An Evaluation of Data Mining Approaches to Network Intrusion Detection

Justin Ethier | Sanjay Srivastava | Theodore Sadler
{jethie1, ssriva1}@towson.edu, TheodSad@aol.com

COSC 757 | Data Mining | Dr. Harry Zhou
Department of Computer & Information Sciences
Towson University, Suite 406
Towson, Maryland 21252

Abstract

Financial institutions, corporations, governments, and private citizens are becoming increasingly dependent on the Internet. Given this dependency it is imperative that security tools such as Intrusion Detection Systems (IDS) are deployed to protect these computer networks from hackers, insiders, and cyber terrorists. In this paper we consider a relatively new technique, counts (bag) of system calls, for use in IDS. This technique will then be compared with traditional approaches to anomaly detection. Evaluations will be performed against derivatives of the University of New Mexico and MIT Lincoln Lab (DARPA) data sets created for evaluation of computer network intrusion detection systems. We will use an array of relevant Data Mining algorithms to perform classification of these data sets and present an in depth analysis of the results. Using primary metrics of Detection and False Positive rates, this analysis will compare and contrast both the new and traditional approaches to anomaly detection in an IDS in an attempt to create an overall framework for introducing these techniques into a production IDS.

1 Introduction

1.1 Background

With the popularity of the Internet, businesses and individuals are becoming increasingly dependant upon computer networks. Thus it is imperative that these networks be secured against malicious users, including insiders and terrorists. This is a challenging problem that is compounded by the fact that new vulnerabilities are discovered on a daily basis, and new exploits are often released before vendors or network operators have time to properly respond.

Intrusion Detection Systems (IDS) attempt to address these issues by monitoring network traffic for suspicious behavior. If a suspicious event is detected, the IDS will raise an alarm – for example, by sending an email to the network administrator.

An example IDS is illustrated below in Figure 1. In this example, an IDS is used at the firewall to monitor all incoming traffic from the internet to the corporate network. It is important to note that in this diagram the IDS is used as a second line of defense after the corporate firewall. A secondary IDS is also in place to monitor traffic to the file server – for example, the server may contain sensitive information that warrants extra security. Finally, all IDS events are logged to a central database server. It is important to note that this is just an example – any real world IDS will be customized to the needs of the particular network on which it is deployed.

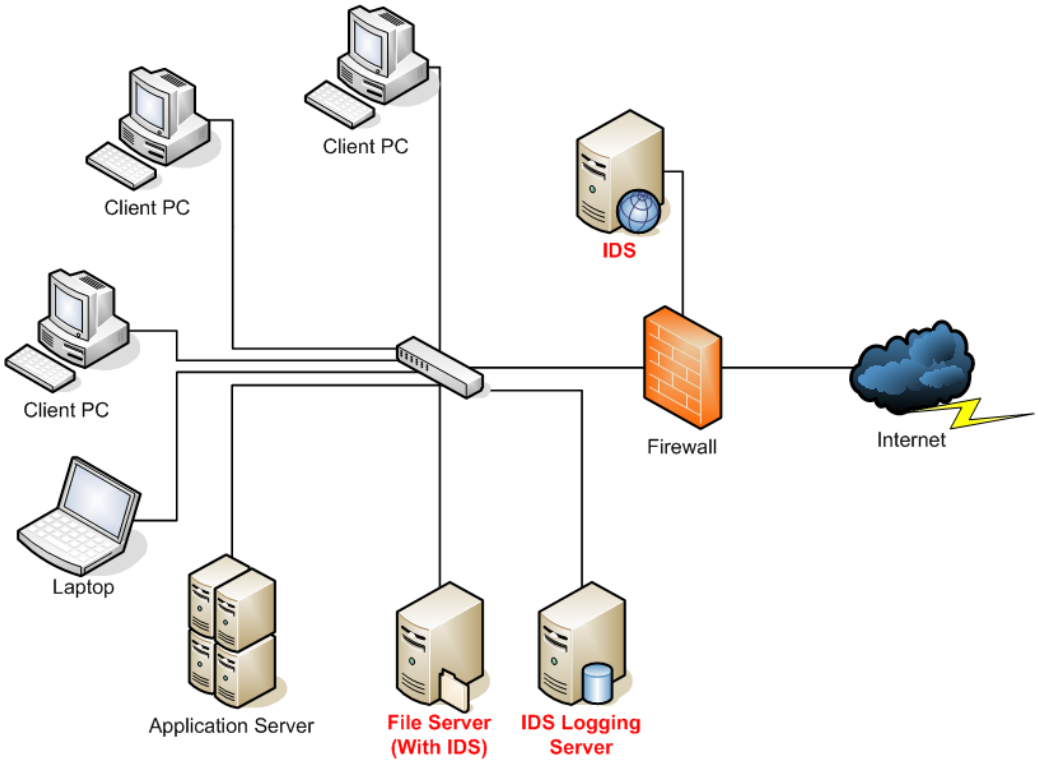

Figure 1: Deployment of an Intrusion Detection System

Two basic intrusion detection approaches have been established – misuse detection and anomaly detection. Misuse detection compares network activity to a database of known attack patterns. However, the primary disadvantage of this approach is that it cannot detect attacks that are not in the database – thus a new type of attack would be undetected by the IDS.

