“Low Cost” Network Intrusion Detection

Carol Taylor and Jim Alves-Foss
Center for Secure and Dependable Software
University of Idaho, Moscow, Idaho 83844

Email: {ctaylor@csds.uidaho.edu, jimaf@csds.uidaho.edu}

Abstract

A new approach to intrusion detection is needed to solve the problems of larger and faster networks and the constraints on system administrator’s time to manage security systems. Current network intrusion detection systems lack solutions to these two problems being complex in design and generally incurring larger costs in terms of operation and maintenance. We propose a new technique to solve these problems that we call “Low Cost” intrusion detection. Our approach features minimal network traffic measurement, an anomaly-based detection method and a limited attack scope. The method is similar to other lightweight approaches in its simplified design, but our approach, being anomaly based, should be more efficient in both operation and maintenance than other lightweight approaches. We present the method and perform an empirical test using MIT Lincoln Lab’s data.

1. Introduction

Intrusion detection is the branch of computer security concerned with monitoring a system for violations of a site’s security policy. The basic assumption of Intrusion Detection Systems is that other forms of security have failed leading to potentially harmful actions against the system being monitored. Generally, Intrusion Detection Systems (IDS’s) screen for security violations regardless of the source that can originate from either outside intruders or inside authorized users. IDS’s are commonly grouped based on their monitoring capability into either host-based or network-based systems. Host based systems generally utilize system log data for input and monitor intrusions affecting one or more hosts. Network based IDS’s focus on network traffic and concentrate on attacks that come from outside the system via the network.

IDS research has been ongoing for the past 15 years producing a number of viable systems, some of which have become profitable commercial ventures [1]. Yet, research has not kept up with today’s rapidly changing computing environment of increasing connectivity. With the growing size and speed of today’s networks, there is a critical need for IDS’s that can process large volumes of network traffic. A recent CMU report on intrusion detection systems noted that most network IDS’s can’t keep up with current Ethernet speeds and the trend is towards much faster networks. Another problem related to securing networks is that network administrators currently have little time for network security [10,14], which will only become worse as networks increase in size. The time constraints of network administration is an often overlooked problem in current IDS research where the trend is towards development of comprehensive solutions that require significant time for configuration and maintenance from the system administrator.

In this paper, we present a method for detecting network intrusions that addresses the problems of monitoring high speed network traffic and the time constraints on administrators for managing network security. Our approach is called “Low Cost” intrusion detection and is highly efficient in terms of machine and human management resources. Low Cost intrusion detection can be distinguished from most existing methods by the following features: minimal traffic measurements, simplified IDS management and limited attack scope. Other features that are shared with existing systems include anomaly based detection and real time operation. This paper will introduce our Low Cost ID approach and discuss it briefly in the context of other network IDS solutions. Details of our method will be presented followed by results from an empirical analysis using the Low Cost approach. Final sections will cover implementation issues and our conclusions.

2. Existing Solutions

As IDS’s can be categorized as either host based or network based, network based approaches to detection can further be divided into systems that only monitor the network and composite systems that watch both hosts and the surrounding network. In this paper, we will discuss these types of systems and their realizations found in the literature.

Strict network based systems include NSM[2], Bro[11], NFR[12] and NetStat[20]. NSM was an early system designed to monitor traffic between hosts on a LAN. Bro functions as a high-speed passive network monitor that filters traffic for certain applications. NFR was designed as a flexible tool for network data generation whose attributes include its own language for creating filters that are then compiled into the tool. NetStat is a network IDS that offers customization of event collectors and the option of automatic placement of network event collectors.

Systems that monitor both hosts and networks include Emerald[9], Grids[19] and Dids[18]. Emerald, was designed to detect intrusions in large distributed networks. It is a large hierarchical system that can respond to threats on local targets and coordinate its monitors to form an analysis hierarchy for network-wide threats. Grids accumulates results from both host and network based components which are displayed in a graph. The graph allows easy viewing of attacks that might span the network. The Dids IDS is an extension of NSM and utilizes data from both host auditing sytems and LAN traffic to detect intrusions.

