardware architecture that splits incoming traffic into several intrusion detection sensors that work in parallel to identify intrusion attempts. An entirely different approach was proposed by Franklin et al. [18]. Their approach is to build an intrusion detection system using Non-Deterministic Finite Automata (NFA). The NFA approach has the advantage of being able to match quite complex regular expressions without the need to convert the NFA into a Deterministic Finite Automata (DFA). One drawback of this approach is the fact that it does not support dynamic reconfiguration. As a result, the FPGA designs need to be re-compiled following rule changes. The scheme proposed by Franklin et al. operates on a 8-bit data block and achieves FPGA clock rates of 30-120 MHz, dependent on the regular expression complexity. As a result, processing complex regular expressions would be (time) expensive.

Hardware-based solutions such as the one mentioned above have certain drawbacks; these include the following: (1) They are expensive; (2) They are available only for a few media types, usually for Ethernet and a few others; and (3) They have little memory on board, thus dramatically limiting the size of programs that can run on the card itself.

² Categorical data is data that fits into a small number of discrete categories.

³ Learning in which the system parameters are adapted using only the information of the input and are constrained by pre-specified internal rules.

A number of commercial products have come out onto the market to respond to the need for high speed IDSs. A number of vendors now claim to have sensors that can operate on high-speed ATM or Gigabit Ethernet links. For example, ISS [19] offers NetICE Gigabit Sentry, a system that is designed to monitor traffic on high-speed links. ISS advertises the system as being capable of performing protocol reassembly and analysis for several application-level protocols (e.g., HTTP, SMTP, and POP) to identify malicious activities. The tool claims to be the first network-IDS that can handle full Gigabit speeds. However, the tests [1] have shown that under real world conditions GigaSentry can only capture slightly more than 500,000 packets/second.

These comments show the actual difficulties of performing network-based intrusion detection on high-speed links. Other IDS vendors (like Cisco [20]) offer comparable products with similar features. Unfortunately, no experimental data gathered on real networks is presented. TopLayer Networks [21] presents a switch that keeps track of application-level sessions. The network traffic is split with regard to these sessions and forwarded to several intrusion detection sensors. Packets that belong to the same session are sent through the same link. This allows sensors to detect multiple steps of an attack within a single session. Unfortunately, the correlation of information between different sessions is not supported. This could result in undetected attacks when attacks are performed against multiple hosts (e.g., ping sweeps) or performed across multiple sessions.

3 Preliminaries

In this section, we review some of the background information that is relevant to understanding SCAN. Section 3.1 reviews simple random packet sampling, a widely utilized sampling technique. In this paper, we compare the performance of our adaptive sampling technique with the simple random packet sampling technique using the sample mean and the Hurst parameter (described in Section 3.2) as performance metrics. In Section 3.3, we give a brief description of the Expectation-Maximization algorithm which forms a basis for the predictive data mining algorithm described in this paper. We compare the performance of our algorithm with the popular k -means clustering algorithm which we describe briefly in Section 3.4.

3.1 Simple Random Packet Sampling

In simple random sampling, a sample of n individuals are drawn from a population of N individuals, in such a way that each individual has equal probability to be drawn. This implies that all combinations of n individuals have the same probability to be drawn.

The essential characteristics of simple random sampling are:

- Each member of the population has an equal chance of being picked
- Selection are independent

There are, however, a number of potential problems with simple random sampling. Firstly, it is time-consuming to draw a random sample one individual at a time. Secondly, if the population is widely dispersed, it is extremely costly to reach them. In other words, it will be expensive to sample outliers and some sections of the population may, by chance, be sampled too heavily and others too lightly. Lastly, determining the appropriate sample size often requires a statistician.

3.2 Self Similarity and the Hurst Parameter

In 1994, Leland et al. [22] performed a statistical analysis of Ethernet traffic in a local area network and showed that the Ethernet traffic consisted of slowly decaying packet count bursts across all time scales. Time series that consist of such a pattern are said to exhibit the property of long range dependence and are termed as “self-similar”⁴. Similar self-similar behavior has also been observed in wide area Internet traffic by other researchers [23]. We believe that a loss in self-similarity of the incoming network traffic is an important indicator of an ongoing attack [24] [25].

One important characteristic of a self-similar process is that its degree of self-similarity can be expressed with a single parameter, namely the Hurst parameter [26] which can be derived from the re-scaled adjusted range (R/S) statistic. It is defined as follows:

For a given set of observations X_1, X_2, \dots, X_n with sample mean, M_n defined as $(1/n)\sum_j X_j$, adjusted range $R(n)$ and sample variance S^2 , the rescaled adjusted range or the R/S statistic⁵ is given by

$$\frac{R(n)}{S(n)} = \left(\frac{1}{S(n)} \right) \left(\text{Max} \sum_k^{j=1} (X_j - M_n) - \text{Min} \sum_k^{j=1} (X_j - M_n) \right). \quad (1)$$

Hurst discovered that many naturally occurring time series are well represented by the relation

⁴ A self-similar time series has the property that when aggregated the new series has the same autocorrelation function as the original

⁵ The R/S statistic is the adjusted range of partial sums of deviations of a times series from its mean, rescaled by its standard deviation.

$$E \left[\frac{R(n)}{S(n)} \right] \sim cn^H, \text{ as } n \rightarrow \infty, \quad (2)$$

with the Hurst parameter H normally around 0.73, and a finite positive constant, c , independent of n . On the other hand, if the X_k 's are Gaussian pure noise or *short range dependent*, then $H = 0.5$ in equation (2), and the discrepancy is referred to as the *Hurst effect*.

3.3 The Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm is a general method of finding the maximum-likelihood estimate of the parameters of a distribution from a given data set when the data is incomplete or has missing values. There are two main applications of the EM algorithm. In the first application, the EM algorithm is applied when the data has missing values, due to problems or limitations of the observation process. In the second application the EM algorithm is used to optimize when optimizing the likelihood function is analytically intractable but when the likelihood function can be simplified by assuming the existence of values for additional but missing (or hidden) parameters. The latter application of the EM algorithm is more commonly seen in pattern recognition.

Let us suppose that,

- \mathcal{X} : Observed (incomplete data) that is generated by some distribution
- \mathcal{Y} : Missing/Unobserved data
- \mathcal{Z} : Complete data set such that $\mathcal{Z}=(\mathcal{X}, \mathcal{Y})$
- $p(z|\Theta)$: Joint conditional density function such that

$$p(z|\Theta) = p(x, y|\Theta) = p(y|x, \Theta)p(x|\Theta), \quad (3)$$

where Θ is the set of means and covariances and is also sometimes known as the *mixture model*. The density function described above arises from the marginal density function $p(x|\Theta)$ and the the assumption of hidden variables and parameter value guesses.

Given the density function, we can define a likelihood function as

$$\mathcal{L}(\Theta|\mathcal{Z}) = \mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y}|\Theta). \quad (4)$$

The likelihood function is also called the complete-data likelihood. The EM algorithm first finds the expected value of the complete-data log-likelihood $\log(p(\mathcal{X}, \mathcal{Y}|\Theta))$

with respect to the unknown data \mathcal{Y} given the observed data \mathcal{X} and the current parameter estimates.

We use the EM algorithm to estimate the parameters of a mixture of k distributions. The multivariate Gaussian density function for vector \mathcal{X} on a d -dimensional space for cluster j , where $j \in \{1, \dots, k\}$, is

$$p(x, M_j, C_j) = \left(\frac{1}{\sqrt{((2\pi)^d |C|)}} \right) e^{-\frac{1}{2}(x-M_j)^T C_j^{-1} (x-M_j)}, \quad (5)$$

where M_j is the d -dimensional mean vector and C_j is a $d \times d$ diagonal matrix representing the covariance vector. The index i is used for points and the index j for clusters.

The EM algorithm makes use of the Mahalanobis distance. Using Mahalanobis distance is particularly useful for skewed data having different sizes. The covariance matrix C_j is used to scale each dimension for distance computation. The squared Mahalanobis distance of point y_i to cluster j is

$$\delta^2(y_i, M_j, C_j) = \delta_{ij}^2 = (y_i - M_j)^T C_j^{-1} (y_i - M_j). \quad (6)$$

The EM algorithm takes a dataset containing n points as an input. The output of the algorithm is the average logarithmic likelihood $L(\theta)$ which is a measure of the quality of the solution and is defined as

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n \log(P(x_i; \Theta)), \quad (7)$$

where Θ is the *mixture model*. The algorithm clusters the dataset into k clusters.

