A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection[†]

Aleksandar Lazarevic*, Levent Ertoz*, Vipin Kumar*, Aysel Ozgur*, Jaideep Srivastava*

Abstract

Intrusion detection corresponds to a suite of techniques that are used to identify attacks against computers and network infrastructures. Anomaly detection is a key element of intrusion detection in which perturbations of normal behavior suggest the presence of intentionally or unintentionally induced attacks, faults, defects, etc. This paper focuses on a detailed comparative study of several anomaly detection schemes for identifying different network intrusions. Several existing supervised and unsupervised anomaly detection schemes and their variations are evaluated on the DARPA 1998 data set of network connections [9] as well as on real network data using existing standard evaluation techniques as well as using several specific metrics that are appropriate when detecting attacks that involve a large number of connections. Our experimental results indicate that some anomaly detection schemes appear very promising when detecting novel intrusions in both DARPA'98 data and real network data.

1 Introduction

As the cost of the information processing and Internet accessibility falls, more and more organizations are becoming vulnerable to a wide variety of cyber threats. According to a recent survey by CERT/CC [1], the rate of cyber attacks has been more than doubling every year in recent times. Therefore, it has become increasingly important to make our information systems, especially those used for critical functions in the military and commercial sectors, resistant to and tolerant of such attacks. The most widely deployed methods for detecting cyber terrorist attacks and protecting against cyber terrorism employ signature-based detection techniques. Such methods can only detect previously known attacks that have a corresponding signature, since the signature database has to be manually revised for each new type of attack that is discovered. These limitations have led to an increasing interest in intrusion detection techniques based on data mining [2, 3, 4, 5, 6].

Data mining based intrusion detection techniques generally fall into one of two categories; misuse detection and anomaly detection. In misuse detection, each instance in a data set is labeled as 'normal' or 'intrusive' and a learning algorithm is trained over the labeled data. These techniques are able to automatically retrain intrusion detection models on different input data that include new types of attacks, as long as they have been labeled appropriately. Research in misuse detection has focused mainly on classification of network intrusions using various standard data mining algorithms [2, 4, 5, 6], rare class predictive models, association rules [2, 5] and cost sensitive modeling. Unlike signature-based intrusion detection systems, models of misuse are created automatically, and can be more sophisticated and precise than manually created signatures. A key advantage of misuse detection techniques is their high degree of accuracy in detecting known attacks and their variations. Their obvious drawback is the inability to detect attacks whose instances have not yet been observed.

Anomaly detection approaches, on the other hand, build models of normal data and detect deviations from the normal model in observed data. Anomaly detection applied to intrusion detection and computer security has been an active area of research since it was originally proposed by Denning [7]. Anomaly detection algorithms have the advantage that they can detect new types of intrusions as deviations from normal usage [7, 8]. In this problem, given a set of normal data to train from, and given a new piece of test data, the goal of the intrusion detection algorithm is to determine whether the test data belong to "normal" or to an anomalous behavior. However, anomaly detection schemes suffer from a high rate of false alarms. This occurs primarily because previously unseen (yet legitimate) system behaviors are also recognized as anomalies, and hence flagged as potential intrusions.

This paper focuses on a detailed comparative study of several anomaly detection schemes for identifying different network intrusions. Several existing supervised and unsupervised anomaly detection schemes and their variations are evaluated on the DARPA 1998 data set of network connections [9] as well as on real network data using existing standard evaluation techniques as well as using several specific metrics that are appropriate when detecting attacks that involve a large number of connections. Our experimental results indicate that some anomaly detection schemes ap-

* Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455, USA

† The authors are grateful to Richard Lippmann and Daniel Barbara for providing data sets and their useful comments. This work was partially supported by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. The content of the work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute

pear very promising when detecting novel intrusions in both DARPA'98 data and real network data.

2 Evaluation of Intrusion Detection Systems

As interest in intrusion detection has grown, the topic of evaluation of intrusion detection systems (IDS) has also received great attention [9, 10, 11, 12]. Evaluating intrusion detection systems is a difficult task due to several reasons. First, it is problematic to get high-quality data for performing the evaluation due to privacy and competitive issues, since many organizations are not willing to share their data with other institutions. Second, even if real life data were available, labeling network connections as normal or intrusive requires enormous amount of time for many human experts. Third, the constant change of the network traffic can not only introduces new types of intrusions but can also change the aspects of the "normal" behavior, thus making construction of useful benchmarks even more difficult. Finally, when measuring the performance of an IDS, there is a need to measure not only detection rate (i.e. how many attacks we detected correctly), but also the false alarm rate (i.e. how many of normal connections we incorrectly detected as attacks) as well as the cost of misclassification. The evaluation is further complicated by the fact that some of the attacks (e.g. denial of service (DoS), probing) may use hundreds of network packets or connections, while on the other hand attacks like U2R (user to root) and R2L (remote to local) typically use only one or a few connections.

Standard metrics that were developed for evaluating network intrusions usually correspond to detection rate as well as false alarm rate (Table 1). Detection rate is computed as the ratio between the number of correctly detected attacks and the total number of attacks, while false alarm (false positive) rate is computed as the ratio between the number of normal connections that are incorrectly misclassified as attacks (false alarms in Table 1) and the total number of normal connections.

Table 1. Standard metrics for evaluations of single-connection intrusions (attacks)

Standard metrics		Predicted connection label	
		Normal	Intrusions (Attacks)
Actual connection label	Normal	True Negative	False Alarm
	Intrusions (Attacks)	False Negative	Correctly detected attacks

There are generally two types of attacks in network intrusion detection: the attacks that involve single connections and the attacks that involve multiple connections (bursts of connections). The standard metrics treat all types of attacks similarly thus failing to provide sufficiently generic and systematic evaluation for the attacks that involve many network connections (bursty attacks). In particular, they do not capture information about the number of network connections associated with an attack that have been correctly detected. Therefore, depending on the type of the attack, two types of analysis may be applied; multi-connection attack analysis for bursty attacks and the single-connection attack analysis for single connection attacks (Figure 1). However, the first step for both analysis types corresponds to computing the score value for each network connection. The score value represents the likelihood that particular network connection is associated with an intrusion (Figure 1).