The second approach to intrusion detection is anomaly detection, which models normal network behavior and identifies attacks as deviations from the norm. In this way, anomaly detection can identify new, unknown types of attacks. However this technique also has an increased false alarm rate. It is important that IDS minimize its false alarm rate, as a system that generates too many false alarms cannot be relied upon, so this is an important criterion for evaluation of any anomaly detection technique.

The remainder of this paper evaluates the use of specific data mining approaches for anomaly detection.

1.2 Problem Description

In this paper we examine a new approach to anomaly detection – analysis of the counts of system calls (bag of system calls) made by network connections. These counts may then be used to classify connections as normal or intrusive. This approach differs from the traditional uses of system calls in anomaly detection, which is to analyze the sequence of system calls made and/or to examine the arguments made to these calls.

The bag of system calls approach differs significantly from traditional sequence of system calls approaches in that it has a simpler data representation, and lower memory requirements [3 – Kang et al]. In addition, the simpler representation allows many data mining techniques to be readily applied to its data sets. That is, a data set that contains a known set of attributes is much more suited to data mining than an approach that contains sequences of arbitrary length – data sets for a sequence of system calls approach would likely require significant preprocessing before data mining techniques could be applied to them.

The bag of system calls approach will then be compared to both traditional system call approaches. In addition, the bag of system calls approach will be compared with the analysis of network connection attributes – a data mining approach that uses three categories of features to classify connections. These features consist of basic network features, content features based on domain knowledge, and traffic features based on the state of other network connections within a two second sliding window. There are several advantages to this approach over the bag of system calls approach, including the potential for detection of malicious connections while the attack (connection) is still in progress.

Multiple approaches are included in the evaluation in order to determine how the analysis of system calls compares with other anomaly detection approaches for intrusion detection systems.

1.3 Approach

In order to evaluate the Bag of System Calls and Network Connection Attribute approaches for anomaly detection, we have used data derived from the DARPA 1998 IDS datasets compiled by MIT's Lincoln Laboratory. Actual data analysis was performed using the Weka Data Mining suite [7 – Weka].

The goal of our evaluation is to determine the usefulness of the Bag of System Calls approach to anomaly detection, as well and determining the relative strengths and

weaknesses of each approach. We will attempt to outline an overall approach to anomaly detection based upon each approach in our evaluation.

1.4 Organization

The remainder of the paper is organized as follows. **Section 2** presents an overview of the data sets we have selected for our evaluations. **Section 3** outlines the classification algorithms we have selected. **Section 4** discusses the tools and metrics we have used to perform our evaluation. **Section 5** presents the results of our evaluation experiments, including comparisons between the bag of system calls and network connection attributes techniques. In **Section 6** we conclude and present the results of our evaluation, including recommendations for future work. Finally, the remainder of the paper lists raw classification data, references, and team contributions to the project.

2 Data Sets

For our evaluations, we used data sets derived from the DARPA 1998 datasets compiled by MIT's Lincoln Laboratory and the Sequence Time-Delay Embedding (stide) datasets compiled by the University of New Mexico (UNM).

Data File	Size	Number of Instances	Source
Live lpr ARFF	836 KB	2,233	UNM's benchmark data
Live lpr MIT ARFF	1.34 MB	3,705	UNM's benchmark data
Synthetic sendmail ARFF	142 KB	371	UNM's benchmark data
Synthetic sendmail CERT ARFF	125 KB	328	UNM's benchmark data
Denial of Service (DoS) ARFF	4.51 MB	13,831	UNM's benchmark data
Monday, 4th week, 1998 ARFF	52.9 KB	228	MIT Lincoln Lab's DARPA evaluation data
Tuesday, 4th week, 1998 ARFF	97 KB	445	MIT Lincoln Lab's DARPA evaluation data
Thursday, 4th week, 1998 ARFF	80.5 KB	375	MIT Lincoln Lab's DARPA evaluation data
Friday, 4th week, 1998 ARFF	55.3 KB	251	MIT Lincoln Lab's DARPA evaluation data
KDD Network Connection Attributes	7.914 MB	65,536	1999 KDD Cup, Derived from DARPA 1998 Data Set

Table 1: Overview of Data Sets

Existing data sets were used for several reasons. Several researchers have published results using the DARPA datasets, thus providing a baseline for comparison of our results. These data sets were also chosen due to time constraints. The data sets have already undergone preprocessing, often the most time consuming phase of data mining. Additionally, the process of creating our own data set would be quite time consuming, as a network would have to be constructed and a series of attacks would have to be run against it over a period of days. Although impractical for our project, it would be worthwhile to generate data in this manner, in order to evaluate our techniques against a modern set of network data.