EM starts from an approximation to Θ . It has two major steps: the Expectation (E) step and the Maximization (M) step. EM iteratively executes the E step and the M step as long as the change in $L(\theta)$ is greater than an accuracy threshold ϵ or as long as a maximum number of iterations has not been reached. The E step computes $P(x_i, M_j, C_j)$ and $P(x_i, \theta)$. The M step updates Θ based on the probabilities computed in the E step. EM is theoretically guaranteed to monotonically increase $L(\theta)$ in each iteration and to converge to a locally optimal solution.

The two major drawbacks of the EM algorithm are that it is slow to converge and for many real world problems, the E- or M-steps may be analytically intractable. To alleviate this and other problems, a number of strategies have been proposed. The most important previous work comes from the machine learning community. Neal *et al.* [27] introduce the idea of using sufficient statistics⁶. In this work, the

⁶ A quantity $T(X)$ that depends on the (observable) random variable X but not on the

authors analyze several variants of EM, conceptualizing the logarithmic-likelihood optimization as an optimization of an energy function. The key idea behind one of the variants presented in their work is to use sufficient statistics. They describe a simple mixture problem with a dimensionality of one and a cluster size of two. They also show experimental evidence of the superiority of their approach. However, their study is incomplete as it does not address the problem of clustering large datasets with high dimensionality. Problems related to numerical stability, proper initialization, and outlier handling are not addressed in their work either. In [28], Bradley et al. present a scalable EM (SEM) algorithm that iterates in memory and also summarizes points through their sufficient statistics. To avoid locally optimal solutions, they re-seed empty clusters and estimate several models concurrently, sharing information across models. The scheme presented by the authors, requires the user to specify what fraction of the working buffer should be from the entire database. The value for this parameter is typically 1%. The authors, however, do not explain why this value gives the best results. BIRCH [29] and newer extensions of the classical EM algorithm [28, 30] use data summarization to accelerate convergence.

3.4 The *K-Means Clustering Algorithm*

The *k*-means algorithm [31] was first introduced by MacQueen in 1967. It is one of the most commonly used clustering algorithms to group data with similar characteristics. The *k*-means clustering algorithm assigns multivariate observations to a pre-determined number of groups (*k*). Each of the *k* groups consists of m_k data items and a group centroid (y_k). In the beginning, the algorithm randomly selects *k* initial clusters from the dataset as the group centroids. Then the *k*-means algorithm calculates the arithmetic mean of each cluster formed in the dataset. In each of the first *k* initial clusters, there is only one record.

Every observation, thereafter, is assigned to the one group to whose centroid they are nearest by using a distance or similarity metric. Once all observations are allocated to groups, the new group centroids are then recalculated and the original observations reallocated to the new groups. The process is repeated until the cluster's arithmetic mean does not shift more than a given cut-off value or the iteration limit is reached.

The drawback of the *k*-means algorithm is that the quality of the local optimum strongly depends on the initial guess (either the centers or the assignments). If we start with a wild guess for the centers it would be fairly unlikely that the process would converge to a good local minimum. Another disadvantage of the *k*-means

(unobservable) parameter θ is called a statistic. A sufficient statistic captures all of the information in X that is relevant to the estimation of θ .

algorithm is that the answers it gives are not unique. Therefore it is difficult to compare the quality of the clusters produced.

4 Description of SCAN

In this paper we describe the design and development of a network based IDS that can be deployed in high speed networks. To achieve this goal, SCAN's architecture combines intelligent sampling and flow aggregation with data reduction and anomaly detection to achieve a high degree of accuracy in detecting intrusions with partial audit data. The design requirements for such a network based IDS were (a) stateless inspection of packets, protocols and/or packet headers at wire speed, (b) low occurrence of false alarms and high detection rate, (c) ability to track TCP states, and (d) ability to report events and/or alarms. Based on these requirements SCAN (see Figure 1), is composed of the following five modules:

- (1) Adaptive weighted packet sampling
- (2) Flow aggregation
- (3) Clustering
- (4) Data reduction
- (5) Anomaly detection

The *adaptive weighted sampling* module intelligently samples incoming network traffic to reduce the amount of network traffic that has to be processed without losing the inherent characteristics of the network traffic. The sampled traffic is then aggregated into flows in the *flow aggregation* module based on the destination host and port information. The aggregated flows are then *clustered* and simultaneously several data summaries are extracted from the flow information for every time slice. Data summarization has two advantages. Firstly the summaries are used to increase the speed of convergence of the clustering algorithm and secondly data summarization enables *data reduction*. The last step in SCAN involves performing *anomalous flow detection* on the clustered data.

Fig. 1. System model.

4.1 Adaptive Weighted Packet Sampling

Traffic measurement and monitoring serves as the basis for a wide range of IP network operations and engineering tasks such as trouble shooting, accounting and usage profiling, routing weight configuration, load balancing, capacity planning, etc. Traditionally, traffic measurement and monitoring is done by capturing every packet traversing a router interface or a link. With today's high-speed (e.g., Gigabit or Terabit) links, such an approach is no longer feasible due to the excessive overheads it incurs on line-cards or routers. As a result, packet sampling has been suggested as a scalable alternative to address this problem. In this paper we have investigated two sampling techniques, viz., *simple random packet sampling* and *adaptive weighted packet sampling*.

Given the dynamic nature of network traffic, static sampling does not always ensure the accuracy of estimation, and tends to over sample at peak periods when efficiency and timeliness are most critical. More generally, static random sampling techniques do not take traffic dynamics into account, thus they cannot guarantee that the sampling error in each block falls within a prescribed error tolerance level.

In the commercial world, *NetFlow* is a widely deployed general purpose measurement feature of Cisco and Juniper routers. The volume of data produced by NetFlow is a problem in itself. To handle the volume and traffic diversity of high speed backbone links, NetFlow resorts to 1 in N packet sampling. The sampling rate is a configuration parameter set manually and seldom adjusted. Setting it too low, causes inaccurate measurement results; setting it too high, can result in the measurement module using too much memory and processing power, especially when faced with increased volume of traffic or unusual traffic patterns.

In other words, under some traffic loads, simple periodic sampling may be poorly suited to the monitoring task. During periods of idle activity or low network loads, a long sampling interval provides sufficient accuracy at a minimal overhead. However, bursts of high activity require shorter sampling intervals to accurately measure network status at the expense of increased sampling overhead. To address this issue, *adaptive sampling* techniques can be employed to dynamically adjust the sampling interval and optimize accuracy and overhead.

In this paper, we investigated adaptive sampling techniques to intelligently sample the incoming network traffic and aid the network anomaly detection process. The main thrust of this phase of the paper has been the development of an intelligent adaptive sampling scheme that samples the incoming network traffic by taking into account the past N observations to determine and control the next sampling interval. The sampled data that was collected in this phase is then used by our estimation algorithm, which uses a statistical estimation procedure, to detect anomalous behavior in the network.

Adaptive sampling techniques dynamically adjust the sampling rate based on the observed sampled data. A key element in adaptive sampling is the prediction of future behavior based on the observed samples. The weighted prediction method described in this section predicts the value of the next sampling interval based on the past N samples. Inaccurate predictions indicate a change in the network behavior and require an increased/decreased sampling rate to determine the new pattern.

Let us assume that the vector, \bar{Z} , holds the values of the N previous samples, such that $Z[N]$ is the most recent sample and $Z[1]$ is the oldest sample. Having fixed a window size of N , when the next sampling occurs, the vector is right shifted such that $Z[N]$ replaces $Z[N-1]$ and $Z[1]$ is discarded. The weighted prediction model will therefore be the prediction of $Z[N]$ given $Z[N-1], \dots, Z[1]$. In general, we can express this predicted value as a function of the N past samples i.e.

$$\hat{Z}[N] = \Phi(Z[N-1], \dots, Z[1]) = \bar{\alpha}^T \bar{Z}, \quad (8)$$

where $\hat{Z}[N]$ is the new predicted value, \bar{Z} is the vector of past $N-1$ samples, and $\bar{\alpha}^T$ are predictor coefficients distributed such that newer values have a greater impact on the predicted value $\hat{Z}[N]$. A second vector, t , records the time that each sample is taken and is shifted in the same manner as Z . The objective of the weighted prediction method is to find appropriate values of $\bar{\alpha}^T$ such that the sum of the square of the errors between the predicted and actual values is minimized, i.e.

$$S = \sum_{i=1}^N w_i (Z_i - \hat{Z}_i)^2, \quad (9)$$

where w_i denotes the weight in the i 'th interval.