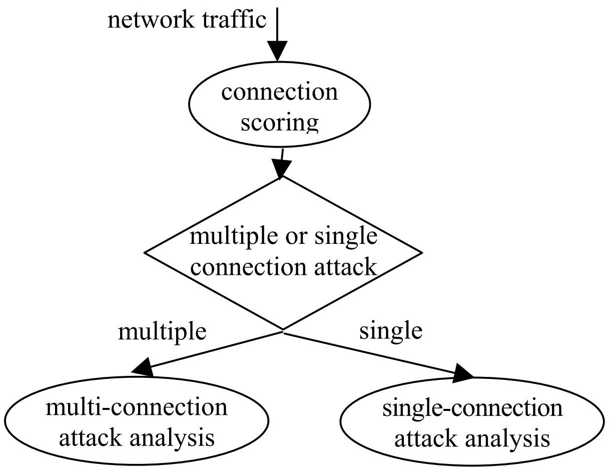

network traffic
connection scoring
multiple or single connection attack
multiple
single
multi-connection attack analysis
single-connection attack analysis

Figure 1. Multi-step approach for evaluation intrusions in the network traffic

Assume that for a given network traffic in some time interval, each connection is assigned a score value, represented as a vertical line (Figure 2). The dashed line in Figure 2 represents the real attack curve that is zero for non-intrusive (normal) network connections and one for intrusive connections. The full line in Figure 2 corresponds to the predicted attack curve, and for each connection it is equal to its assigned score. These two curves allow us to compute the error for every connection as the difference between the real connection value (1 for connections associated with attacks and 0 for normal connections) and the assigned score to the connection, and to further derive additional metrics.

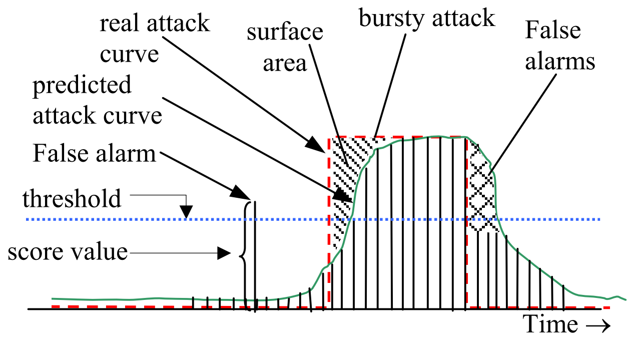
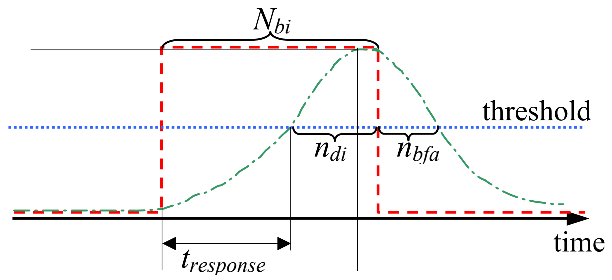


Figure 2. Assigning scores in network intrusion detection

The multi-step approach shown in Figure 1 utilizes computed errors for each connection in order to derive additional evaluation metrics. The first derived metric corresponds to the surface areas between the real attack curve and the predicted attack curve (surfaces denoted as \\\ in Figure 2). The smaller the surface between these two attack curves, the better the intrusion detection algorithm. However, the surface area itself is not sufficient to capture many relevant aspects of intrusion detection algorithms (e.g. how many connections are associated with the attack, how fast the intrusion detection algorithm is, etc.). Therefore, additional metrics may be used in order to support the basic metric of surface area under the attack curve. Assume that the total number of network connections in considered data set is N. The number N is equal to the sum of the total number of normal network connections (N_n) and the total number of network connections that are associated with the intrusions (N_i). The number (n_{fa}) corresponds to the number of the non-intrusive (normal) network connections (n_{fa}) that have the score higher than prespecified threshold (dotted line in Figure 3) and therefore misclassified as intrusive ones. Now, the additional metrics may be defined as follows:

1. *Burst detection rate* (*bdr*) is defined for each burst and it represents the ratio between the total number of intrusive network connections n_{di} that have the score higher than prespecified threshold within the bursty attack (dotted line in Figure 3) and the total number of intrusive network connections within attack intervals (N_{bi}) (Figure 3). $bdr = n_{di} / N_{bi}$., where $\sum_{all\_bursts} N_{bi} = N_i$.

   Similar metric was used in DARPA 1998 evaluation [9].



Metric	Definition
bdr	*burst detection rate* = n_{di}/N_{bi}
n_{di}	number of intrusive connections that have score value higher than threshold
n_{bfa}	number of normal connections that follow attack and that are misclassified as intrusive
$t_{response}$	*response time* – time to reach the prespecified threshold

Figure 3. The additional metrics relevant for IDS evaluation

2. *Response time* represents the time elapsed from the beginning of the attack till the moment when the first network connection has the score value higher than prespecified threshold ($t_{response}$ in Figure 3). Similar metric was used in DARPA 1999 evaluation [11] where 60s time interval was allowed to detect the bursty attack.

3. Anomaly Detection Techniques

3.1. Related Work. Most research in supervised anomaly detection can be considered as performing generative modeling. These approaches attempt to build some kind of a model over the normal data and then check to see how well new data fits into that model. An approach for modeling normal sequences using look ahead pairs and contiguous sequences is presented in [13]. A statistical method for ranking each sequence by comparing how often the sequence is known to occur in normal traces with how often it is expected to occur in intrusions is presented in [14]. One approach uses a prediction model obtained by training decision trees over normal data [2], while others use neural networks to obtain the model [15] or non-stationary models [16] to detect novel attacks. Lane and Brodley [17] performed anomaly detection on unlabeled data by looking at user profiles and comparing the activity during an intrusion to the activity during normal use. Similar approach of creating user profiles using semi-incremental techniques was also used in [18]. Barbara used pseudo-Bayes estimators to enhance detection of novel attacks while reducing the false alarm rate as much as possible [5]. A technique developed at SRI in the EMERALD system [8] uses historical records as its normal training data. It then compares distributions of new data to the distributions obtained from those historical records and differences between the distributions indicate an intrusion. Recent works such as [19] and [20] estimate parameters of a probabilistic model over the normal data and compute how well new data fits into the model.

In this paper our focus is on several outlier detection algorithms as well as on unsupervised support vector machine algorithms for detecting network intrusions.

3.2. Outlier Detection Schemes for Anomaly Detection. Most anomaly detection algorithms require a set of purely normal data to train the model, and they implicitly assume that anomalies can be treated as patterns not observed before. Since an outlier may be defined as a data point which is very different from the rest of the data, based on some measure, we employ several outlier detection schemes in order to see how efficiently these schemes may deal with the problem of anomaly detection.

The statistics community has studied the concept of outliers quite extensively [21]. In these techniques, the data points are modeled using a stochastic distribution, and points are determined to be outliers depending upon their relationship with this model. However, with increasing dimensionality, it becomes increasingly difficult and inaccurate to estimate the multidimensional distributions of the data points [22]. However, recent outlier detection algorithms that we utilize in this study are based on computing the full dimensional distances of the points from one another [23, 24] as well as on computing the densities of local neighborhoods [25].