The DARPA datasets were compiled to simulate traffic over a nine-week period for a typical U.S Air Force network receiving multiple network-based attacks. A total of over four gigabytes of compressed TCP dump data was collected, consisting of around seven million connection records [6].

Attacks in the DARPA data fall into four main categories:

- Denial of Service
- Unauthorized access from a remote machine
- Unauthorized access to local super user (root) privileges
- Probing, including port scanning and related techniques

The Bag of System Calls data consists of a set of files in Weka ARFF format. The data is based upon a subset of the DARPA 1998 data, and has been preprocessed to transform it from network connection data to listings of calls made to the Operating System by applications handling these connections. Each file contains a set of numeric attributes for each network connection along with a single nominal attribute indicating if the connection is normal or intrusive. Each numeric attribute corresponds to the number of calls to that particular system call during the duration of the connection. The datasets are available online for further inspection, at [3 – Kang]. It is important to note that these data sets are much smaller than the full DARPA data, and contain only a few thousand instances each.

access	ioctl	putmsg
audit	kill	putmsg (connect)
auditon	link	readlink
chdir	login	rename
chmod	logout	rmdir
chown	lstat	setaudit
chroot	memcntl	setegid
close	mkdir	seteuid
creat	mmap	setgroups
exit	munmap	setpgrp
fchdir	old_nice	stat

fchown	old_setgid	statvfs
fcntl	old_setuid	su
fork	old_utime	sysinfo
fork1	open (read)	unlink
ftp_access	open (write)	vfork
getaudit	pathdonf	
getmsg	pipe	

Table 2: Partial List of System Calls Used for Anomaly Detection

The network connection attribute data consists of a set of CSV files that were converted to ARFF format for data analysis. These data files were compiled by [6 – UCI] for the 1999 KDD Cup competition, and are derived from the DARPA 1998 data. Each data instance consists of several attributes describing a single network connection, along with a label classifying the connection as normal or as one of several exploits.

The attributes used by the KDD data fall into three categories: Basic, Content, and Traffic features [6]. Basic features consist of typical features of a single network connection. Content features are domain-specific features that have been developed in an attempt to describe intrusive behavior – for example, the number of failed logon attempts. Finally, traffic features represent attributes of similar connections, using a sliding window of two seconds in length – for example, number of connections to the same service. A full list of attributes is shown below, summarized from the 1999 KDD Cup Description [6]:

Category	Feature Name	Feature Description
Basic	Duration	Length of the connection
Basic	Protocol type	Type of Protocol (TCP, UDP, etc)
Basic	Service	Network service (EG: SSH, SMTP, etc)
Basic	Source bytes	Number of data bytes from source → destination
Basic	Destination bytes	Number of data bytes from destination → source
Basic	Flag	Normal / Error status of connection
Basic	Land	Flag indicating if connection is from same host/port
Basic	Wrong fragment	Number of wrong fragments
Basic	Urgent	Number of urgent packets
Content	Hot	Number of "hot" indicators
Content	Num Failed Logins	Number of failed logins
Content	Logged in	1 if login successful, 0 else
Content	Num compromised	Number of "compromised" conditions
Content	Root shell	1 if root shell is obtained, 0 else
Content	Su attempted	1 if "su root" attempted, 0 else
Content	Num root	Number of root accesses
Content	Num file creations	Number of file creation ops
Content	Num shells	Number of shell prompts
Content	Num access files	Number of operations on access files
Content	Num outbound cmds	Number of outbound commands (FTP

Category	Feature Name	Feature Description
		only)
Content	Is hot login	1 if logon belongs to the "hot" list, 0 else
Content	Is guest login	1 if login is "guest", 0 else
Traffic	Count	Number of connections to the same host as current connection
Traffic (Same Host)	Serror rate	Percent of connections with SYN errors
Traffic (Same Host)	Rerror rate	Percent of connections with REJ errors
Traffic (Same Host)	Same srv rate	Percent of connections to the same service
Traffic (Same Host)	Diff srv rate	Percent of connections to different services
Traffic (Same Host)	Srv count	Number of connections to the same service as the current connection in past 2 seconds
Traffic (Same Host / Same Service)	Srv serror rate	Percent of connections with SYN errors
Traffic	Srv rerror rate	Percent of connections with REJ errors
(Same Host / Same Service)	Srv diff host rate	Percent of connections to different hosts

Table 3: Network Connection Attributes used by KDD Data

Finally, it should be noted that there are limitations to the data sets we have selected, including:

- The data is UNIX-only. This is not expected to be a significant limitation, and in fact could open up future possibilities for the analysis of windows-based networks.
- Data is based upon sample datasets that may not accurately reflect today's network/OS usage.