The coefficient vector is given by:

$$\hat{\alpha}^T = (\bar{Z}^T W \bar{Z})^{-1} \bar{Z}^T W y, \quad (10)$$

where W and W_y are $N \times N$ diagonal weight matrices and represents the individual weights in these matrices. The weights are determined according to two criteria:

- (1) The ‘‘freshness’’ of the past N samples. The more recent a sample is the greater is its weight.
- (2) The similarity between the predicted value at the beginning of the time-slot and the actual current value. The similarity between two vectors is measured by the distance between them. The smaller the distance is, the more similar they are to each other.

Based on the above two criteria, we define a weight coefficient as

$$w_i = \frac{1}{(t[N] - t[i])} \left(\frac{1}{|Z[N] - \hat{Z}[N]|^2 + \eta} \right), \quad (11)$$

where η is a quantity introduced to avoid a division by zero error.

The predicted output, $\hat{Z}[N]$, which has been derived from the previous N samples, is then compared with the actual value of the sample, $Z[N]$. A set of rules is applied to adjust the current sampling interval, $\Delta T_{Curr.} = t[N] - t[N-1]$, to a new value, ΔT_{New} , which is used to schedule the sampling query. The rules used to adjust the sampling interval compare the rate of change in the predicted sample value, $\hat{Z}[N] - Z[N-1]$, to the actual rate of change, $Z[N] - Z[N-1]$. The ratio between the two rates is defined as R , where

$$R = \left| \frac{\hat{Z}[N] - Z[N-1]}{Z[N] - Z[N-1]} \right|. \quad (12)$$

The value of R will be equal to 1 when the predicted behavior is same as the observed behavior. We define a range of values $R_{MIN} \leq 1 \leq R_{MAX}$, such that

$$\begin{aligned} R < R_{MIN} &\Rightarrow \Delta T_{New} < \Delta T_{Curr.} \Rightarrow \Delta T_{New} = (R) \times \Delta T_{Curr.} \\ R_{MIN} < R < R_{MAX} &\Rightarrow \Delta T_{New} = 2 \times \Delta T_{Curr.} \\ R > R_{MAX} &\Rightarrow \Delta T_{New} < \Delta T_{Curr.} \Rightarrow \Delta T_{New} = (1 + R) \times \Delta T_{Curr.} \\ R_{undefined} &\Rightarrow \Delta T_{New} = 2 \times \Delta T_{Curr.} \end{aligned} \quad (13)$$

4.2 Flow Aggregation

In the context of SCAN, a *flow* is all the connections with a particular destination IP address and a port number combination. The measurement and characterization of the sampled data proceeds in time slices. We process the sampled packets into *connection records*. Because of its compact size as compared to other types of records (e.g., packet logs), connection records are appropriate for data analysis. A connection record provides the following fields:

$$(SrcIP, SrcPort, DstIP, DstPort, ConnStatus, Proto, Duration),$$

where

SrcIP and *DstIP* : Source and destination IP addresses
SrcPort and *DstPort* : Source and destination port numbers
ConnStatus : Status of connection (closed/open)
Proto : Network layer protocol identifier
Duration : Duration of connection

Each field can be used to correlate network traffic data and extract intrinsic features of each connection.

The flow aggregation algorithm employed by SCAN is shown in Fig. 2. Bloom filters [32] and associated arrays are used to store, retrieve and run membership queries on incoming flow data in a given time slice. When a packet is sampled, we first identify the flow the packet belongs to by executing a membership query in the Bloom filter based on the flow's *flow ID*. The *flow ID* is a unique ID that is generated from a combination of the destination IP address and the destination port number. If an existing *flow ID* cannot be found, a new *flow ID* is generated based on the information from the connection record. This is called the *insertion phase*. If the *flow ID* exists, the corresponding array entry that is being used to track the flow is updated. If the *flow ID* does not exist, we run the *flow ID* through each of the k hash functions (that form a part of the Bloom filter), and use the result as an offset into the bit vector, turning on the bit we find at that position. If the bit is already set, we leave it on.

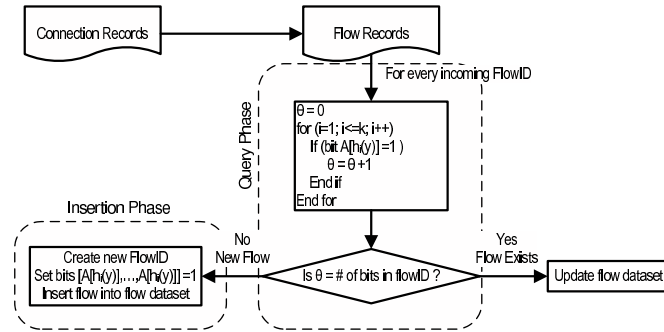


Fig. 2. Online sampling and flow aggregation algorithm.

Bloom filters [32] were first proposed as a space efficient data structure for answering approximate membership queries over a given set. One drawback of Bloom filters is the risk of false positives (i.e. returning a “yes” indicating that a given element is in the set when it is not). The false positive probability is a function of the length of the filter and the number of items stored in it.

To succinctly represent a set $S = \{x_1, x_2, \dots, x_n\}$ of n elements and support membership queries, we first select an array A consisting of m 1-bit elements and initialize them to zero. In addition, we select k distinct hash functions h_1, \dots, h_k each with a range of $[1, m]$. During the *insertion phase*, for each element $x \in S$, the bits at the position $h_1(x), \dots, h_k(x)$ in A are set to 1. In the *query phase*, to check if an element

y is in S , we check the value of the bits at positions $h_1(y), \dots, h_k(y)$ in A . The answer to the query is a *yes* if all of these bits are one and *no* otherwise.

4.3 POTION: EM Algorithm-Based Clustering Algorithm

The goal of clustering is to divide flows into natural groups. The instances contained in a cluster are considered to be similar to one another according to some metric based on the underlying domain from which the instances are drawn. In this section, we describe POTION (exPectatiOn maximizaTIon algOriThm for aNomaly detection), SCAN’s EM-based clustering algorithm which has advantages over the k -means clustering algorithm in the context of anomaly detection.

A stochastic clustering method (such as the EM algorithm) assigns a data point to each group with a certain probability. Such statistical approaches make sense in practical situations where no amount of training data is sufficient to make a completely firm decision about cluster memberships. Stochastic clustering algorithms aim to find the most likely set of clusters given the training data and prior expectations. The underlying model, used in the stochastic clustering algorithm, is called a *finite mixture model*. A mixture is a set of probability distributions—one for each cluster—that models the attribute values of the members of that cluster.

In the finite mixture model under consideration, we consider a scenario where we assume that the distribution \mathbf{T} that describes incoming network traffic⁷ to a server s is distributed as a mixture of two populations [4]: a *normal* (\mathbf{N}) distribution and an *anomalous* (\mathbf{A}) distribution such that the i^{th} traffic element⁸, x_i , in a given time slice t is generated from the *normal* distribution \mathbf{N} with a probability of p or from an *anomalous* distribution \mathbf{A} with a probability of $1 - p$. In other words,

$$\mathbf{T} = p\mathbf{N} + (1 - p)\mathbf{A}. \quad (14)$$

During the training phase, we set the baseline activity profile assuming that all incoming traffic elements are generated from the normal distribution \mathbf{N} .

The EM algorithm used by SCAN to cluster data is shown in Figure 3. The EM algorithm is a general method of finding the maximum-likelihood estimate of the parameters of a distribution from a given data set when the dataset has missing values. For this discussion, let us assume that we have a density function $P(Z|\Theta)$ that is governed by the set of parameters Θ . In addition, we have a dataset of size n drawn from this distribution, $Z = \{z_1, \dots, z_n\}$. The function $L(\Theta|Z)$ is called the likelihood function and is defined as the likelihood of the parameters given the

⁷ It should be pointed out here that the incoming traffic could be composed of either packets, flows or connections.

⁸ The term *traffic element* is used to denote any of the elements in one connection record.

dataset, i.e.

$$L(\Theta|Z) = P(Z|\Theta) = \prod_{i=1}^n P(z_i|\Theta). \quad (15)$$

In a maximum likelihood problem, our goal is to find Θ that maximizes L .

POTION starts with an initial guess for the parameters of the mixture model for each cluster and then iteratively applies the *Expectation Step (E)* and the *Maximization Step (M)* in order to converge to the maximum likelihood fit. In the E-step, the EM algorithm first finds the expected value of the complete-data log-likelihood, given the observed data and the parameter estimates. In the M-step, the EM algorithm maximizes the expectation computed in the E-step. The two steps of the EM

Input: Dataset D and initial estimates
Output: Clustered set of data points.
Algorithm:

- (a) **Initialize:** Set an initial guess for the probability p to an arbitrary non-singular choice $p_j^{[0]}$
- (b) **Iterate:** At step k
 - (i) **E Step:** Determine cluster memberships. Compute the sufficient statistics, assuming the parameter values $p_j^{[k]}$.
 - (ii) **M Step:** Derive $p_j^{[k+1]}$ as a maximum likelihood estimate based on the sufficient statistics collected in the E-Step
- (c) **Terminate:** Terminate when
 - 1. The difference in the log-likelihood between two consecutive steps
$$L(T, p_j^{[k+1]}) - L(T, p_j^{[k]}) \leq \epsilon.$$

where $L()$ is the likelihood function
and ϵ is the accuracy threshold (We use $\epsilon=10^{-3}$)
 - OR**
 - 2. Number of iterations = N (We use N=100),
whichever comes first

Fig. 3. EM algorithm for clustering.

algorithm are repeated until an increase in the log-likelihood of the data, given the current model, is less than the accuracy threshold ϵ .