IDS detection methods fall into two general categories of rules (or signature based) and anomaly based detection. Rules based detection typically is done with an expert system by filtering activity according to a predefined set of rules. Signature-based methods match intrusions to exact patterns of stored misuse behavior. Anomaly based methods seek to characterize normal system behavior and detect deviations from normal. The trade-off between anomaly based and rule based methods is that rule based methods can’t detect new or novel attacks but their false-positive rate is lower. Anomaly-based detection methods have a potential higher false-positive rate due to inexact methods of intrusion identification, but these inexact methods allow detection of new attacks.

Most current network IDS's use rule based methods of detection. This includes Bro, NSM, NFR, NetStat (states are similar to rules), Emerald, Grids and Dids. Emerald is the only current system that uses anomaly detection with the inclusion of a statistical component that complements the expert system.

For completeness, we will mention two public domain network IDS's that claim to be "lightweight" IDS's. These systems are Snort[13] and Shadow[10]. Snort is a rule-based network IDS for small, lightly utilized networks. Snort's features include simple rule format for easy rule creation, packet payload inspection for pattern matching, and a streamlined architecture. Shadow is more of a network sensor than an IDS an relies heavily on tcpdump[3]. Input from a Shadow sensor is passed to an Analyzer that utilizes tcpdump rules, the results of which are fed to a web interface. Shadow does not run in real time but performs periodic dumps of collected network traffic. While both of these systems claim to be "lightweight", we believe the reliance on rules is contrary to a truly lightweight approach. For each new attack, new rules must be generated which over time could create a prohibitively large rule base. System administrators must constantly update their system to stay current with known attacks. Although the systems may be "lightweight" in terms of the size and complexity of their executables, the day-to-day operation and maintenance of these systems brings them out of the realm of a true lightweight system; one that fits more into the plug-and-go category of applications.

3. The Low Cost Method

Our Low Cost method differs from all of these systems in its approach to IDS. The most important feature of our approach is its emphasis on little human involvement in the system's management and configuration. We have implemented a statistical anomaly detection method that differs from previous statistical approaches in its speed and ease of use. Prior anomaly based methods have not been easy to configure or maintain requiring knowledge of normal system parameters [4, 17]. Our method is designed to be self-configuring requiring no human input on the normal system state. The main advantage of anomaly detection over rule or signature based methods, is that it minimizes system administration because it detects new attacks automatically without the need for rule creation or update. A possible weakness is a potential higher false positive rate, which could be managed by allowing adjustment of the detection significance threshold.

In terms of data collection, Low Cost ID emphasizes minimal traffic measurement. Measuring only packet headers allows for high speed traffic monitoring. Our method also deliberately limits attacks to those that exploit vulnerabilities in the network protocols. Unlike most other network IDS's we do not attempt to catch a wide variety of intrusion types. Our intention is to develop a network tool that would work in concert with other tools such as a host based IDS to provide a comprehensive solution. The advantages of deliberately limiting the attack scope, allows for streamlining of the system and a large reduction in complexity.

4. Traffic Measurement

A "lightweight" intrusion detection system must handle high-speed traffic in a real-time format. Our Low Cost approach to network intrusion detection seeks to measure the minimum amount of information from the network and still be able to determine normal from anomalous traffic. Previous authors have noted that measuring just packet headers as opposed to packet contents greatly speeds throughput [10,11]. We have found that looking at tcpdump logs of network traffic generated by known attacks, one obvious behavior is that TCP flag distribution changes significantly between normal and attack traffic. Many attacks can be characterized by large numbers of TCP control packets such as Syn, Fin and Reset packets along with low numbers of P and Ack packets. Also, in anomalous streams of traffic, there is an absence of the normal flow of data contained in the packets that can be monitored by counting the number of bytes transferred. Therefore, we decided to monitor counts of the TCP flags and the number of bytes transferred for each packet. Also, we observed that in anomalous traffic there is a low number of packets transferred for any particular source to destination ip+port combination. This feature can be captured by aggregating the traffic into sessions consisting of all traffic between a unique source and destination ip+port. Aggregating at the session level highlights intruder traffic since these anomalous sessions contain a different distribution of packets than normal sessions.

5. Statistical Methods

Statistical methods have long been used to detect anomalies in system and network audit data [4,17]. To date, statistical anomaly detection has relied on probability-based methods by comparing new sets of measurements to a normal database of measurements or summary statistics. If the new measurement has a low probability that it came from the historical measurement distribution, then a flag is raised for a potential intrusion [4,17].