3.2.1. *Mining Outliers Using Distance to the k-th Nearest Neighbor* [24]. This approach is based on computing the Euclidean distance of the *k*-th nearest neighbor from the point *O*. For a given *k* and a point *O*, $D^k(O)$ denotes the distance from the point *O* to its *k*-th nearest neighbor. Therefore, the distance $D^k(O)$ may be considered as a measure of the outlierness of the example *O*. For instance, points with larger values $D^k(O)$ for have more sparse neighborhoods and they typically represent stronger outliers than points belonging to dense clusters that usually tend to have lower values for $D^k(O)$. Since generally user is interested in top *n* outliers, this approach defines an outlier as follows: Given a *k* and *n*, a point *O* is an outlier if the distance to its *k*-th nearest neighbor is smaller than the corresponding value for no more than (*n*-1) other points. In other words, the top *n* outliers with the maximum $D^k(O)$ values are considered as outliers.

3.2.2. *Nearest Neighbor (NN) Approach*. This method is a slight modification of the outlier detection scheme presented in previous section 3.2.1., when $k = 1$. We specify an "outlier threshold" that will serve to determine whether the point is an outlier or not. The threshold is based only on the training data and it is set to 2%. In order to compute the threshold, for all data points from training data (e.g. "normal behavior" data) distances to their nearest neighbors are computed and then sorted. All test data points that have distances to their nearest neighbors greater than the threshold are detected as outliers.

3.2.3. *Mahalanobis-distance Based Outlier Detection*. Since the training data corresponds to "normal behavior", it is straightforward to compute the mean and the standard deviation of the "normal" data. The Mahalanobis distance [ref] between the particular point *p* and the mean μ of the normal data is computed as:

$$d_M = \sqrt{(p-\mu)^T \cdot \Sigma^{-1} \cdot (p-\mu)},$$

where the Σ is the covariance matrix of the "normal" data. Similarly to the previous approach, the threshold is computed according to the most distant points from the mean of the "normal" data and it is set to be 2% of total number of points. All test data points that have distances to the mean of the training "normal" data greater than the threshold are detected as outliers.

Computing distances using standard Euclidean distance metric is not always beneficial, especially when the data has a distribution similar to that presented in Figure 4. It is obvious that examples p_1 and p_2 do not have the same distance to the mean of the distribution when the distances are computed using standard Euclidean metric and Mahalanobis metric. When using standard Euclidean metric, the distance between p_2 and its nearest neighbor is greater than the distance from p_1 to its nearest neighbor. However, when using the Mahalanobis distance metric, these two distances are the same. It is apparent that in these scenarios, Mahalanobis based approach is beneficial compared to the Euclidean metric.

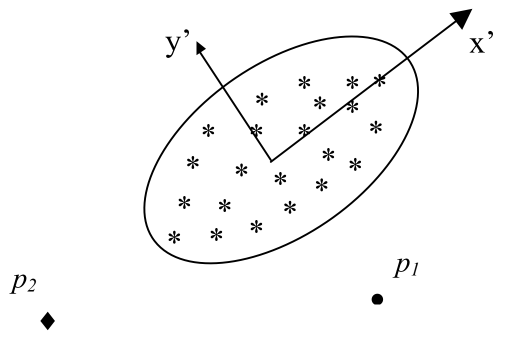


Figure 4. Advantage of Mahalanobis-distance based approach when computing distances.

3.2.4. *Density Based Local Outliers (LOF approach)*. The main idea of this method [25] is to assign to each data example a degree of being outlier. This degree is called the *local outlier factor (LOF)* of a data example. The algorithm for computing the *LOF*s for all data examples has several steps:

1. For each data example *O* compute *k*-distance (the distance to the *k*-th nearest neighbor) and *k*-distance neighborhood (all points in a *k*-distance sphere).
2. Compute reachability distance for each data example *O* with respect to data example *p* as: *reach-dist(O,p)* = max{*k-distance(p), d(O,p)*}, where *d(O,p)* is distance from data example *O* to data example *p*.
3. Compute local reachability density of data example *O* as inverse of the average reachabaility distance based on the *MinPts* (minimum number of data examples) nearest neighbors of data example *O*.
4. Compute *LOF* of data example *O* as average of the ratios of the local reachability density of data example *O* and local reachability density of *O*'s *MinPts* nearest neighbors.

To illustrate advantages of the LOF approach, consider a simple two-dimensional data set given in Figure 5. It is apparent that there is much larger number of examples in the cluster C_1 than in the cluster C_2, and that the density of the cluster C_2 is significantly higher that the density of the cluster C_1. Due to the low density of the cluster C_1 it is apparent that for every example *q* inside the cluster C_1, the distance between the example *q* and its nearest neighbor is greater than the distance between the example p_2 and the nearest neighbor from the cluster C_2, and the example p_2 will not be considered as outlier. Therefore, the simple nearest neighbor approaches based on computing the distances fail in these scenarios. However, the example p_1 may be detected as outlier using only the distances to the nearest neighbor. On the other side, *LOF* is able to capture both outliers (p_1 and p_2) due to the fact that it considers the density around the points.

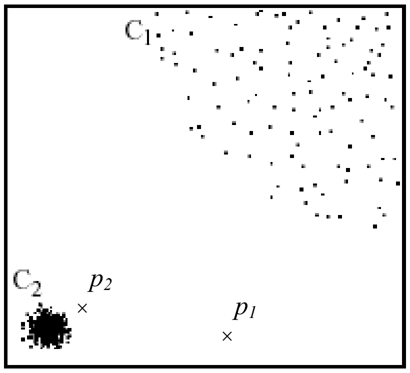


Figure 5. Advantages of the *LOF approach*

3.3. Unsupervised Support Vector Machines. Unlike standard supervised support vector machines (SVMs) that require labeled training data to create their classification rule, in [27], the SVM algorithm was adapted into unsupervised learning algorithm. This unsupervised modification does not require training data to be labeled to determine a decision surface. Whereas the supervised SVM algorithm tries to maximally separate two classes of data in feature space by a hyperplane, the unsupervised algorithm attempts to separate the entire set of training data from the origin, i.e. to find a small region where most of the data lies and label data points in this region as one class. Points in other regions are labeled as another class.

By using different values for SVM parameters (variance parameter of radial basis functions (RBFs), expected outlier rate), the models with different complexity may be built. For RBF kernels with smaller variance, the number of support vectors is larger and the decision boundaries are more complex, thus resulting in very high detection rate but very high false alarm rate too. On the other hand, by considering RBF kernels with larger variance, the number of support vectors decreases while the boundary regions become more general, which results in lower detection rate but lower false alarm rate as well.

4. Experiments

We applied the proposed anomaly detection schemes to 1998 DARPA Intrusion Detection Evaluation Data [9] as well as to the real network data from the University of Minnesota.