However, due to the difficulty in obtaining labeled IDS data these limitations are considered acceptable, and are not expected to significantly impact the results of our evaluation.

3 Algorithms

The following Data Mining algorithms were selected to analyze both IDS data sets:

1. **Nearest Neighbor (IB1)** – K = 1 nearest neighbor was chosen due to both its simplicity and success in analyzing data in the IDS application domain [5 – Lazarevic et al]. Additionally it is the simplest lazy (instance-based) classifier supported by the WEKA suite. Simple methods are often surprisingly effective for machine learning [8 – Witten], thus this algorithm seems like a good place to start for instance-based learning. The algorithm works by choosing the closest neighbor based upon Euclidian distance, and predicts the classification based upon this neighbor's class. Lazy (instance-based) classifiers are ideal for production IDS systems because updating their data model only requires the addition of new, labeled instances to their training data.

2. **Locally Weighted Learning (LWL)** – Locally weighted learning was chosen because it attempts to improve upon nearest neighbor by using local weights to aid in classification. In theory, this is an improvement over nearest neighbor because it is able to effectively distinguish between instances belonging to clusters of different densities [5 – Lazarevic et al]. This algorithm was selected in order to examine its performance with respect to nearest neighbor, and to determine if it is more suitable for use in anomaly detection. Finally, for our experiments we used the default DecisionStump classifier for this algorithm.

3. **RIPPER Rule Induction Algorithm** – The RIPPER (Repeated Incremental Pruning to Produce Error Reduction) rule induction algorithm was chosen because it is one of the most popular techniques used by Intrusion Detection Systems [2 – Kang et al]. This algorithm will, in particular, be evaluated against the Decision Table classifier.

4. **Simple Decision Table Majority Classifier** – A simple Decision Table classifier was chosen in order to measure the performance of a rule-based classifier that is less complex than RIPPER. Again, a simple algorithm was selected with the assumption that it can be surprisingly effective for machine learning.

5. **C4.5 Decision Tree Learner (J48)** – J48 was selected in order to consider a decision tree-based learning algorithm, and to compare the results of this classification technique to the more established rule and instance-based techniques in the IDS domain.

6. **K-Means** – The k-means algorithm was selected in order to evaluate the performance of a clustering algorithm for anomaly detection. In particular, we will consider proper settings for initial cluster size, and the practicality of choosing this algorithm for use in a real-world Intrusion Detection System.

4 Evaluation

4.1 WEKA – The Bird

4.1.1 WEKA – The Software

WEKA is a machine learning/data mining software package written in Java and distributed under the GNU Public License. It is used for research, education, and applications and complements the "Data Mining" text written by Witten & Frank [8].

The main features of this package include:

- Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
- Graphical user interfaces (incl. data visualization)
- Environment for comparing learning algorithms

This package also provides an extensive collection of Data Mining algorithms, tools, and visualizes that were invaluable for performing experiments and analysis during our evaluation. This allowed us to be more productive, as time did not need to be wasted on creating tools / techniques for performing these tasks.

4.1.2 WEKA – Versions

There are several versions of WEKA, including:

- WEKA 3.0: "book version" compatible with description in data mining book
- WEKA 3.2: "GUI version" adds graphical user interfaces (book version is command-line only)
- WEKA 3.3: "development version"
- WEKA 3.4: " new development version" with lots of improvements

Our evaluation is based upon the latest stable version of WEKA, 3.4.6

4.1.3 WEKA – Deals only with Flat files

The following is an example ARFF file, illustrating the generic ARFF file format:

```
@relation heart-disease-simplified
@attribute age numeric
@attribute sex { female, male}
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes}
@attribute class { present, not_present}
@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...
@relation heart-disease-simplified
@attribute age numeric
@attribute sex { female, male}
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes}
@attribute class { present, not_present}
@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
```

67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...

4.1.4 WEKA – GUI

4.2 Experimental Process

The Weka Experiment Environment was used to perform the majority of classifications. Unless otherwise noted, 10 fold cross validation was used in order to ensure each class was properly represented in the training/testing data. Additionally, each cross validation procedure was repeated 10 times for a total of 100 runs, in order to ensure that the final results contained a proper error estimate.