We propose a completely different statistical procedure for anomaly detection based on multivariate statistics. Multivariate statistics are appropriate for any data set where multiple measurements are taken with possible correlations between the measurements. Multivariate techniques in general, account for the correlation structure of the variables being analyzed often yielding a more complete picture of the analysis results than if the variables had been analyzed separately [5].

5.1 Cluster Analysis

Cluster analysis is a multivariate technique used for finding groups in observed data. The objective is to form groups in such a way that objects in each group are similar to each other but as different from other groups as possible [6]. Cluster analysis is used when researchers have no apriori hypothesis about their data but are in an exploratory stage of research [6]. We chose cluster analysis as a means of forming normal groups of TCP/IP sessions; we have found no reference to similar techniques in the literature.

Cluster analysis forms clusters based on dissimilarities between objects. Quantitatively, dissimilarities are distances computed from the measured variables, which in our case are the TCP flag and byte counts. The most common distance measure used in cluster analysis is Euclidian Distance, the geometric distance in multidimensional space [5].

Euclidian distance can be measured as follows:

$$\boldsymbol{dist(x,y) = \left(\sum_{i=1}^{p}(x_i - y_i)\right)^{1/2}}$$

Where x and y are two objects to be clustered.
i = represents the variables measured, and ranges from 1 to p.

Most texts on cluster analysis place cluster algorithms into two major categories: partitioning and hierarchical methods [6]. Partitioning works by dividing the data into k groups where $k < n$, and n is the number of objects to be clustered. Hierarchical methods (agglomerative) start with all n objects as single clusters and combine objects until ultimately a single cluster is formed [15].

A common problem in cluster analysis is determining some optimum number of clusters [6]. Unfortunately, there is no well-defined rule for determining the number of clusters in a given data set. Usually, there are some general statistics given as guidelines, but deciding the number of clusters is often left up to the researcher. One of the general statistics for determining the number of clusters is the Pseudo T^2 statistic that closely resembles a multivariate T^2 test for deciding group differences. If the value is small, then the two clusters can be combined, if large, then the two clusters show differences and should probably not be combined [15].

Another common method for verifying cluster results is through visual confirmation of cluster formation. If the number of variables is greater than three, then the data cannot be easily plotted and some type of data reduction will be needed. A commonly used data reduction technique is Principal Components Analysis (PCA). PCA combines variables based on the correlation between them. Highly correlated variables will be combined into an aggregated variable called a principal component. The goal in using PCA is to reduce the multidimensional space down to a few dimensions so the data can be plotted to identify potential clusters [15]. The cluster results are then overlaid onto this PCA plot to see how well the cluster solution fits the natural distribution of the data points.

5.2 Sampling Methodology

In performing an analysis of network data, that constitutes a potentially unlimited population, we must somehow limit the amount of information that we input into our normal database. Unlike previous probability based techniques, which constantly update the normal state with new information [4], our technique collects the data once and then updates the database later only if necessary to incorporate new normal behavior[1]. Therefore, it is important to determine when we have collected enough data to build our normal cluster database. If we collect too few data points then the sample will not be representative of the normal network state and later testing between new sessions and the clusters will produce a large number of false positives since normal points will be identified as anomalies. In an effort to capture a wide range of normal behavior, we decided to include samples of each major network traffic type. Thus, each of the most frequent traffic types will be sampled as a separate population using traditional statistical sampling theory. For example, separate random samples of FTP, HTTP, SMTP and other types of traffic will be independently collected. We believe that applications will form clusters if not strictly by type, then by similarities between the types such as HTTP and FTP-DATA which both focus on file transfer. By insuring that each traffic type is adequately represented, the combined samples should represent the range of normal

[1] Unlike anomaly-based systems that profile user behavior, which is often erratic at best, our approach profiles the behavior of protocols, which change over much longer time periods.

network traffic. Infrequent network types can be included along with the major traffic types.

The general procedure for computing sample size is to get an initial estimate of a sample standard deviation, which is used in calculating a bound on the sample error.

The formula for computing a sample bound is:

$$B = 2\left(\left(\frac{s^2}{n}\right)\frac{N-n}{n}\right)^{1/2} \qquad D = \frac{B^2}{4}$$

Where B is the bound on the error, s^2/n is the sample variance, N is the population total and D is the adjusted bound to be used in the subsequent calculation of sample size. The sample error bound is then used to compute a sample size as follows:

$$n = \frac{Ns^2}{((N-1)D + s^2}$$

where n = the sample size calculated from the formula, N is the population total, D is from the previous equation and s^2 is the population variance which can be estimated from s^2/n the sample variance [16].