The EM algorithm is *guaranteed* to converge to a local maximum which may or may not be the same as the global maximum. Typically, the EM algorithm is executed multiple times with different initial settings for the parameter values. Then a given data point is assigned to the cluster with the largest log-likelihood score.

The EM Algorithm extends the traditional k -means algorithm in two important ways. First, instead of assigning cases or observations to clusters to maximize the differences in the value of means for continuous variables, the EM clustering algorithm computes probabilities of cluster memberships based on one or more probability distributions. The goal of the clustering algorithm is to maximize the overall probability or likelihood of the data, given the final clusters. Second, unlike the

classic implementation of k -means clustering, the general EM algorithm can be applied to both continuous and categorical variables.

The classical EM algorithm does have a few disadvantages. In general, the algorithm is hard to initialize and the quality of the final solution depends on the quality of the initial solution. It may also converge to a poor locally optimal solution. This means that a solution may be acceptable, but is far from the optimal one. The EM algorithm needs an unknown number of iterations to converge to a good solution. That is, it is hard to set a threshold on the maximum number of iterations that is sufficient. The EM algorithm usually requires several passes over a dataset.

4.4 Data Summaries for Data Reduction

Our aim was to design a clustering algorithm which is fast and requires only a few passes over the data. The data space is assumed to contain data points with both numerical and categorical attributes. The attributes of each data point are the *words* from the corresponding audit file line. Note that we are using the term *word* to loosely describe a monitored attribute (e.g., IP addresses, port numbers, etc.). From the flow records that have been hashed and stored in the Bloom filters and the associated arrays, we calculate *data summaries* for every time slice. A list of data summary items are shown in Fig. 4. Data reduction techniques like sufficient statistics and data summaries are often used to enhance the speed of convergence of the EM algorithm [30].

Our enhancements to the classical EM algorithm consist of three stages:

- Stage I: At the end of each time slice, we make a pass over the connection record dataset and build data summaries. Data summaries are particularly important in the EM algorithm framework as they reduce the I/O time by avoiding repeated scans over the data. In addition, data summaries allow periodic parameter estimation as the records are being read. This enhances the speed of convergence of the EM algorithm.
- Stage II: We make another pass over the dataset to build cluster candidates using the data summaries collected in the first stage. After the frequently appearing words have been identified, the algorithm builds all cluster candidates during this pass. The cluster candidates are hashed and stored in Bloom filters and associated arrays. The dataset is processed line by line. When a line is found to belong to one or more dense regions (i.e. one or more frequently appearing words have been discovered in the line), a cluster candidate is formed. This is the E-step of the EM algorithm, where the cluster membership is determined. If the cluster candidate is not present in the Bloom filter, it is inserted into the filter with the value of one. If the cluster candidate is present in the Bloom filter, the corresponding value in the array is incremented

- *Flow Concentration Factor*: Total number of TCP flows that have c as the source, s as the destination server and p as the destination port in the i^{th} time slice.
- *Total number of data points per cluster*: Number of points per cluster.
- *Percentage of control packets*: Percentage of control packets is specific to an application. A change in the rate should identify an anomaly.
- *Percentage of data packets*
- *Average flow duration over all flows*
- *Maximum number of flows to a particular service*
- *Average flow duration per destination*
- *Sum of points*: Sum of all the points y_i in the j^{th} cluster Y_j i.e.

$$S_j = \sum_{\forall(y_i \in Y_j)} y_i. \quad (16)$$

- *Sum of square of points*: Sum of square of all the points y_i in the j^{th} cluster Y_j i.e.

$$SS_j = \sum_{\forall(y_i \in Y_j)} (y_i y_i^T). \quad (17)$$

- *Percentage of same service to same host*: Percentage of incoming traffic that originates from a particular source port and terminates at a particular destination IP address.
- *Percentage of same host to same service*: Percentage of incoming traffic that originates from a particular source IP address and terminates at a particular destination port.
- *Resent rate*: Number of bytes that have been resent. Control packets count as one byte.
- *The variance of the count of packets*: This variance is an indicator of the distribution of hosts contacted in a time interval. During the event of a port scan, a high variance in the packet counts across all source-destination pairs should reveal an unusual spread in the number of machines contacted in an interval
- *Wrong resent rate*: Number of bytes that were sent even after being acknowledged.
- *Duplicate ACK rate*: Number of duplicate acknowledgements received.
- *Data bytes sent in either direction*: Tracks the number of data bytes exchanged per flow.

Fig. 4. Data summaries.

- **Stage III**: Clusters are selected from the set of candidates.

Learning steps (periodic M-steps) are used to accelerate convergence. By using data summaries, the clustering algorithm can run the M-step at different frequencies while scanning the connection records. We can minimize the convergence time by running the M-step after every point. Unfortunately, the EM algorithm would not produce the globally optimal solution in such a scenario. On the other hand, we can produce a solution that is closer to the globally optimum solution by executing the M-step after all the n points have been read (as in the classical EM algorithm). This, however, would require a longer convergence time. Therefore, it is desirable to set the frequency of M-step execution in between the two aforementioned cases, but closer to the latter scenario. When a good approximation to the solution has been reached, we can execute normal EM iterations until the algorithm converges. During the final step of the clustering algorithm, the Bloom filter and the associated arrays are inspected, and the regions with values equal or greater than the threshold value are reported by the algorithm as clusters.

4.5 Anomalous Flow Detection

The first task in an anomaly detection process is *network baselining*. Network baselining can be defined as the act of measuring and rating the performance of a network. Providing a network baseline requires evaluating the normal network utilization, protocol usage, peak network utilization, and average throughput of the network usage over a period of time. Our approach to network baselining recognizes the fact that any parametric model of network traffic is an approximation to reality. Since our ultimate goal is anomaly detection, which we formulate as detecting a marked change in the baseline model parameters, we will not be addressing the more general problem of parameter estimation as an intermediate step to anomaly detection.

In our model, elements of the network flow that are anomalous are generated from the anomalous distribution (**A**), and normal traffic elements are generated from the normal distribution (**N**). Therefore, the process of detecting anomalies involves determining whether the incoming flow belongs to distribution **A** or distribution **N**. We use an approach similar to [4] and calculate the likelihood of the two cases to determine the distribution to which the incoming flow belongs to. We assume that for normal traffic, there exists a function F_N which takes as parameters the set of normal elements N_t , at time t , and outputs a probability distribution P_{N_t} over the data T generated by distribution **T**. Similarly, we have a function F_A for anomalous elements which takes as parameters the set of anomalous elements A_t and outputs a probability distribution P_{A_t} . Assuming that p is the probability with which a flow x_t belongs to **N**, the likelihood L_t of the distribution **T** at time t is defined as

$$\begin{aligned}
 L_t(\mathbf{T}) &= \prod_{k=1}^n P_T(x_k) \\
 &= \left[(p)^{|N_t|} \prod_{x_i \in N_t} P_{N_t}(x_i) \right] \left[(1-p)^{|A_t|} \prod_{x_j \in A_t} P_{A_t}(x_j) \right]. \quad (18)
 \end{aligned}$$

The likelihood values are often calculated by taking the natural logarithm of the likelihood rather than the likelihood itself because likelihood values are often very small numbers that are close to zero. We calculate the log-likelihood as follows:

$$\begin{aligned}
 LL_t(\mathbf{T}) &= |N_t| \ln(p) + \sum_{x_i \in N_t} \ln P_{N_t}(x_i) \\
 &\quad + |A_t| \ln(1-p) + \sum_{x_j \in A_t} \ln P_{A_t}(x_j). \quad (19)
 \end{aligned}$$

We measure the likelihood of each flow, x_t , belonging to a particular group by comparing the difference $LL_t(\mathbf{T}) - LL_{t-1}(\mathbf{T})$. If this difference is greater than some value α , we declare the flow anomalous. We repeat this process for every flow in the time interval. Note that we have to recompute the probability distributions

$P_{N_t}(x_t)$ and $P_{A_t}(x_t)$ at every step because of changes in their distributions.