The DARPA'98 data contains two types: training data and test data. The training data consists of 7 weeks of network-based attacks inserted in the normal background data. Attacks in training data are labeled. The test data contained 2 weeks of network-based attacks and normal background data. 7 weeks of data resulted in about 5 million connection records. The data contains four main categories of attacks:

- DoS (Denial of Service), for example, ping-of-death, teardrop, smurf, SYN flood, etc.,
- R2L, unauthorized access from a remote machine, for example, guessing password,
- U2R, unauthorized access to local superuser privileges by a local unprivileged user, for example, various buffer overflow attacks,
- PROBING, surveillance and probing, for example, port-scan, ping-sweep, etc.

Although DARPA'98 evaluation represents a significant advance in the field of intrusion detection, there are many unresolved issues associated with its design and execution. In his critique of DARPA evaluation, McHugh [28] questioned a number of their results, starting from usage of synthetic simulated data for the background (normal data) and using attacks implemented via scripts and programs collected from a variety of sources. In addition, it is known that the background data contains none of the background noise (packet storms, strange fragments, …) that characterize real data. However, in the lack of better benchmarks, vast amount of the research is based on the experiments performed on this data.

The evaluation of any intrusion detection algorithm on real network data is extremely difficult mainly due to the high cost of obtaining proper labeling of network connections. However, in order to assess the performance of our anomaly detection algorithms in a real setting, we also present the evaluation results of applying our techniques to real network data from the University of Minnesota.

4.1. Feature construction. We used *tcptrace* utility software [29] as the packet filtering tool in order to extract information about packets from TCP connections and to construct new features. The DARPA98 training data includes "list files" that identify the time stamps (start time and duration), service type, source IP address and source port, destination IP address and destination port, as well as the type of each attack. We used this information to map the connection records from "list files" to the connections obtained using *tcptrace* utility software and to correctly label each connection record with "normal" or an attack type. The same technique was used to construct KDDCup'99 data set [2], but this data set did not keep the time information about the attacks. Therefore, we constructed our own features that were similar in nature.

The main reason for this procedure is to associate new constructed features with the connection records from "list files" and to create more informative data set for learning. However, this procedure was applied only to TCP connection records, since *tcptrace* software utility was not able to handle ICMP and UDP packets. For these connection records, in addition to the features provided by DARPA, we used the features that represented the number of packets that flowed from source to destination. The list of the features extracted from "raw tcpdump" data using *tcptrace* software is shown in Table 2.

Table 2. The extracted "content based" features from raw tcpdump data using *tcptrace* software

Feature Name	**Feature description**
num_packets_src_dst	The number of packets flowing from source to destination
num_packets_dst_src	The number of packets flowing from destination to source
num_acks_src_dst	The number of acknowledgement packets flowing from source to destination
num_acks_dst_src	The number of acknowledgement packets flowing from destination to source
num_bytes_src_dst	The number of data bytes flowing from source to destination
num_bytes_dst_src	The number of data bytes flowing from destination to source
num_retransmit_src_dst	The number of retransmitted packets flowing from source to destination
num_retransmit_dst_src	The number of retransmitted packets flowing from destination to source
num_pushed_src_dst	The number of pushed packets flowing from source to destination
num_pushed_dst_src	The number of pushed packets flowing from destination to source
num_SYNs_src_dst	The number of SYN packets flowing from source to destination
num_FINs_src_dst	The number of FIN packets flowing from source to destination
num_SYNs_dst_src	The number of SYN packets flowing from destination to source
num_FINs_dst_src	The number of FIN packets flowing from destination to source
connection_status (*discrete*)	Status of the connection (0 – Completed; 1 - Not completed; 2 – Reset)

Since majority of the DoS and probing attacks may use hundreds of packets or connections, we have constructed time-based features that attempt to capture previous recent connections with similar characteristics. The similar approach was used for constructing features in KDDCup'99 data [2], but our own features examine only the connection records in the past 5 seconds. Table 3 summarizes these derived time-windows features.

Table 3. The extracted "time-based" features

Feature Name	**Feature description**
count_src	Number of connections made by the same source as the current record in the last 5 seconds
count_dest	Number of connections made to the same destination as the current record in the last 5 seconds
count_serv_src	Number of different services from the same source as the current record in the last 5 seconds
count_serv_dest	Number of different services to the same destination as the current record in the last 5 seconds

There are, however, several "slow" probing attacks that scan the hosts (or ports) using a much larger interval than 5 seconds (e.g. one scan per minute or even one scan per hour). As a consequence, these attacks cannot be detected using derived "time based" features. In order to capture these types of the attacks, we also derived "connection based" features that capture similar characteristics of the connection records in the last 100 connections from the same source. These features are described in Table 4.

Although we have use all three groups of features for our experiments, it is well known that constructed features from the data content of the connections are more important when detecting R2L and U2R attack types, while "time-based' and "connection-based" features were more important for detection DoS and probing attack types [2].

Table 4. The extracted "connection-based" features

Feature Name	**Feature description**
count_src1	Number of connections made by the same source as the current record in the last 100 connections
count_dest1	Number of connections made to the same destination as the current record in the last 100 connections
count_serv_src1	Number of connections with the same service made by the same source as the current record in the last 100 connections
count_serv_dst1	Number of connections with the same service made to the same destination as the current record in the last 100 connections

4.2. Experimental Results on DARPA'98 Data. Since the amount of available data is huge (e.g. some days have several million connection records), we sampled sequences of normal connection records in order to create the normal data set that had the same distribution as the original data set of normal connections. We used this normal data set for training our anomaly detection schemes, and then examined how well the attacks may be detected using the proposed schemes.

We used the TCP connections from 5 weeks of training data (499,467 connection records), where we sampled 5,000 data records that correspond to the normal connections, and used them for the training phase. For testing purposes, we used the connections associated with all the attacks from the first 5 weeks of data in order to determine detection rate. Also we considered a random sample of 1,000 connection records that correspond to normal data in order to determine the false alarm rate. It is important to note that this sample used for testing purposes had the same distribution as the original set of normal connections.

First, features from Table 2 are extracted using the *tcptrace* software utility and then connection based and time based features are constructed. The next step involved standard normalization of obtained features and the final step was to identify bursts of attacks in the data. The performance of anomaly detection schemes was tested separately for the attack bursts, mixed bursty attacks and non-bursty attacks.

Experiments were performed using the *nearest neighbor approach* (section 3.2.2), the *Mahalanobis-based approach* (section 3.2.3) the *local outlier factor* (*LOF*) scheme (section 3.2.4) as well as the *unsupervised SVM approach* (section 3.2.5).