4.3 Evaluation Criteria

In evaluating the experimental results, two criteria are of primary importance – attack detection rate and false positive rate.

Attack detection rate indicates the percentage of attacks that are correctly identified by the algorithm. Note that this metric corresponds to the true positive rate.

$$AttackDetectionRate = \frac{NumberOfTruePositives}{NumberOfTruePositives + NumberOfFalseNegatives}$$

Equation 1: Attack Detection Rate

The false positive rate indicates the percentage of normal connections that are (incorrectly) classified as intrusive. This metric is extremely important to end users of IDS, and can be seen as a measure of the rate of false alarms the system generates. For example, consider IDS with a false positive rate of 1%, monitoring network traffic of a major university network generating an average of 7 million connection records per day. Such a system would generate over 70,000 false alarms every day, and would be useless to the network's system administrator.

$$FalsePositiveRate = \frac{NumberOfFalsePositives}{NumberOfFalsePositives + NumberOfTrueNegatives}$$

Equation 2: False Positive Rate

An additional evaluation criteria used in the IDS literature is to hold the level of false positives constant (for example, at 2%) and to only record attacks as detected until this false positive rate is reached. For our evaluation we did not use this evaluation method, although it might be worthwhile to consider for future evaluation of the Bag of System Calls approach.

Finally, we include an additional metric, percent correctly classified, in our result sets. This metric simply indicates the percentage of how many instances are correctly classified – as either normal or intrusive – and is of lesser importance than the other metrics.

5 Results

5.1.1 Experimental Results for Bag of System Calls

Based on the classification results we have compiled, it is clear that the Bag of System Calls approach is highly effective for attack detection. For each dataset the data mining algorithms were able to consistently detect over 97% of all attacks. Given the simplicity of the Bag of System calls approach, this figure is significant.

Although successful for attack detection, it can be seen that there is a high fluctuation in false positive rate between data sets for this approach. We initially thought this discrepancy could be due to the size of the data sets – for both UNM Sendmail data sets, the number of testing instances was less than one hundred. However, when the data sets were re-tested using fewer folds (and thus more training/testing instances), there was still a high rate of false positives. It is worth noting that the LPR and STIDE data sets consisted of much larger training and testing sets and mining of this data produced excellent detection rates with few false positives.

False positive rates were held below 2% for all data sets except the UNM Synthetic Sendmail and MIT Lincoln Lab Friday data. But why do these data sets have such higher false positive rates? Our team used Weka to extract the key attributes (that is, attributes that most affect the classification) from the UNM Synthetic Sendmail and UNM LPR MIT data sets. The average counts of these system calls are shown below. Note: The LPR MIT dataset also contained the read system call, but this was left out because of the high degree of variance between normal (average of 468 calls) and intrusive (average of 16 calls) connections.