5 Simulation Results

Before we present the details of the simulation results, we would like to set forth three important assumptions that we have made. These are: (1) user and/or network activities are observable, (2) normal and intrusive activities each have distinctive behavior, and (3) anomalous activity forms a small percentage of total network activity.

We evaluated SCAN using a synthetic dataset that was generated by combining the data from the 1999 DARPA intrusion detection project [33] and the Widely Integrated Distributed Environment (WIDE) project [34]. The WIDE backbone network consists of links of various speeds, from 2Mbps (Constant Bit Rate) CBR ATM up to 10Gbps Ethernet. The WIDE dataset we analyzed consisted of approximately 40GB of data from a 24 hour long trace that was collected on September 22, 2005.

The 1999 DARPA dataset that we analyzed consisted of five weeks of TCPdump data. Week 1 and 3 have normal attack-free network traffic. Week 2 consists of network traffic with labelled attacks, while week 4 and week 5 contain 201 instances of 58 different attacks, 177 of which are visible in the inside tcpdump data. We trained our model on the DARPA dataset using week 1 (5 days) and week 3 (7 days), then evaluated the detector on weeks 4 and 5. Like the original participants of the 1999 DARPA Intrusion Detection Evaluation team, evaluation is done in an off-line manner. For our experiments, we only use the inside TCPdump data.

The TCPDump files consisted of hundreds of thousands of lines of network data. Perl scripts were used to parse the TCPDump files into connection records. The connection records were then used to build the data summaries at the end of every time interval. We also aggregated the incoming network traffic into flows based on the flowID, and calculated various network parameters (see Fig. 4) for every time interval. The duration of a time interval is 60 seconds. The resulting flow records and the data summaries were stored in plain text files, which were then used as inputs to the clustering algorithm. SCAN was implemented in the *MATLAB* environment.

The experimental evaluation of SCAN has been divided into three steps. In Section 5.1, we evaluate the performance of the sampling algorithm and compare its performance with the simple random sampling algorithm. Section 5.1 is followed by Section 5.2, where we compare the performance of the EM algorithm based clustering algorithm with the traditional k -means clustering algorithm. Lastly, in Section 5.3 we evaluate the performance of the anomalous flow detection algo-

rithm.

5.1 Evaluation of the Sampling Algorithm

This section describes a set of simulations that were performed to compare the performance of the proposed adaptive sampling technique with the simple random sampling technique. The metrics used to compare the sampled data sets are also described in this section.

5.1.1 Experimental Setup

Simulations were conducted to compare and evaluate the performance of the proposed adaptive sampling algorithm with the simple random sampling algorithm. The proposed adaptive sampling algorithm as well as the simple random sampling algorithm were implemented and simulated in an offline manner using the Perl programming language. The Perl scripts were implemented on an Intel[®] Pentium 4 laptop with 1 gigabyte of RAM, 100 gigabyte of hard disk space and running the Slackware[®] Linux operating system. The algorithms were evaluated using data from the Widely Integrated Distributed Environment (WIDE) project [34]. As mentioned above, the WIDE backbone network consists of links of various speeds, from 2Mbps CBR (Constant Bit Rate) ATM up to 10 Gbps Ethernet.

5.1.2 Metrics for Evaluation

In order to compare the performance of the proposed adaptive sampling algorithm with the simple random sampling algorithm, a useful criterion to use is the mean square error (MSE) of the estimate or its square root, the root mean squared error, measured from the population that is being estimated. Formally we can define the mean square error of an estimator X of an unobservable parameter θ as $MSE(X) = E[(X - \theta)^2]$. The root mean square error is the square root of the mean square error and the root mean square error is minimized when $\theta = E(X)$ and the minimum value is the standard deviation of X .

In the second set of experiments, we verified whether the traffic data sampled by the proposed sampling scheme has the self similar property. For this verification, we used two different parameters: the mean of the packet count and the Hurst parameter. The peak-to-mean ratio (PMR) can be used as an indicator of traffic burstiness. PMR is calculated by comparing the peak value of the measure entity with the average value from the population. However, this statistic is heavily dependent on the size of the intervals, and therefore may or may not represent the actual traffic characteristic. A more accurate indicator of traffic burstiness is given by the Hurst

parameter. The Hurst parameter (H), as mentioned above in Section 3.2, is a measure of the degree of self-similarity. In this paper we use the R-S statistical test to obtain an estimate for the Hurst Parameter (H). We run the test on both the original and the sampled data.

5.1.3 Experimental Results

In figure 5, we compare the proposed sampling scheme with the simple random sampling algorithm using the standard deviation of packet delay as the comparison criterion.

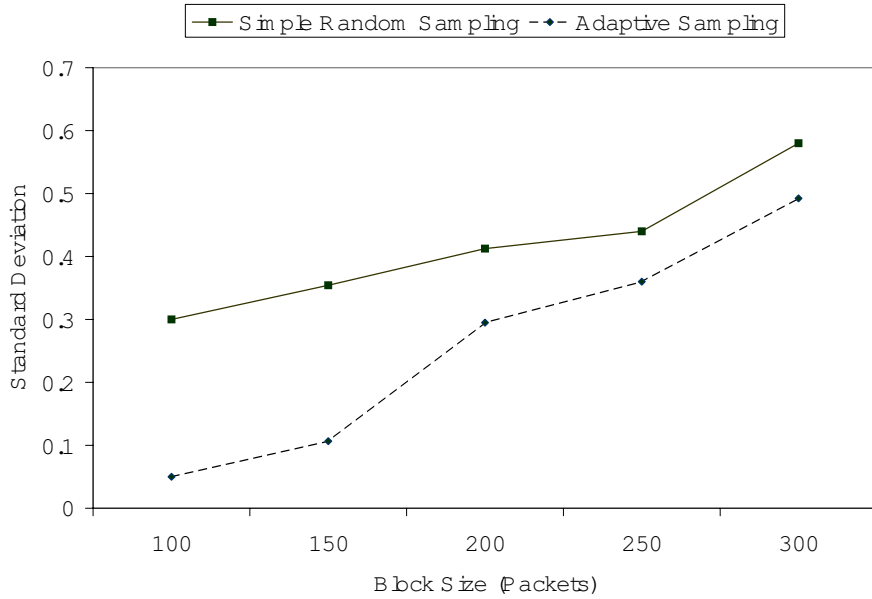


Fig. 5. Standard deviation of packet delay.

Packet delay is an important criterion for detecting DoS attacks, especially attacks that focus on degrading the quality of service in IP networks [35]. The results show that over different block sizes, the proposed adaptive scheme has a lower standard deviation when compared with the simple random sampling algorithm. Since standard deviation is directly proportional to the root mean square error criterion, this implies that the proposed algorithm predicts the packet mean delay better than the simple random sampling algorithm while reducing the volume of traffic.

In figure 6 and figure 7 we show the average sampling error for the Hurst parameter and the sample mean respectively. As one can see from figure 6, the random sampling algorithm resulted in higher average percent error for the Hurst parameter when compared to adaptive sampling. This could be the result of missing data spread out over a number of sampling intervals. In figure 7, the average percentage error for the mean statistic was marginally higher for our sampling algorithm when compared with the simple random sampling algorithm, albeit the difference

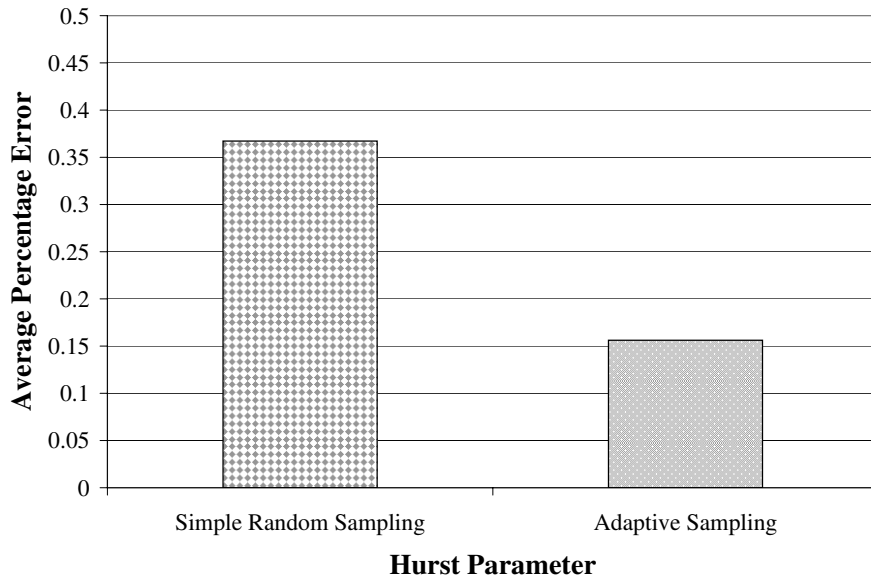


Fig. 6. Average percentage error for the Hurst parameter.