This sampling technique will be applied to each major network group where “major” is loosely defined as a certain percentage of network traffic.

6. Empirical Analysis

In order to test the feasibility of our Low Cost ID approach, we evaluated the method against a real data set. Using these results, we can determine its effectiveness in detecting intrusions, identify weaknesses and assess the potential false positive rate.

6.1 Data Set Selection

It was decided to use a publicly available data set for our analysis explicitly created for testing IDS’s. The data set was created by, MIT Lincoln Labs, for their 1998-1999 IDS evaluation study [8]. We chose this particular data set due to the time required to create a good data set and the fact that using a known data set would lend credibility to our results.

The Lincoln Labs data consists of network traffic dumps and host audit logs saved as files. For our analysis, we selected outside tcpdump data that contained simulated network traffic captured outside the firewall of a medium sized LAN. The tcpdump files were generated daily with some of the files containing embedded labeled attacks.

6.2 Experimental Method

The MIT Lincoln Labs data set contains several weeks of daily tcpdump files for both 1998 and 1999. Each daily file contains up to 1 milllion TCP records. We developed methods for selecting a reasonable subset of data from this huge population of records and for aggregating the individual packets into sessions of unique source ip+port to destination ip+port. In addition, we identified a set of attacks that we could reasonably hope to detect from among the attacks embedded in the MIT data.

6.3 Subset Identification

Because the data set was so large, we had to develop a record screening method that would insure we had captured the entire range of normal network traffic. As previously outlined in Section 5.2, we included records according to traffic type frequency. In looking at traffic type frequency, it was noticed that about nine groups dominated the tcpdump traffic. Other types of traffic occurred within the data but the frequency was so low that we elected to ignore these types for the purposes of this empirical evaluation. The nine groups identified along with their frequencies are listed in Table 1.

As can be seen from the group frequencies, the data is completely dominated by HTTP records, which means that most of the network traffic is web traffic. The second most frequent traffic types are SMTP, electronic mail and FTP-DATA/FTP, file transfer traffic. The other groups represent very few records but were included in an attempt to capture a wide range of normal traffic seen on this network.

Within a single days worth of traffic, the less frequent groups such as Pop3, Auth, Time, and Telnet, had only a few records. It was thus decided to select one of the daily files as a base file, collect records from other files for the low frequency groups, and add these records to the base file. One important assumption with this method is that there are no systematic differences in the network traffic between days. Since we are interested in getting representation from all of the traffic types we wanted to have at least some minimum number of records from each of the nine dominant types and then take a random sample from each group (see Section 5.2). It was decided that a minimum number of 30 records would needed for each group since it was difficult to find more than 30 records in two weeks worth of data for the least frequent groups.

Traffic Type	***Frequency – Percent %***	***Description***
http	80-98	Web based traffic
smtp	7-33	Mail transfer protocol
ftp-data	1-3	File transfer – data
telnet	.5-2	Remote connection
ftp	.2-1	File transfer – connection
finger	.2-.5	Identification information
auth	<.2	Authentication svice
time	<.2	Time server – service
pop3	< .2	Mail protocol

Table 1: Frequency of network traffic types

In order to insure that there was no bias in selecting the records and to provide further empirical evidence for testing our method, two separate data files were created. After constructing the data files, Mon99 and Wed99, using two separate base files from different weeks in the 1999 MIT data, a SAS[15] program was written to perform random sampling from each group to get an error estimate for the highest variance variable, Total Packets. Sample size calculations indicated that samples of 28-30 would be needed for the high frequency data types, HTTP, SMTP, FTP-DATA and FTP and smaller sample sizes of 20-24 would be enough for the low frequency types, Auth, Pop3, Time and Telnet. Since there was so little difference in the sample sizes between groups, it was decided to sample all nine groups at 30 providing a total data set of 270 sessions. The sample size calculations were performed on Wed99 and extrapolated to Mon99 since there was little difference between the variability of Total Packets between the two files.