In all the experiments, the percentage of the outliers in the training data (allowed false alarm rate) is changed from 1% to 12%. It is interesting to note that the maximum specified false alarm (false positive) rate was also maintained when detecting normal connections from test data.

4.2.1. *Evaluation of Bursty Attacks*. Our experiments were first performed on the attack bursts, and the obtained *burst detection rates* (*bdr*) for all four anomaly detection schemes are reported in Table 5. We consider a burst to be detected if the corresponding *burst detection rate* is greater than 50%. Since we have a total of 19 bursty attacks, overall detection rate in Table 5 was computed using this rule. Experimental results from Table 5 show that the two most successful outlier detection schemes were *nearest neighbor* (*NN*) and *LOF* approaches, where the *NN approach* was able to detect 14 attack bursts and the *LOF approach* was able to detect 13 attack bursts. The *unsupervised SVMs* were only slightly worse than the previous two approaches, showing a great promise in detecting network intrusions. The *Mahalanobis-based approach* was consistently inferior to the *NN approach* and was able to detect only 11 multiple-connection attacks. This poor performance of *Mahalanobis-based scheme* was probably due to the fact that the normal behavior may have several types and cannot be characterized with a single distribution, that we have used in our experiments. In order to alleviate this problem, there is a need to partition the normal behavior into several more similar distributions and identify the anomalies according to the Mahalanobis distances to each of the distributions.

Table 5. *Burst detection rates (bdr)* for all the burst from 5 weeks of data are given in parentheses, while the number of connections from the attack burst that are successfully associated with the attacks are given outside the parentheses.

Burst position	burst length (# of connections)	Attack type and category	*LOF approach*	*NN approach*	*Mahalanobis-based approach*	*Unsupervised SVM approach*
Week1, burst1	15	neptune (DOS)	15 (100%)	15 (100%)	4 (26.7%)	15 (100%)
Week2, burst1	50	guest (U2R)	49 (98%)	49 (98%)	49 (98%)	48 (96%)
Week2, burst2	102	portsweep (probe)	31 (30.3%)	63 (61.7%)	25 (24.5%)	48 (47.1%)
Week2, burst3	898	ipsweep (probe)	158 (17.6%)	428 (47.7%)	369 (41.1%)	375 (41.8%)
Week2, burst4	1000	back (DOS)	752 (75.2%)	62 (6.2%)	44 (4.4%)	825 (82.5%)
Week3, burst1	15	satan (probe)	0 (0%)	0 (0%)	0 (0%)	1 (6.7%)
Week3, burst2	137	portsweep (probe)	15 (10.9%)	118 (86.1%)	84 (61.3%)	115 (83.9%)
Week3, burst3	105	nmap (probe)	61 (58.1%)	105 (100%)	105 (100%)	97 (92.4%)
Week3, burst4	1874	nmap (probe)	1060 (57%)	1071 (57.1%)	993 (53%)	850 (45.4%)
Week3, burst5	5	imap (r2l)	4 (80%)	5 (100%)	4 (80%)	5 (100%)
Week3, burst6	17	warezmaster (u2r)	16 (94.1%)	15 (88.2%)	15 (88.2%)	16 (94.1%)
Week4, burst1	86	warezclient (u2r)	33 (38.4%)	38 (44.2%)	38 (44.2%)	42 (48.8%)
Week4, burst2	6104	satan (probe)	5426 (89%)	5558 (91.1%)	5388 (88.3%)	5645 (92.5%)
Week4, burst3	1322	pod (DOS)	957 (72.4%)	969 (73.3%)	680 (51.4%)	645 (48.8%)
Week4, burst4	297	portsweep (probe)	221 (74.4%)	259 (87.2%)	230 (77.4%)	271 (91.2%)
Week4, burst5	2304	portsweep (probe)	1764 (76.6%)	1809 (79%)	1095 (47.5%)	1969 (85.5%)
Week5, burst1	3067	satan (probe)	2986 (97.4%)	3022 (99%)	2983 (97%)	2981 (97.2%)
Week5, burst2	5	ffb (r2l)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Week5, burst3	1021	portsweep (probe)	937 (92%)	978 (98%)	938 (92%)	942 (92.3%)
Total	18424	-	13/19	14/19	11/19	12/19
Detection rate			68.4%	73.7%	57.9%	63.2%

Table 6. The comparison of anomaly detection schemes when applied on all the attack bursts from 5 weeks of data (SA – Surface Area between the real attack curve and the predicted (score) attack curve, $t_{response}$ – response time in the number of connections)

Burst position (burst length)	Attack type and category	*LOF approach*		*NN approach*		*Mahalanobis-based* approach		*Unsupervised SVM*	
		SA	$t_{response}$	SA	$t_{response}$	SA	$t_{response}$	SA	$t_{response}$
Week1, burst1	neptune (DOS)	0.03	1	0.22	1	0.25	1	0.02	1
Week2, burst1	guest (u2r)	0.22	1	0.01	1	0.03	1	0.04	1
Week2, burst2	portsweep (probe)	0.5	20	0.38	21	0.54	37	0.23	15
Week2, burst3	ipsweep (probe)	0.61	2	0.5	1	0.55	2	0.41	1
Week2, burst4	back (DOS)	0.3	3	0.74	3	0.82	5	0.37	2
Week3, burst1	satan (probe)	0.89	-	0.94	-	0.95	-	0.69	9
Week3, burst2	portsweep (probe)	0.8	30	0.2	1	0.32	4	0.28	2
Week3, burst3	nmap (probe)	0.3	2	0	1	0.1	3	0.09	2
Week3, burst4	nmap (probe)	0.33	13	0.34	1	0.52	5	0.27	3
Week3, burst5	imap (r2l)	0.14	2	0.0004	1	0.2	2	0.03	1
Week3, burst6	warezmaster (u2r)	0.08	1	0.12	1	0.15	1	0.07	1
Week4, burst1	warezclient (u2r)	0.56	1	0.58	1	0.69	2	0.52	1
Week4, burst2	satan (probe)	0.12	10	0.08	13	0.11	19	0.06	7
Week4, burst3	pod (DOS)	0.34	1	0.34	1	0.59	28	0.32	1
Week4, burst4	portsweep (probe)	0.48	17	0.13	21	0.39	37	0.12	16
Week4, burst5	portsweep (probe)	0.2	1	0.41	1	0.54	4	0.19	1
Week5, burst1	satan (probe)	0.06	21	0.02	38	0.08	47	0.03	14
Week5, burst2	ffb (r2l)	0.86	-	0.89	-	0.93	-	0.73	-
Week5, burst3	portsweep (probe)	0.49	8	0.04	8	0.06	12	0.05	9
Total: 18424	Detection rate	14/19 (73.7%)		15/19 (78.9%)		10/19 (52.63%)		12/19 (84.2%)	