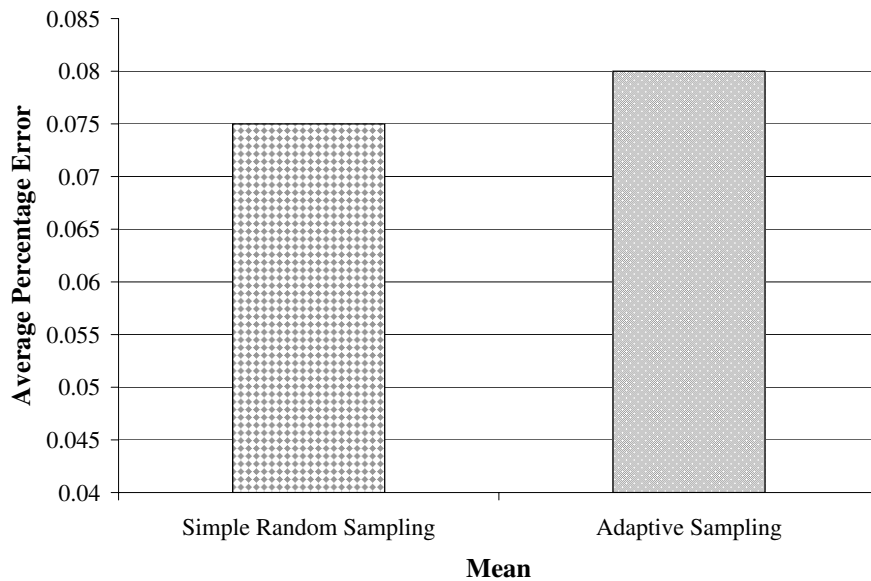


Fig. 7. Average percentage error for the mean statistic.

was insignificant. One possible reason for this marginal difference is the inherent adaptive nature of our sampling algorithm—i.e., the proposed sampling algorithm is more likely to miss short bursts of high network activity in periods that typically have low network traffic. The simple random sampling scheme would be less likely to have the same problem.

5.2 Evaluation of the Clustering Algorithm

To evaluate quality of results and performance we tested our clustering algorithm with the DARPA/MIT Lincoln labs intrusion detection dataset. POTION is compared against the k -means clustering algorithm. The k -means clustering algorithm assigns multivariate observations to a pre-determined number of groups (k). Each of the k groups consists of m_k data items and a group centroid (y_k). In the beginning, the algorithm randomly selects k initial clusters from the dataset as the group centroids. Then the k -means algorithm calculates the arithmetic mean of each cluster formed in the dataset. In other words, the objective it tries to achieve is to minimize total intra-cluster variance

$$\sum_{i=1}^k \sum_{j \in S_i} |x_j - \mu_i|^2,$$

where there are k clusters $S_i = 1, 2, \dots, k$ and μ_i is the centroid or mean point of all the points $x_j \in S_i$. The steps outlining the basic k -means algorithm are shown in figure 8.

Input: A set of N data vectors $X = \{x_1, \dots, x_n\}$ and number of clusters k .
Output: A partition of the data vectors given by the cluster identity vector:
 $Y = \{y_1, \dots, y_n\}, y_n \in \{1, \dots, k\}$
Algorithm:

- (a) **Initialize:** Arbitrarily assign classes to datapoints.
- (b) **Iterate for p iterations:** At step k
 - (i) For each datapoint in the dataset, assign that datapoint to a class such that the distance from this datapoint to the center of that class is minimized.
 - (ii) For each class, recalculate the means of the class based on the datapoints that belong to that class.

Fig. 8. k -means algorithm for clustering.

The drawback of the k -means algorithm is that the quality of the local optimum strongly depends on the initial guess (either the centers or the assignments). If we start with a wild guess for the centers it would be fairly unlikely that the process would converge to a good local minimum. Another disadvantage of the k -means algorithm is that the answers it gives are not unique. Therefore it is difficult to compare the quality of the clusters produced.

5.2.1 Experimental Setup

Since the end goal is to accurately detect intrusions using the anomaly detection paradigm, as a first step we processed the “incoming” data as described in Section 4.2. In addition to cluster sizes, inter-cluster distances are used in our clustering-based detection process. For the k -means algorithm, the inter-cluster distance is defined as the Euclidean distance between two cluster centroids. For the

EM algorithm, it is defined as the Mahalanobis distance between two cluster centroids. The EM and the k -means algorithms were written and compiled in Matlab and simulated in an offline manner. The processed datasets were stored as plain text files. All experiments were run on a Intel Pentium computer running at 1.7 GHz, having 100 gigabytes of disk space and 1 gigabyte on main memory.

5.2.2 Metrics for Evaluation

Accuracy is the main quality concern for clustering. In our case one cluster is considered accurate if there is no more than ϵ error in its centroid/mean. If μ_j is the correct mean of cluster j and M_j is the value estimated by the algorithm, then the cluster j is considered accurate if

$$\frac{|\mu_j - M_j|}{\mu_j} \leq e,$$

where e is the estimation error and in this paper has been assumed to be 0.1. That is, we will consider a cluster to be accurate if it differs by no more than 10% of its “true” mean. The accuracy of clustering of POTION was also estimated by varying the percentage of missing audit data.

Since the end goal of employing the clustering algorithm is to use it for outlier detection based intrusion detection, we also evaluated the performance of the anomaly detection algorithm when the input data was clustered using POTION and k -means respectively. We use the Receiver-Operating Characteristic (ROC) curves in both cases to evaluate the efficiency and accuracy of the anomaly detection process. The ROC curve approach analyzes the tradeoff between false alarm and detection rates for detection systems. ROC analysis was originally developed in the field of signal detection. ROC curves for intrusion detection indicate how the detection rate changes as internal thresholds are varied to generate more or fewer false alarms to tradeoff detection accuracy against analyst workload. Measuring the detection rate alone only indicates the types of attacks that an intrusion detection system may detect. Such measurements do not indicate the human workload required to analyze false alarms generated by normal background traffic. False alarm rates in the range of a few hundred per day make a system almost unusable, even with high detection accuracy, because putative detections or alerts generated can not be believed and security analysts must spend many hours each day dismissing false alarms. Low false alarm rates combined with high detection rates, however, mean that the putative detection outputs can be trusted and that the human labor required to confirm detections is minimized.

5.2.3 Experimental Results

During the *training phase* we set up a baseline of normal activity. This is achieved by feeding the flow records and data summaries, obtained after processing the *normal tcpdump* data, as inputs to the clustering algorithm. The cluster boundaries that are obtained at the conclusion of this phase gives us the “boundary” of a normal cluster.

The cluster centers, covariance matrices, and probabilities for each cluster that are obtained during the *training phase* are used as baseline inputs when SCAN is brought online. This is the *testing phase*. As incoming network data is processed and clustered by SCAN, any points that lie beyond a cluster boundary are termed as *outliers*. Based on the tolerance level of the IDS and the relative distance of the outliers from the cluster centers, the outliers will be labelled as normal points or anomalies. In our simulations, we assume that the tolerance level is zero. This implies that during the *testing phase*, any point that lies outside a cluster boundary is considered an anomaly.

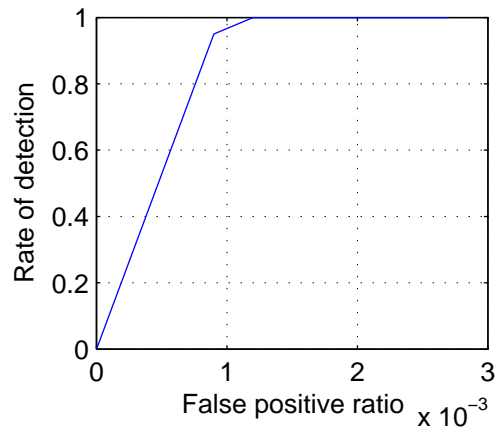


Fig. 9. ROC curve for detection of the SYN Flood attack using POTION.

As mentioned above, we also compared the performance of POTION with the traditional *k*-means clustering algorithm. The *k*-means algorithm was trained using a “normal” dataset to cluster the normal behavior points. For the test data set, the probability of it belonging to the most probable cluster, was computed. If this was below a threshold, the instance was flagged as anomalous. After several runs ($n=10$), the outlier threshold for probability of belonging to a cluster was selected as 0.7. Figure 9 and figure 10 show the performance of the the traditional *k*-Means clustering algorithm against POTION while detecting the SYN Flood attack [36] ⁹.