6.4 Attack Identification

Of the attacks included in the MIT Lincoln Labs data, attacks that were classified as remote probes and DOS types of attacks could potentially be detected from analyzing network headers [7]. Thus, five TCP attacks were selected for testing our Low Cost method. These attacks are listed in Table 2, along with attack descriptions and the attack impacts as defined by Kendell in his classification of attacks included in the Lincoln Lab study [7].

Attack	***Description***	***Attack Impact***
Portsweep	Scans multiple ports for services	Probe Services
Neptune	Syn flood denial of service	Deny temporary
Satan	Network probing tool	Probe Services
nmap	Network mapping using nmap	Probe machines/services
Mailbomb	DOS for the mail server	Deny temporary/perm

Table 2: TCP attacks from MIT Lincoln Labs data set

6.5 Statistical Analysis

Prior to conducting the cluster analysis, the variables were analyzed to see if all of the variables should be included in the analysis. Summary statistics[2] including the min, max, means and standard deviations were computed for each traffic type for both the data files, Mon99 and Wed99. The variables measured included TCP flag counts for Syn, Fin, Ack, Push, Reset and Urgent plus the a count of the Total Packets and Total Bytes transferred. After looking at the summary statistics, it was decided to drop the TCP flag variables, Syn, Fin and Reset from further analysis because they showed little variability between the groups. Urgent count was also dropped since this count was zero for all the records analyzed. Total Bytes displayed a huge range of values from 0 to over 470,000 and was discarded because the high values from this variable would dominate the cluster analysis. To retain the usefulness of the control flag counts in later discriminating between normal

[2] All statistical analysis was performed with the SAS statistical software system [SAS].

and anomalous sessions, it was decided to create a new variable, Totcon, consisting of Syn+Fin+Reset / Total Packets. This variable would represent the percent of control packets per session. Similarly, to utilize bytes transferred but dampen the effect of the high values, another variable was created, Average Bytes, which equaled Total Bytes/Total Packets. The set of variables included for further analysis were Average Bytes, Totcon, Push and Ack flag counts and Total Packets.

In order to graph the data to see its distribution, the data was reduced using PCA (see Section 5.1) and graphed as 3D plots (Figures 1 and 2). As can be seen from the plots of the two data sets, the data does not form well-defined clusters, but instead displays a large number of points distributed centrally with long fingers of data extending in several directions. The lack of cleanly separated groups may be normal for network traffic or might be due to the simulated nature of the Lincoln Lab data. Much of the data appears to be highly similar. Analysis of real network data would confirm whether this distribution of data points is typical of network data given the nine traffic types included in the analysis.