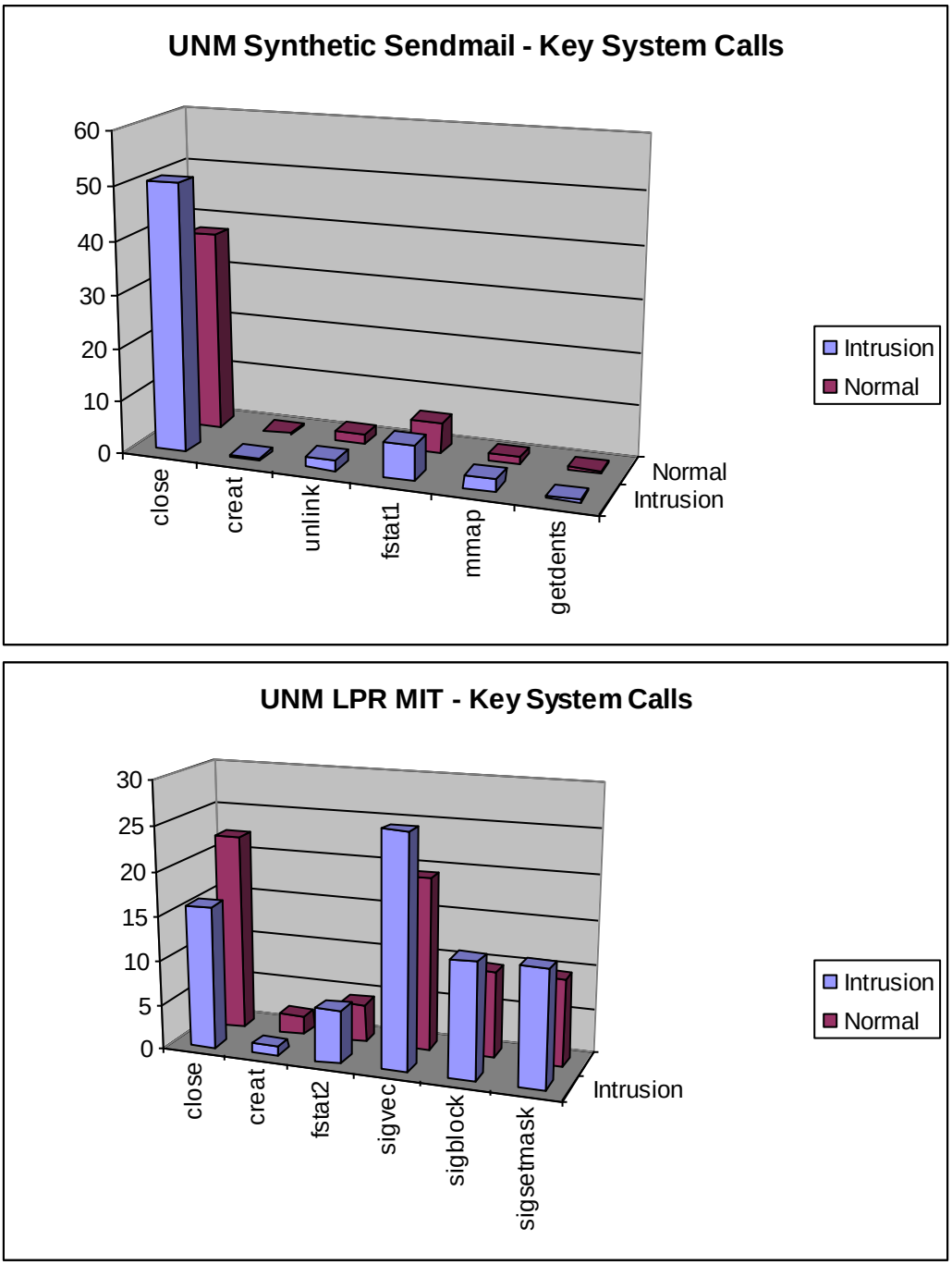

Figure 2: Comparison of Average Counts of Selected Attributes

As can be seen from the diagrams, each system call in the LPR MIT data set tends to have different average counts while there is not much difference between the average system call counts for the Sendmail data set. This may be why the Data Mining models generate so many false positives for the Sendmail data. In any case, it highlights an

important example of where the Bag of System Calls data does not provide enough data to properly classify connections.

Table 9 lists the selected attributes for each data set, as determined by the Weka SubsetEval attribute evaluator and BestFirst search method. Taking a closer look at the selected attributes for each data set, there is no specific pattern between all data sets. This is to be expected as each data set incorporates a different set of normal network traffic and patterns of attack. However, for the UNM data the open, close, and fstat (all variations) system calls are consistently relevant across data sets. In the DARPA data a single system call, audit, is all that is needed for classification. The following figures outline an example classification rule (Figure 3) and tree (Figure 4) generated by the bag of system calls classifiers. The simplicity of these models highlights a significant advantage the bag of system calls approach over more established system call based methods.

(unlink >= 4) and (getuid >= 2) and (read >= 16) => class = intrusion (1001.0/1.0)
=> class = normal (2704.0/1.0)

Figure 3: RIPPER Classification Rules for the UNM LPR MIT Data Set

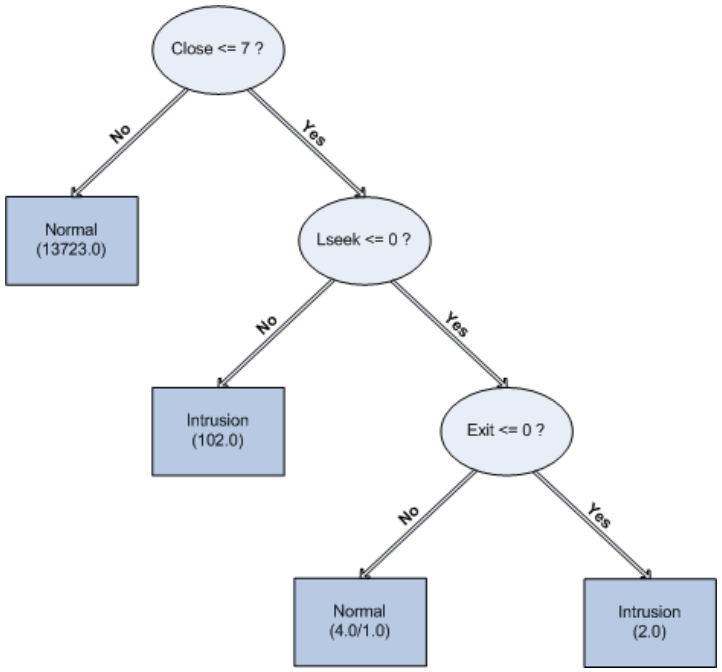

Figure 4: J48 Decision Tree for the UNM STIDE Data Set

It is interesting to note that the experimental results were similar for each algorithm. The LWL and RIPPER algorithms did not perform significantly better than their counterparts