⁹ A SYN flood attack or a neptune attack is a network-based denial of service attack that is perpetrated by sending large amounts of TCP SYN packets to a target host from a spoofed source address. The spoofed addresses are nonexistent on network. The victim’s server then responds with a SYN/ACK back to the nonexistent address. Because no address receives this SYN/ACK, the victim just waits for the ACK from the client. The ACK never

The ROC curves show that POTION outperforms the traditional k -Means clustering algorithm.

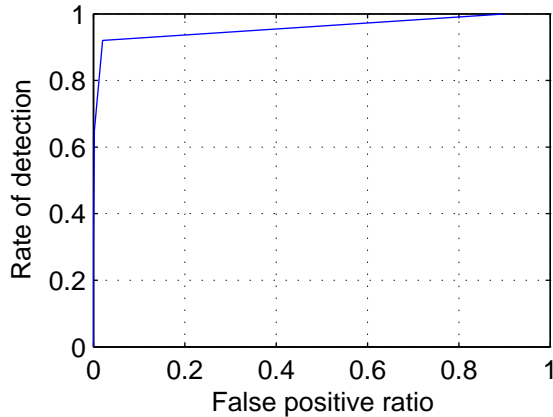


Fig. 10. ROC curve for detection of the SYN Flood attack using the k -means clustering algorithm.

From figure 10 we see that the k -means clustering approach incurs a large number of false positives when all the features (figure 4) were used for clustering. This is in line with the known k -means clustering algorithm’s weakness of poor performance in the presence of a large number of features in the dataset.

5.3 Evaluation of the Anomaly Detection Algorithm

5.3.1 Experimental Setup

In the last stage of the evaluation process, we evaluated the performance of the anomalous flow detection module in SCAN under two scenarios: when *complete* audit data is available and when only *partial* audit data is available. In the later case, to generate partial audit data, we assumed that the missing data are missing completely at random (MCAR). When we say that data are missing completely at random, we mean that the probability that an observation (X_i) is missing is unrelated to the value of X_i or to the value of any other variables. To introduce $N\%$ missing features to a data set of D records, each of which has F features, we select randomly $\lfloor (N \times D \times F) / 100 \rfloor$ records. For each selected record, we delete a randomly chosen feature to obtain a MCAR dataset.

We however contend that our scheme will work as well, in cases where we have bursty missing data. Such scenarios might occur when, for example, a network el-

arrives, and the victim’s server eventually times out. If the attacker sends SYN requests often enough, the victim’s available resources for setting up a connection will be consumed waiting for these bogus ACK’s. These resources are usually low in number, so relatively few bogus SYN requests can create a DoS event.

ement is temporarily overwhelmed with the sudden surge in flow of network traffic and stops collecting data. Because, SCAN does not differentiate between bursty missing data and MCAR datasets, in such a case, it would determine the probable values of the network parameters based on the past information and use the thus calculated values to detect intrusions.

5.3.2 Metrics for Evaluation

As in Section 5.2, we use the Receiver-Operating Characteristic (ROC) curves in both cases to evaluate the efficiency and accuracy of the anomaly detection process. We use a practical assumption about the intrusion data — the number of normal instances is much larger than that of attack instances. This is usually true in reality. However, unlike in Portnoy et al. [37], we do not make the strict hypothetical requirement that the percentage of attacks has to be less than a certain threshold (e.g., 1.5%).

5.3.3 Experimental Results

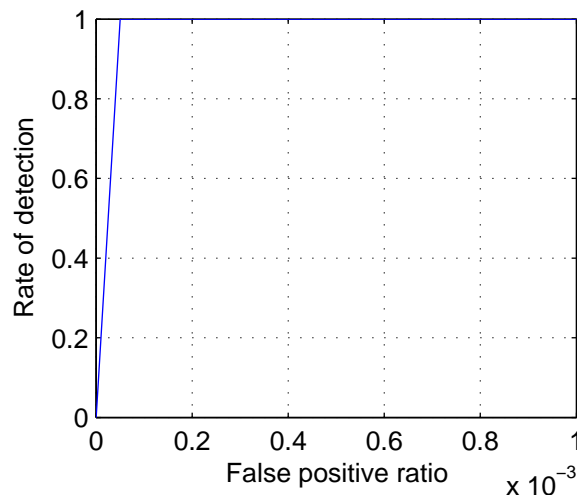


Fig. 11. ROC curve for detection of the SSH Process Table attack using complete audit data.

The performance of SCAN at detecting a SSH Process Table attack when complete audit data is available is shown in figure 11. The attack is similar to the Process Table attack in that the goal of the attacker is to cause the *SSHD* daemon to spawn the maximum number of processes that the operating system will allow. In figure 12, the performance of SCAN at detecting a SYN flood attack is evaluated. In a SYN flood attack, an attacker makes a large number of half-open connections to a given port of a server during a short period of time. This causes the data structure in the ‘tcpd’ daemon in the server to overflow. Due to the buffer overflow, the system will be unable to accept any new incoming connections till the queue empties.

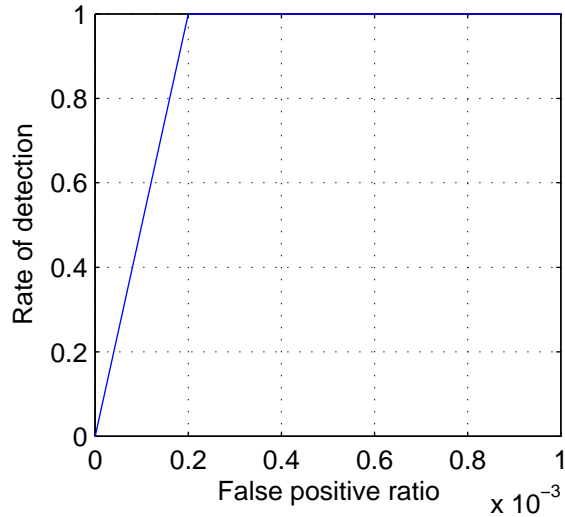


Fig. 12. ROC curve for SYN Flood (Neptune) attack detection using complete audit data.

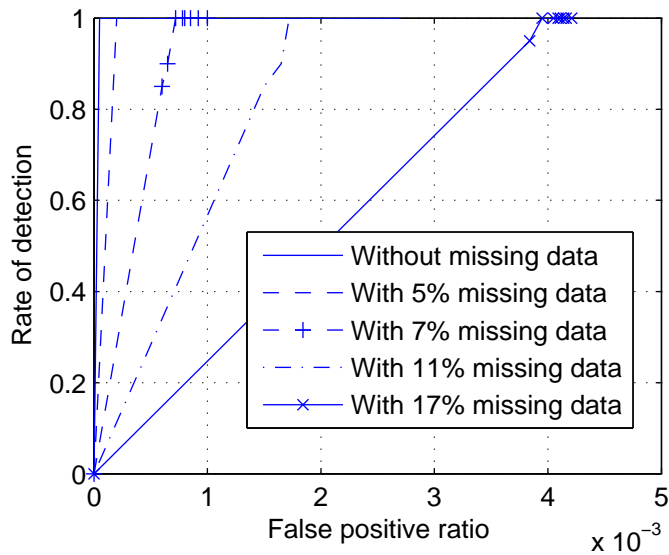


Fig. 13. ROC curve for SYN Flood attack detection using varying degrees of missing data (via random sampling).

In figure 13, we plot SCAN's ROC curves for the case in which complete audit data is available, and in cases when 5%, 7%, 11% and 17% of the audit data are missing. The simple random sampling algorithm was used to sample the network traffic data.

Figure 14 shows the accuracy of clustering¹⁰ versus percentage of missing data. It can be seen that even with 10% missing data, the accuracy of clustering is in the high eighties. As expected, we see that the accuracy of clustering as well as the

¹⁰ Accuracy of clustering is measured as the average percentage of correctly classified data over multiple runs.