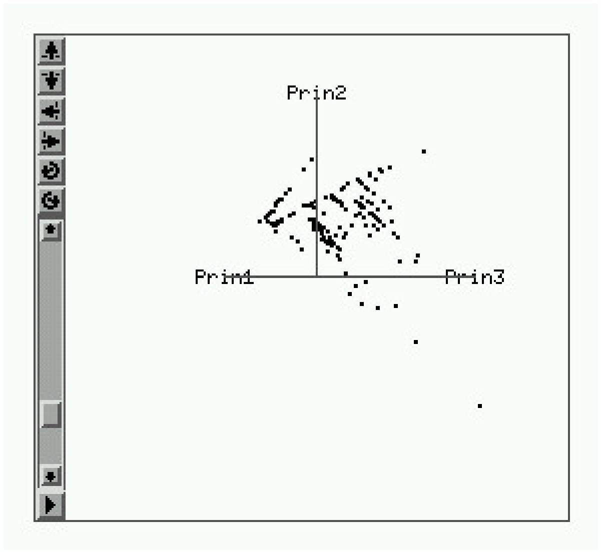

Figure 1: Wed 99 3-D plot of first three principal components

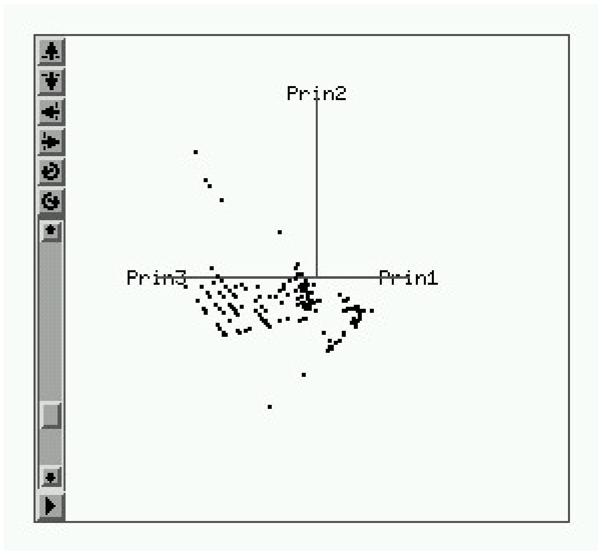

Figure 2: Mon99 3-D plot of first three principal components

Next, cluster analysis was performed for each of the data sets, Mon99 and Wed99 yielding the clusters described in Table 3. For the data file, Wed99, seven clusters were identified and for Mon99, five clusters were sufficient to describe the data. The decision to select these seven and five clusters respectively was based on the Pseudo T^2 statistic as described in Section 5.1

Wed 99			**Mon99**	
Cluster	**Traffic Type**	**Main Trait**	**Traffic Type**	**Main Trait**
1	time, finger, telnet	Low packets	time, auth, pop3	Low packets
2	auth, ftp-data	Higher bytes, 1	auth	Med. packets
3	finger, ftp-data	Higher bytes, 2	finger, smtp, http	Higher bytes, 2
4	pop3	Higher acks, 1,2	ftp, telnet	Higher acks P, 1-3
5	ftp,telnet	Highest Ack P Total	telnet	Highest Ack P Total
6	http, ftp-data	Low P Ack		
7	smtp, pop3	Higher P than Ack		

Table 3: Cluster composition for Wed99 and Mon99 data files

In examining the clusters, it appears that the traffic types are split between clusters with few clusters consisting of a single type. However, similar types do tend to group together such as SMTP and Pop3, both mail protocols, HTTP and FTP-DATA, file transfer types, and Telnet and FTP, which feature user interaction.

6.6 Intrusion Identification

Once the normal cluster DB was created for each data set, the five selected attacks were tested against each cluster DB. While Euclidian distance was used in creating the clusters, for anomaly detection, we needed a distance metric that would map to a known statistical distribution so significance could be calculated. The selected distance measure is the Mahalanobis Distance that maps directly to a ?^{2} distribution. The formula for the Mahalanobis Distance is:

$$\boldsymbol{D(x_{i,}y_i) = (x_i - y_{ik})\, \Sigma_k^{-1} (x_i - y_{ik})'}$$

Where $x_{i,}$ is the new session vector, $y_{ik,}$ is the cluster mean for the kth cluster, $S^{\boldsymbol{-1}}$ is the variance-covariance matrix for the kth cluster, and i ranges from 1 to p, the number of variables measured.

Tables 4 and 5 show the Mahalanobis Distances computed between the five attacks and the normal clusters along with the distances between selected sessions and the normal clusters.

Type	*Cluster 1*	*Cluster 2*	*Cluster 3*	*Cluster 4*	*Cluster 5*	*Cluster 6*	*Cluster 7*
Portsweep[3]	319	2393	3391	43282	2796	71	611358
Satan	275	2102	3834	41203	2722	62	618048
Neptune	84	681	671	16596	1455	30	267237
Mailbomb	8769	5354	1617	268	60	9.3*	269
Pop3	1389	2212	1107	.81*	68	8.3*	251
Auth	355	2.6*	422	1025	263	9.5*	29772
Time	.15*	177	15*	2397	448	7.2*	45275
Ftp-data	420940	73426	658	40219	60	9.3*	269
Telnet	9045497	2792646	62197424	3046285	10*	3368	6016501
Telnet	68792220	21089242	4746545	2349253	3*	24085	4718378
Smtp	3155646	97687	218268	10709158	4*	11126	2159827

Table 4. Mahalanobis Distances for attack and normal sessions, Wed99

Note: distance translates to a χ^2 distribution with 5 df which at .001 significance level is 20.5
*means distance is not significant

In evaluating the distances between the attacks, Portsweep, Satan and Neptune, we find that all of the distances between these attack sessions and the clusters are significantly different. Mailbomb, however, matches Cluster 6 with a non-significant distance of 9.3. Since individual Mailbomb sessions appear normal, it is likely that this attack would match one of the normal clusters. The anomalous nature of Mailbomb appears at a higher level of aggregation when multiple sessions are evaluated. A possible solution to Mailbomb and other similar attacks will be addressed in the section on Implementation Issues. In contrast to the results from the attacks, all of the normal sessions match at least one and sometimes several of the clusters, which is indicated by non-significant distances as is expected. The normal sessions tested were selected from non-clustered sessions and represents a wide range of values.