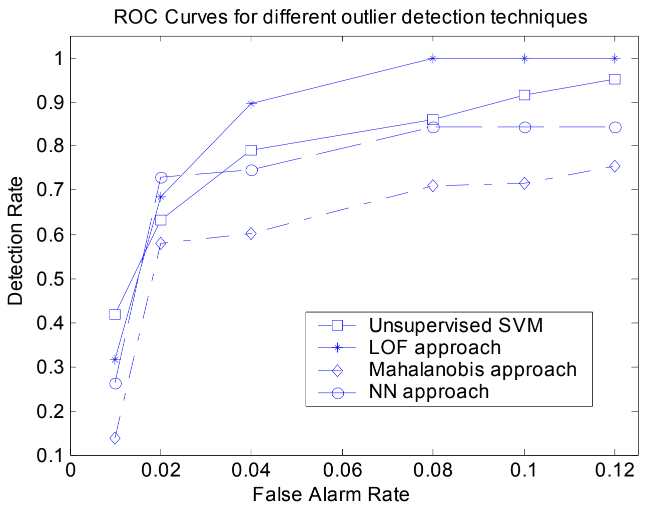


Figure 6. ROC curves showing the performance of anomaly detection algorithms on bursty attacks.

For fair comparison of all outlier detection schemes, ROC curves are computed for all proposed algorithms and illustrated in Figure 6, which shows how the detection rate and false alarm rate vary when different thresholds are used. It is apparent form Figure 6 that the most consistent anomaly detection scheme is the *LOF approach*, since it is only slightly worse than the *NN approach* for low false alarm rates (1% and 2%), but significantly better than all other techniques for higher false alarm rates (greater than 2%).

Table 6 reports on additional metrics for evaluation of bursty attacks, namely surface area and response time. As defined in section 2.1, the smaller the surface area between the real and the predicted attack curve, the better the intrusion detection algorithm. It is important to note that surface area in Table 6 was normalized, such that the total surface area was divided by the total number of connections from the corresponding attack burst. Since different bursty attacks involved different time intervals, we decided to measure response time as the number of connections. Therefore, the response time represents the first connection for which the score value is larger than the prespecified threshold. When considering these additional evaluation metrics, we also attempted to measure detection rate. In Table 6, we consider an attack burst detected if the normalized surface area is less than 0.5. Again, the two most successful intrusion detection algorithms were *NN* and *LOF* approaches, with 15 and 14 detected bursts respectively. When using the proposed additional metrics, the *Mahalobis-based approach* was again inferior to the *NN approach*, while on the other side the *unsupervised SVM* approach achieved again slightly worse detection rate that *LOF* and *NN* approaches.

It is interesting to note that the performance of both *NN* and *LOF* approaches was slightly better when using these additional metrics than the standard metrics. Since both schemes are based on computing the distances, they have similar performance on the bursty attacks because the major contribution in distance computation comes from the time-based and connection-based features. Namely, due to the nature of bursty attacks there is very large number of connections in a short amount of time and/or that are coming from the same source, and therefore the time-based and connection-based features end up with very high values that significantly influence the distance computation.

However, there are also scenarios when these two schemes have different detecting behavior. For example, the burst shaded gray in Table 5 (burst 2, week 2) corresponds to the attack that, when using the standard detection rate metric, was not detected with the *LOF* approach, but it was detected with the *NN approach*. Figure 7 illustrates the detecting of this burst using *NN* and *LOF*. It is apparent that the *LOF approach* has a smaller number of connections that are above the threshold than the *NN* approach (smaller *burst detection rate*), but it also has a slightly better response performance than the *NN* approach. It turns out that for specified threshold both schemes have similar *response time*. In addition, both schemes demonstrate some instability (low peaks) in the same regions of the attack bursts that are probably due to occasional "reset" value for the feature called "connection status". However, when detecting this bursty attack, the *NN approach* was superior to other two approaches. The dominance of the *NN approach* over the *LOF approach* probably lies in the fact that the connections of this type of attack (*portsweep* attack type, *probe* category) are located in the sparse regions of the normal data, and the *LOF approach* is not able to detect them due to low density, while distances to their nearest neighbors are still rather high and therefore the *NN approach* was able to identify them as outliers. The dominance of the *NN approach* over the *Mahalanobis-based approach* can be again explained by the multi-modal normal behavior. Finally, Figure 7 evidently shows that in spite of the limitations of the *LOF approach* mentioned above, it was still able to detect the attack burst, but with higher instability which is penalized by larger surface area.

When detecting the bursty attacks, very often there are scenarios when the normal connections are mixed with the connections from the attack bursts which makes the task of detecting the attacks more complex. It turns out that in these situations, the *LOF approach* is more suitable for detecting these attacks than the *NN approach* simply due to the fact that the connections associated with the attack are very close to dense regions of the normal behavior and therefore the *NN approach* is not able to detect them only according to the distance. For example, the burst 4 from week 2 involves 1000 connections, but within the attack time interval there are also 171 normal connections (Figure 8). Table 5 shows that the *LOF approach* was able to detect 752 connections associated with this attack, while the *NN approach* detected only 62 of them. In such situations the presence of normal connections usually causes the low peaks in score values for connections from attack bursts, thus reducing the burst detection rate and increasing the surface area (Figure 8). In addition, a large number of normal connections are misclassified as connections associated with attacks, thus increasing the false alarm rate.

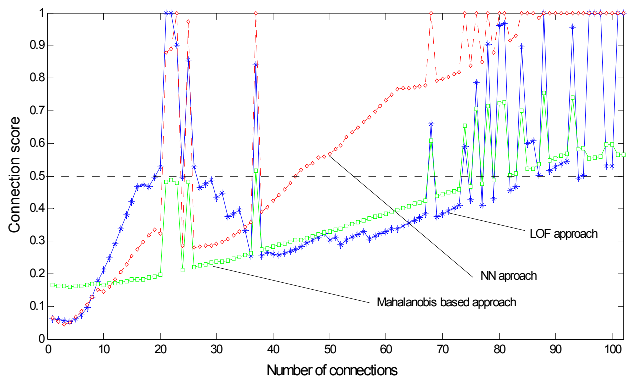


Figure 7. The score values assigned to connections from burst 2, week 2 (Figure is best viewed in color)

Table 7. The comparison of anomaly detection schemes applied on interleaved bursts of attacks. The first one was slow probing attack, the second one was DoS attack within the slow probing attack, and the third one was low traffic U2R attack.