for classification – and in some instances, the simpler algorithms actually performed better. For example, for classification of the UNM LPR MIT data set the IB1 algorithm had a slightly higher attack detection rate than LWL while still maintaining the same false positive rate.

In addition to the standard classification algorithms, the K-Means clustering algorithm was also used to perform classification experiments. The results for this algorithm are shown in Table 8. Overall the detection and false positive rates follow a similar pattern as for the other algorithms, with all data sets performing well except the UNM Synthetic Sendmail data sets. Nevertheless, there were some important differences in these metrics. The K-Means algorithm had detection rates that were, on average, 18.76 percent lower than those of the classification algorithms. Moreover, false positive rates were an average of 5.173 percent lower – a significant drop. Generally IDS algorithms have a tendency to generate fewer false positives as the detection rate decreases, so these results match our expectations for an anomaly detection algorithm with a noticeable decrease in detection rate.

An additional percent correctly classified metric has been included in our results for K-Means. Many data sets separated the IDS data into multiple clusters – however for each data set our classifier labeled only one cluster as normal and only one as intrusive. Thus, if the normal data was actually distributed over 4 clusters, any instances mapped to the other three clusters were marked as misclassified. This explains why the correctly classified metric was often so poor for the selected data sets, even while detection and false positive rates for the same data were excellent. For those data sets that separated into only two clusters, the percent correctly classified was steady at approximately 100%. It is interesting that normal or intrusive data would be separated over multiple clusters – this might be due to network data from multiple applications or protocols. All the data would receive the same label (normal) even though each protocol has an inherently different system call signature.

Since a real-world IDS has strict time constraints, we shall briefly discuss the time requirements for classification. Training and testing times for analysis of the UNM Stide Data Set are indicated below in Table 4. This data set was chosen because it is the largest Bag of System Calls Data set, with 164 attributes and almost 14,000 instances. These times were calculated by taking the average time of each algorithm's test runs. All test runs for this data set were performed on a 1.75 GHz Pentium M processor with 512 MB of RAM.

Algorithm	Training Time	Testing Time
weka.classifiers.trees.J48	3.2762	0.0076
weka.classifiers.rules.DecisionTable	32.3737	0.0276
weka.classifiers.rules.JRip	8.6975	0.0087
weka.classifiers.lazy.IB1	0.1326	401.2664
weka.classifiers.lazy.LWL	0.1165	14968.4537

Table 4: Running Times for Classification of UNM STIDE Data Set

As you can see, the lazy (instance-based) algorithms had a negligible training time as well as significant testing times. This is to be expected, as these algorithms perform all of their work at classification time. However, this suggests that lazy algorithms may in general be unsuitable for use in IDS, where the extra overhead may be a significant factor – or at the very least, that one must be very careful when selecting an instance-based algorithm. For example, the LWL algorithm was tested on an average of 1383 testing instances. This means that, using the testing time above, it would take an average of 10.8 seconds to classify a single instance – this is simply impractical for a system that is expected to process millions of connections per day.

On the other hand, the tree and rule-based algorithms had a noticeable training time but a negligible testing time. In particular, the J48 algorithm has an excellent balance of training and testing time, with the lowest times for all non-instance-based algorithms sampled.

5.1.2 Experimental Results for Network Connection Attributes

Experimental data for network connection attributes contained a subset of 65536 instances from the full KDD dataset, thus allowing training sets of approximately 50,000 instances and testing sets of approximately 16,000 instances when using 10-fold cross validation. These records contained several instances of normal connections, interspersed with Neptune and Smurf network attacks. Also, it should be noted that IB1 was tested with 4 folds and 2 runs, and LWL was tested with 10 folds and 1 run, both due to the extreme length of time both algorithms took to process the subset of KDD data.

Experimental results for this approach were also very positive – attack detection rates for each algorithm were over 99%, while false positive rates were kept under two-tenths of a percent for each algorithm surveyed. In fact, false positive rates were significantly better for this technique than the bag of system calls representation.