Type	*Cluster 1*	*Cluster 2*	*Cluster 3*	*Cluster 4*	*Cluster 5*
Portsweep	6390	5819	323	3284	48075880
Satan	5522	5244	331	3302	48052114
Neptune	1048	4694	120	1795	26993707
Mailbomb	462	22492	2.3*	70.8	1711039
Pop3	6.7*	5450	2.7*	81	1897470
Auth	83	352	11.3*	446	6512639
Time	.96*	1507	12.6*	569	9428474
Ftp	14945	127630	107.6	2.0*	226046
Telnet	20303736	53685766	196250	14.3*	89.5
Telnet	4962511	19016405	49720	20.6**	987

Table 5. Mahalanobis Distances for attack and normal sessions, Mon99

*means distance is not significant

[3] Nmap matched Portsweep's results exactly

** means distance is barely significant

Examining the results of the second data file, Mon99, with a fewer number of clusters, it appears that the results are similar to Wed99. The attacks, with the exception of Mailbomb, all display distances that are significantly different from the normal clusters while the normal sessions match one or more of the five clusters. One interesting result comes from testing an extremely large telnet session, which was outside the range of the data clustered, but was still barely significant at a distance of 20.6. This can be considered a borderline false positive. The problem of false positives and adjusting the sensitivity level of the method will be addressed in the next section.

7. Implemention Issues

Implementing the Low Cost approach to IDS efficiently for real time operation posses some interesting challenges. Currently, the method is based on tcpdump files created and saved for offline analysis. However, tcpdump is capable of real-time operation and based on other system's favorable reports[10] of tcpdump's capacity to handle large traffic volumes, we plan to continue using it to monitor the network traffic stream.Our Low Cost ID tool will need two distinct phases of operation. Phase 1 will consist of data collection and database creation, while Phase 2 will be real-time operation and anomaly detection. During the data collection phase, the system would need to be closely monitored for possible attacks since this phase is supposed to capture only normal data. If attacks exist in the normal data used to create the clusters, then future occurrences of these attacks will be labeled normal. This is a common concern with anomaly based IDS's [1]. Also, the data collection method must determine automatically when enough normal data has been gathered. A real-time method similar to the one described in this paper could be developed where the number of major groups is first identified and then sampled using individual group random sampling. Perhaps the greatest implementation challenge will be to create a cluster algorithm that will automatically create clusters that encapsulate the normal behavior of the network without human assistance. It is not expected that network system administrators will be able to assist in the creation of the normal network database. Statistics that assist in deciding optimal cluster solutions such as the pseudo T^2 statistic can be built into the cluster routine.

All anomalies will be saved in a log file for analysis and inspection. Ideally, there should be a way to incorporate normal misidentified behavior into the cluster database without having to rebuild the database. Normal sessions that need to be added to the database must be identified by the system administrator and then incorporated into the DB. Each normal session would be added to the cluster according to the smallest distance or a new cluster could be created if the session represents new behavior. The database can thus change in response to new behavior not captured in the original creation of the database.

False positives could be a potential problem with real-time operation. Adjusting the significance level of the distance measure would change the detection rate by allowing larger distances. The significance level could be left as a tunable parameter that could be set in response to the number of false positives.

Finally, in solving the problem of individual attack sessions that appear to be normal, the strictly anomalous detection method could be supplemented by some general rules. For example, individual sessions that are normal but together appear to flood a given service could easily be detected by keeping track of the total number of bytes during a set time period. For the Mailbomb attack and other script generated sessions, a rule for detection might be if several sessions over a short time period all have the exact same counts of packets and bytes. In looking at a tcpdump of Mailbomb, a variable number of packets appear to be sent until the stream is aggregated into sessions. Then, a long stream of identical sessions appears that individually look normal but taken all together try to flood the smtp service.