Burst position (burst length)	Attack type and category	*LOF approach*	*NN approach*	*Mahalanobis based approach*	*Unsupervised SVM approach*
burst1 (999)	DOS	679 (68)	204 (20.4)	163 (16.3)	749 (74.9)
burst2 (866)	Probe	377 (43.5)	866 (100)	866 (100)	811 (93.7)
Burst 3 (5)	U2R	2 (40)	2 (40)	2(40)	2 (40)
Detection rate		1 / 3	1 / 3	1 / 3	2 / 3

Table 8. Number of attacks detected and detection rate for detecting single-connection attacks

Number of attacks	Attack type and category	*LOF approach*	*NN approach*	*Mahalanobis based approach*	*Unsupervised SVM approach*
13	U2R	6 (46.2%)	7 (53.8%)	5 (38.5%)	7 (76.9%)
11	R2L	7 (63.7%)	1 (9.1%)	1 (9.1%)	3 (63.7 %)
1	DOS	1 (100%)	1 (100%)	1 (100%)	1 (100 %)
Detection rate		14 / 25 (56.0%)	9 / 25 (36.0%)	8/25 (28 %)	11 /25 (44 %)

4.2.2. *Evaluation of Single Connection Attacks*. The performance of anomaly detection schemes for detecting single-connection attacks was measured by computing the ROC curves for all the proposed algorithms. Figure 8 shows how detection rate changes when specified false alarm rate ranges from 1% to 12%. Figure 8 shows that the *LOF approach* was again superior to all other techniques and for all values of false alarm rate. All these results indicate that the *LOF* scheme may be more suitable than other schemes for anomaly detection of single connection attacks especially for R2L intrusions.

Table 8 reports the detection rate for all individual single-attack types when the false alarm was specified to 2%. It is obvious that the *LOF, NN* and *unsupervised SVM* approaches outperformed the *Mahalanobis-based* scheme for all attack types. In this case, however, the *LOF approach* is distinctly better than the *unsupervised SVM* and *NN* approaches especially for R2L attacks, where the *LOF* approach was able to detect 7 out of 11 attacks, where the *unsupervised SVM* and NN approaches were able to pickup only three and one attacks respectively. Such superior performance of the *LOF* approach comparing to the *NN* approach may be explained by the fact that majority of single connection attacks are located close to the dense regions of the normal data and thus not visible as outliers by *the NN approach.*

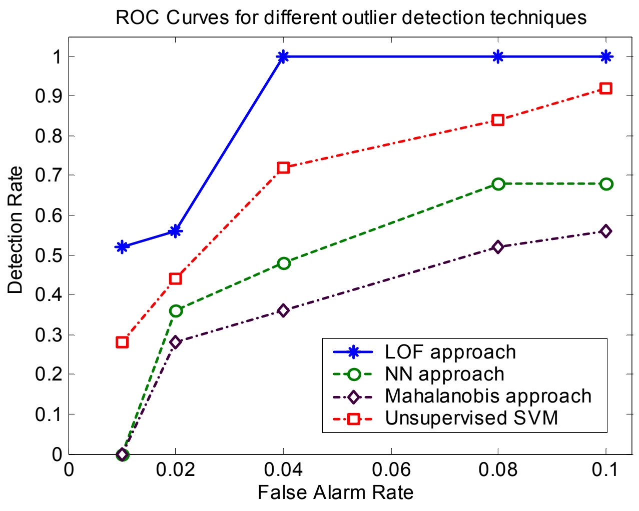


Figure 8. ROC curves showing the performance of anomaly detection algorithms on single-connection attacks.

4.3. Results from Real Network Data. Due to various limitations of DARPA'98 intrusion detection evaluation data discussed above [28], we have repeated our experiments on live network traffic at the University of Minnesota. When reporting results on real network data, we were not able to report the detection rate, false alarm rate and other evaluation metrics reported for DARPA'98 intrusion data, mainly due to difficulty to obtain the proper labeling of network connections.

Since a human analyst needs to manually evaluate outliers, it was not practical to investigate all of the outlier detection algorithms on the real network data. For this purpose we have selected the *LOF approach*, since it achieved the most successful results on publicly available DARPA'98 data set and it is more robust than other anomaly detection schemes that we used. The *LOF* technique also showed great promise in detecting novel intrusions on real network data. During the past few months it has been successful in automatically detecting several novel intrusions at the University of Minnesota that could not be detected using state-of-the-art intrusion detection systems such as SNORT [30]. Many of these attacks have been on the high-priority list of CERT/CC recently. Examples include:

- On August 9th, 2002, CERT/CC issued an alert for "widespread scanning and possible denial of service activity targeted at the Microsoft-DS service on port 445/TCP" as a novel Denial of Service (DoS) attack. In addition, CERT/CC also expressed "interest in receiving reports of this activity from sites with detailed logs and evidence of an attack." This type of attack was the top ranked outlier on August 13th, 2002, by our anomaly detection module in its regular analysis of University of Minnesota traffic. The port scan module of SNORT could not detect this attack, since the port scanning was slow. Figure 9 shows the number of incidents related to scanning of port 445/TCP reported to Internet Storm Center [31] in the last month all around the World.
- On June 13th, 2002, CERT/CC sent an alert for an attack that was "scanning for an Oracle server". Our anomaly detection module detected an instance of this attack on August 13th from the University of Minnesota network flow data by ranking connections associated with this attack as the second highest ranked block set of connections (the top ranked block

of connections belonged to the denial of service activity targeted at the Microsoft-DS service on port 445/TCP). This type of attack is difficult to detect using other techniques, since the Oracle scan was embedded within much larger Web scan, and the alerts generated by Web scan could potentially overwhelm the human analysts.

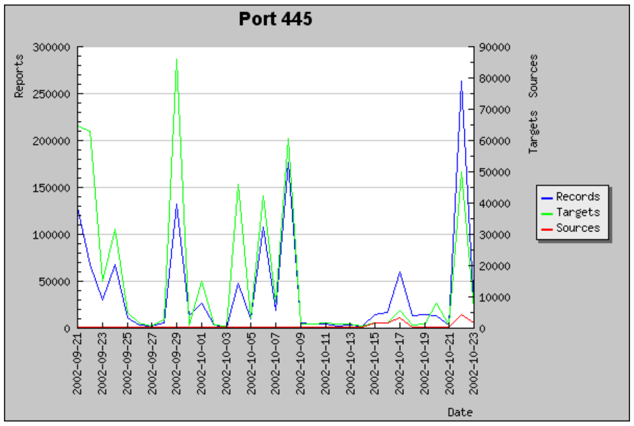


Figure 9. Number of scanning activities on Microsoft DS service on port 445/TCP reported in the World in the last month (Source www.incidents.org)

- On August 8th and 10th, 2002, our anomaly detection techniques detected a machine running a Microsoft PPTP VPN server, and another one running a FTP server, which are policy violations, on non-standard ports. Both policy violations were the top ranked outliers. Our anomaly detector module flagged these servers as anomalous since they are not allowed, and therefore very rare. Since SNORT is not designed to look for rogue and unauthorized servers, it was not able to detect these activities. In addition, for the PPTP VPN server, the collected GRE traffic is part of the normal traffic, and not analyzed by tools such as SNORT.