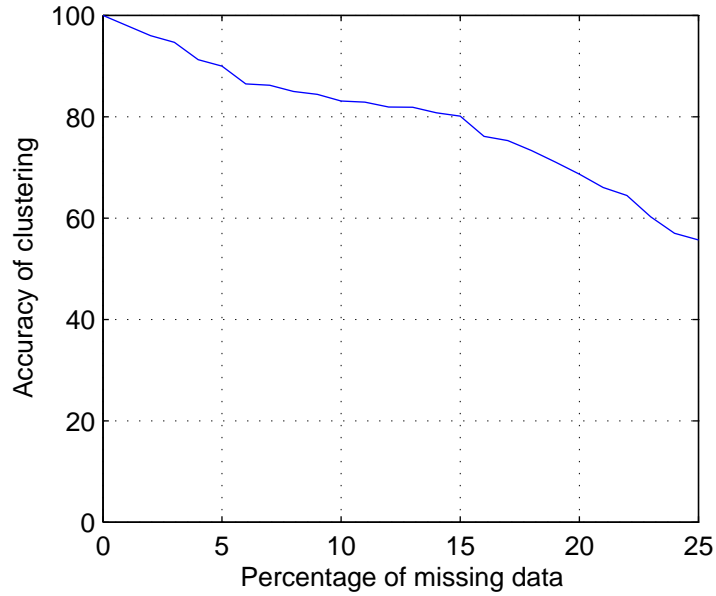


Fig. 14. Accuracy of clustering with POTION vs. percentage of missing data.

performance of the anomaly detection scheme degrades with increasing percentage of missing data.

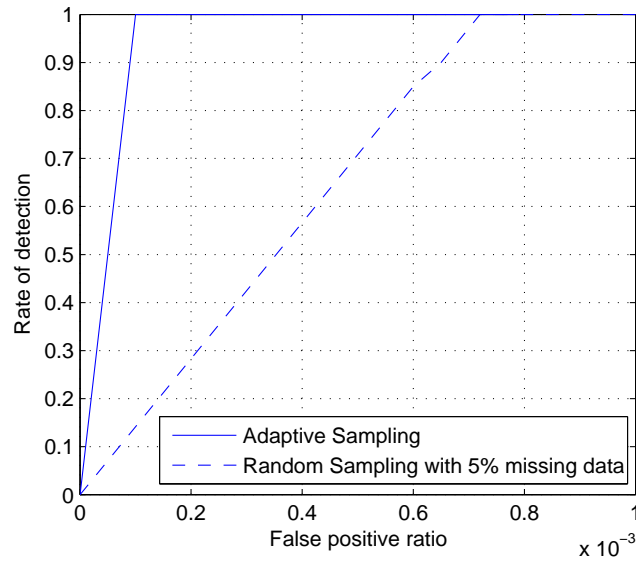


Fig. 15. Comparison of adaptive sampling and random sampling in the detection of the SYN Flood attack.

Lastly, in figure 15, we compared the performance of the adaptive sampling technique and the random sampling technique for detecting the SYN Flood attack when clustering was done using POTION in both cases. As expected, the adaptive sampling technique showed superior performance.

6 Conclusion

In this paper, we have presented an anomaly detection scheme—called SCAN—that has the ability to detect network based denial-of-service attacks in gigabit-speed networks with a relatively high degree of accuracy in the absence of complete audit data. By employing an intelligent sampling scheme, SCAN reduces the computational complexity [38] by reducing the volume of audit data that is processed without losing the intrinsic characteristics of the network traffic. In addition, SCAN also employs an improved Expectation-Maximization algorithm based clustering technique to impute the missing values and further increase the accuracy of anomaly detection.

References

- [1] B. Yocom, R. Birdsall, and D. Poletti-Metzel, “Gigabit intrusion detection systems.” <http://www.nwfusion.com/reviews/2002/1104rev.html>, 2002.
- [2] G. Cormode and S. Muthukrishnan, “Whats new: Finding significant differences in network data streams,” in *IEEE INFOCOMM 2004: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1534–1545, IEEE Press, 2004.
- [3] R. Sekar, Y. Guang, S. Verma, and T. Shanbhag, “A high-performance network intrusion detection system,” in *ACM Conference on Computer and Communications Security*, pp. 8–17, 1999.
- [4] E. Eskin, “Anomaly detection over noisy data using learned probability distributions,” in *17th International Conference on Machine Learning*, pp. 255–262, 2000.
- [5] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, “A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data,” in *Applications of Data Mining in Computer Security* (D. Barbara and S. Jajodia, eds.), 2002.
- [6] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” in *Journal of the Royal Statistical Society*, vol. 39 of *B*, pp. 1–38, 1977.
- [7] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, and A. Valdes, “Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (nides),” Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, May 1994.
- [8] M. Mahoney and P. K. Chan, “Phad: Packet header anomaly detection for identifying hostile network traffic,” Technical Report CS-2001-2, Computer Science Department, Florida Institute of Technology, 150 W. University Blvd. Melbourne, FL 32901, 2001.

- [9] M. Mahoney and P. K. Chan, "Learning models of network traffic for detecting novel attacks," Technical Report CS-2002-8, Department of Computer Science, Florida Institute of Technology, 150 W. University Blvd. Melbourne, FL 32901, 2002.
- [10] M. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *SIGKDD*, July 2002.
- [11] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," in *15th International Conference on Data Engineering*, (Stanford University, CA, USA), pp. 512–521, IEEE Computer Society, March 1999.
- [12] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 392–403, Morgan Kaufmann Publishers Inc., 1998.
- [13] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.
- [14] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," in *7th USENIX Security Symposium*, (San Antonio, TX), 1998.
- [15] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real time data mining-based intrusion detection," in *Second DARPA Information Survivability Conference and Exposition*, pp. 85–100, 2001.
- [16] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *IEEE Symposium on Security and Privacy*, (Washington DC, USA), pp. 130–143, IEEE Computer Society, 2001.
- [17] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful intrusion detection for high-speed networks.," in *IEEE Symposium on Research on Security and Privacy*, pp. 285–294, May 2002.
- [18] B. Hutchings, R. Franklin, and D. Carver, "Assisting network intrusion detection with reconfigurable hardware.," in *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 111–120, April 2002.
- [19] ISS, *BlackICE Sentry Gigabit*. Internet Security Solutions, 2001.
- [20] CISCO, *CISCO Intrusion Detection System*. Cisco Systems, 2001.
- [21] T. Networks, "Toplayer networks." <http://www.toplayer.com/>, 2005.
- [22] W. E. Leland, M. S. Taqq, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," in *ACM SIGCOMM* (D. P. Sidhu, ed.), (San Francisco, California), pp. 183–193, 1993.
- [23] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

- [24] M. Li, W. Jia, and W. Zhao., “Decision analysis of network based intrusion detection systems for denial-of-service attacks.,” in *Proceedings of the IEEE Conferences on Info-tech and Info-net*, vol. 5, Dept. of Computer Sci., City Univ. of Hong Kong, China, IEEE, October 2001.
- [25] P. Owezarski, “On the impact of DoS attacks on internet traffic characteristics and QoS,” in *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pp. 269–274, LAAS-CNRS, Toulouse, France, IEEE, October 2005.
- [26] H. E. Hurst, “Methods of using long-term storage in reservoirs,” in *In Proceedings of the Institution of Civil Engineers*, no. Part 1, pp. 519–577, 1955.
- [27] R. Neal and G. Hinton, “A view of the em algorithm that justifies incremental, sparse and other variants,” *Learning in graphical models*, pp. 355–368, 1999.
- [28] P. Bradley, U. Fayyad, and C. Reina, “Scaling em (expectation- maximization) clustering to large databases,” Technical Report MSR-TR-98-35, Microsoft Research, 1998.
- [29] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: an efficient data clustering method for very large databases,” in *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 103–114, ACM Press, 1996.
- [30] C. Ordonez and E. Omiecinski, “Frem: fast and robust em clustering for large data sets,” in *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, (New York, NY, USA), pp. 590–599, ACM Press, 2002.
- [31] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.
- [32] B. Bloom, “Space/time tradeoffs in hash coding with allowable errors.,” in *Communications of the ACM*, 1970.
- [33] M. L. Laboratory, “Darpa intrusion detection evaluation data set.” <http://www.ll.mit.edu/>.
- [34] WIDE, “The widely integrated distributed environment project.” <http://tracer.csl.sony.co.jp/mawi/>.
- [35] E. Fulp, Z. Fu, D. S. Reeves, S. F. Wu, and X. Zhang, “Preventing denial of service attacks on quality of service,” in *DISCEX '01: Proceedings of the DARPA Information Survivability Conference and Exposition II*, vol. 2, pp. 159–172, IEEE Press, June 2001.
- [36] CERT, “Cert advisory ca-1996-21 tcp syn flooding and ip spoofing attacks.” <http://www.cert.org/advisories/CA-1996-21.html>, September 1996.
- [37] L. Portnoy, E. Eskin, and S. J. Stolfo, “Intrusion detection with unlabeled data using clustering,” in *ACM Workshop on Data Mining Applied to Security*, 2001.

- [38] K. Claffy, G. Polyzos, and H. Braum, "Application of sampling methodologies to network traffic characterization.," in *Computer Communication Review*, vol. 4, pp. 194–203, 1993.