7. Conclusion and Future Work

In this paper, we presented our work on Low Cost intrusion detection based on minimal network traffic measurement. Empirical evaluation with an established data set was highly successful in identifying intrusions while creating few false positives among the normal sessions evaluated. Our approach represents a departure from the status quo of large, complex network IDS's which are comprehensive solutions to ID. Low Cost ID is consistent with the current trend of a layered approach to security with the deployment of multiple tools with complimentary functionality.

Future work includes implementing and testing a similar approach for the UDP and ICMP protocols. Since UDP traffic does not keep state information, experimentation with counts of specific UDP packet fields must be done to identify the information that distinguishes normal from anomalous traffic.

Another area to be investigated will be to identify a different distance metric other than Mahalanobis distance. This measure requires computation of the variance-covariance matrix for each cluster and that could be time consuming if the database needs frequent updates. Forming an empirical distribution based on Euclidian distances of each point with its cluster mean might prove better for real-time operation. Then distances between cluster means and new sessions could be compared to a cut-off value of the empirical distribution.

8. References

[1] J. Allen et al, "State of the practice Intrusion Detection Technologies", Carnegie Mellon, SEI, Tech Report, CMU/SEI-99-TR-028, ESC-99-028, January 2000

[2] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, " A Network Security Monitor", In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, April 1990, pp 296-304

[3] V. Jacobson, C. Leres, and S. McCanne, **tcpdump**, LBNL, University of California, June 1997, ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

[4] H. S. Javitz, and A. Valdes, "The NIDES Statistical Component: Description and Justification", Tech. Report, Computer Science Lab., SRI-Int., Menlo Park , CA, March 1994

[5] D. E. Johnson, **Applied Multivariate Methods for Data Analysis**, Brooks/Cole Publishing Co., 1998

[6] L. Kaufman and P. J. Rousseeuw, **Finding Groups in Data: An Introduction to Cluter Analysis,** Wiley Series in Probability and Mathematical Statistics, John Wiley and Sons, Inc., 1990

[7] K. Kendell. “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems”, Masters Thesis, MIT, June 1999

[8] R. Lippmann and M. Zissman, “Intrusion Detection Technical Evaluation – 1998 Project Summary”, www.darpa.mit/ito.

[9] P. G. Neumann, P. A. Poras, “Experiences with Emerald to Data”, *Proceedings of 1st Usenix Workshop on Intrusion Detection and Network Monitoring,* Santa Clara, CA, Apr. 11-12, 1999

[10] S. Northcutt, V. Irwin, B. Ralph, **Shadow**, Naval Surface Warfare Center Dahlgren Lab., 1998

[11] V. Paxson, “Experiences Learned from Bro”, *login; The Usenix Assoc. Magazine,* Sept. 1999, 21-22

[12] M. J. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, E. Wall, “Implementing a Generalized Tool for Network Monitoring”, *Proceedings of 11th Syst. Admin. Conf. (LISA 97),* San Diego, CA, Oct. 1997

[13] M. Roesch, “Snort – Lightweight Intrusions Detection for Networks”, www.clark.net/~roesch/security.html

[14] D. Ruiu, “Cautionary Tales: Stealth Coordinated Attack HowTo”, www.nswc.navy.mil/ISSEC/CID/Stealth_Coordinated_Attack.html, 1999

[15] SAS Institute, **SAS/STAT Users’ Guide, Version 6, Fourth Edition, Vol. 1**, SAS Institute, 1990

[16] R. L. Scheaffer, W. Mendenhall III and R. L. Ott, **Elementary Survey Sampling**, Wadsworth Publ. Co., 1996

[17] S. E. Smaha, “Haystack: An Intrusion Detection System”, *Proceedings IEEE Fourth Aerospace Computer Science Applications Conference*, Orlando, FL, Dec. 1988

[18] S. E. Smaha, T. Grance, D. M. Teal and D. Mensur, “Dids – Motivation, Architecture, and an Early Prtotype”, *Proceedings of 14th National Computer Security Conference,* Washington, DC, Oct. 1991, pg. 167-176

[19] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle, “GrIDS – A Graph-Based Intrusion Detection System for Large Networks”, *The 19th National Information Systems Security Conference,* Oct. 1998, pp 361-370

[20] G. Vigna, R. A. Kemmerer, “NetStat: A Network-based Intrusion Detection Approach”, *Proceedings of the 14th Annual Computer Science Applications Conference*, Scottsdale, AZ, Dec. 1998