5. Conclusions and Future Work

Several anomaly detection schemes for detecting network intrusions are proposed in this paper. To support applicability of anomaly detection schemes, a procedure for extracting useful statistical content based and temporal features is also implemented. Experimental results performed on DARPA 98 data set indicate that the most successful anomaly detection techniques were able to achieve the detection rate of 74% for attacks involving multiple connections and detection rate of 56% for more complex single connection attacks, while keeping the false alarm rate at 2%. When the false alarm rate is increased to 4%, the achieved detection rate reaches 89% for bursty attacks and perfect 100% for single-connection attacks. Computed ROC curves indicate that the most promising technique for detecting intrusions in DARPA'98 data is the *LOF approach*. In addition, when performing experiments or real network data, the *LOF approach* was very successful in picking several very interesting novel attacks.

Considering the DARPA'98 data, performed experiments also demonstrate that for different types of attacks, different anomaly detection schemes were more successful than others. For example, the unsupervised SVMs were very promising in detecting new intrusions since they had very high detection rate but very high false alarm rate too. Therefore, future work is needed in order to keep high detection rate while lowering the false alarm rate. In addition, in the Mahalanobis based approach, we are currently investigating the idea of defining several types of "normal" behavior and measuring the distance to each of them in order to identify the anomalies. Since our experimental results exhibited very low detection rate for single-connection attacks that are very similar to normal connections, we will also scrutinize whether these attacks demonstrate different densities than the normal connections.

Our long-term goal is to develop an overall framework for defending against attacks and threats to computer systems. Although our developed techniques are promising in detecting various types of intrusions they are still preliminary in nature. Data generated from network traffic monitoring tends to have very high volume, dimensionality and heterogeneity, making the performance of serial data mining algorithms unacceptable for on-line analysis. Therefore, development of new anomaly detection algorithms that can take advantage of high performance computers is a key component of this project. According to our preliminary results on real network data, there is a significant non-overlap of our anomaly detection algorithms with the SNORT intrusion detection system, which implies that they could be combined in order to increase coverage.

References

1. Successful Real-Time Security Monitoring, *Riptech Inc. white paper*, September 2001.
2. W. Lee, S. J. Stolfo, Data Mining Approaches for Intrusion Detection, *Proceedings of the 1998 USENIX Security Symposium*, 1998.
3. E. Bloedorn, et al., Data Mining for Network Intrusion Detection: How to Get Started, *MITRE Technical Report*, August 2001.
4. J. Luo, Integrating Fuzzy Logic With Data Mining Methods for Intrusion Detection, *Master's thesis, Department of Computer Science, Mississippi State University*, 1999.
5. D. Barbara, N. Wu, S. Jajodia, Detecting Novel Network Intrusions Using Bayes Estimators, *First SIAM Conference on Data Mining*, Chicago, IL, 2001.
6. S. Manganaris, M. Christensen, D. Serkle, and K. Hermix, *A Data Mining Analysis of RTID Alarms*,

Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (*RAID 99*), West Lafayette, IN, September 1999.

7. D.E. Denning, An Intrusion Detection Model, *IEEE Transactions on Software Engineering*, SE-13:222-232, 1987.
8. H.S. Javitz, and A. Valdes, The NIDES Statistical Component: Description and Justification, *Technical Report, Computer Science Laboratory, SRI International*, 1993.
9. R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. P. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation, *Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000*, Vol 2, pp. 12-26, IEEE Computer Society Press, Los Alamitos, CA, 2000.
10. M. Ranum, Experiences Benchmarking Intrusion Detection Systems, *NFR Security Technical Publication*, December 2001.
11. R. P. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. W. Webster, M. Zissman, Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, *Proceedings of the Second International Workshop on Recent Advances in Intrusion Detection (RAID99)*, West Lafayette, IN, 1999.
12. Defense Advanced Research Projects Agency. DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/index.html
13. S. A. Hofmeyr, S. Forrest, and A. Somayaji, Intrusion Detection Using Sequences of System Calls, *Journal of Computer Security*, 6:151-180, 1998.
14. P. Helman, J. Bhangoo, A Statistically Base System for Prioritizing Information Exploration Under Uncertainty, *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 27(4):449-466, 1997.
15. A. Ghosh, A. Schwartzbard, A Study in Using Neural Networks for Anomaly and Misuse Detection, *Proceedings of the 8th USENIX Security Symposium*, 1999.
16. M. Mahoney, P. Chan, Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks, *Proceeding of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 376-385, Edmonton, Canada, July 2002.
17. T. Lane, C. E. Brodley, Sequence Matching and Learning in Anomaly Detection for Computer Security, *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, 43-49, 1997.
18. K. Sequeira, M. Zaki, ADMIT: Anomaly-base Data Mining for Intrusions, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, July 2002.
19. N. Ye, A Markov Chain Model of Temporal Behavior for Anomaly Detection, *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, 2000.
20. E. Eskin, W. Lee, S. J. Stolfo, Modeling System Calls for Intrusion Detection with Dynamic Window Sizes, *Proceedings of DARPA Information Survivability Conference and Exposition II (DISCEX II*), Anaheim, CA, 2001.
21. V. Barnett, T. Lewis, *Outliers in Statistical Data*, John Wiley and Sons, NY 1994.
22. C. C. Aggarwal, P. Yu, Outlier Detection for High Dimensional Data, *Proceedings of the ACM SIGMOD Conference*, 2001.
23. E. Knorr, R. Ng, Algorithms for Mining Distance-based Outliers in Large Data Sets, *Proceedings of the VLDB Conference*, 1998.
24. S. Ramaswamy, R. Rastogi, K. Shim, Efficient Algorithms for Mining Outliers from Large Data Sets, *Proceedings of the ACM SIGMOD Conference*, 2000.
25. M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, LOF: Identifying Density-Based Local Outliers, *Proceedings of the ACM SIGMOD Conference*, 2000.
26. P.C. Mahalanobis, On Tests and Meassures of Groups Divergence, *International Journal of the Asiatic Society of Benagal*, 26:541, 1930.
27. B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the Support of a High-dimensional Distribution, *Neural Computation*, vol. 13, no. 7, pp. 1443 1471, 2001.
28. J. McHugh, The 1998 Lincoln Laboratory IDS Evaluation (A Critique), *Proceedings of the Recent Advances in Intrusion Detection*, 145-161, Toulouse, France, 2000.
29. Tcptrace software tool, www.tcptrace.org.
30. SNORT Intrusion Detection System. www.snort.org.
31. Incident Storm Center, www.incidents.org.