The following table lists attributes selected by Weka as the most important for classification. Of the key attributes listed in the table, half are traffic-based features. This might help to explain why this approach has a lower false positive rate than the Bag of System Calls method. Attacks often occur in bursts [5 – Lazarevic et al]. These traffic features allow the classifier to consider other connections occurring within a two-second sliding window while performing classification for a single instance. This may help the classifier to properly identify attack bursts. The Bag of System Calls data only contains information for the current connection and thus would have a limited ability to classify attack bursts.

protocol_type	hot	diff_srv_rate
service	logged_in	dst_host_same_src_port_rate
src_bytes	count	dst_host_srv_diff_host_rate
dst_bytes	serror_rate	dst_host_srv_serror_rate

Table 5: Selected Attributes for Classification of Network Connection Data

Overall each algorithm seemed to have similar results, although the rule-based Decision Table and RIPPER rule induction ultimately had the best balance between detection rate and false alarm rate. However, the LWL classifier performed poorly in comparison to the other algorithms. Although LWL had excellent attack detection rate of over 99%, its false alarm rate of 9% was almost 3 orders of magnitude higher than any of the other classifiers. These false alarms could be due to its high detection rate, although the other algorithms were able to maintain a similar detection rate with much lower false alarm rates.

Due to time constraints, results are not available for evaluation of the K-Means technique on this data set. Thus results for this algorithm cannot be compared between this technique and the bag of system calls technique. This is not considered a serious impact to our evaluation, as the results of K-Means for the bag of system calls dataset were valuable for analysis of that dataset, and provided enough conclusive information to suggest that the K-Means algorithm does not have a significant advantage over other techniques.

In order to compare the complexity of data models generated using this technique with those from the bag of system calls, a RIPPER rule set and J48 decision tree were generated from this data set. The RIPPER algorithm generated a total of 19 rules, illustrated below in Figure 5, compared to the 2 rules generated by RIPPER for the bag of system calls UNM LPR MIT data set. A J48 tree created during classification of the KDD data set consisted of 101 leaves and 134 total nodes. Due to the unwieldy size of the tree, it is not illustrated in this paper.

It is clear that the data models for the KDD Network Connection Attributes technique are much more complex than those for the bag of system calls. Each is over an order of magnitude larger in terms of number of rules/nodes used. Although the KDD technique was more effective for classification, it is important to note the substantial increase in complexity.

(num_compromised >= 2) and (dst_host_count <= 2) => label=buffer_overflow. (3.0/0.0)
(dst_host_srv_count <= 2) and (dst_host_diff_srv_rate <= 0.01) and (service = ftp_data) => label=multihop. (2.0/0.0)
(duration >= 179) and (dst_host_srv_count <= 2) => label=multihop. (2.0/0.0)

(dst_host_count <= 2) and (num_access_files >= 1) => label=ftp_write. (3.0/0.0)
(wrong_fragment >= 1) and (protocol_type = icmp) => label=pod. (20.0/0.0)
(diff_srv_rate >= 0.36) and (src_bytes <= 0) => label=portsweep. (40.0/1.0)
(num_failed_logins >= 1) and (dst_bytes <= 179) => label=guess_passwd. (51.0/0.0)
(wrong_fragment >= 1) => label=teardrop. (99.0/0.0)
(flag: = SH) => label=nmap. (102.0/0.0)
(protocol_type = udp) and (service = private) => label=nmap. (26.0/1.0)
(dst_host_srv_diff_host_rate >= 0.5) and (protocol_type = icmp) => label=ipsweep. (563.0/0.0)
(dst_host_diff_srv_rate >= 1) and (src_bytes <= 0) => label=ipsweep. (91.0/0.0)
(dst_host_count <= 2) and (service = ftp_data) and (flag: = REJ) => label=ipsweep. (3.0/0.0)
(src_bytes >= 29200) and (service = http) => label=back. (1995.0/0.0)
(flag: = RSTR) => label=back. (8.0/1.0)
(count >= 334) => label=smurf. (11206.0/0.0)
(service = ecr_i) and (src_bytes >= 1032) => label=smurf. (52.0/0.0)
(dst_host_srv_serror_rate >= 0.83) and (flag: = S0) => label=neptune. (11955.0/0.0)
=> label=normal. (39315.0/19.0)

Figure 5: RIPPER Classification Rules for KDD Network Connection Data Set

6 Conclusion

6.1 Evaluation of Selected Algorithms

All algorithms performed roughly the same for classification of IDS data, with no clearly superior algorithm. However, the LWL algorithm did perform significantly worse that the other algorithms for certain data sets and thus can not be recommended for use in this domain. The nearest neighbor algorithm appeared fairly stable, with slightly lower false positive rates than other algorithms for “trouble” data sets such as UNM Sendmail with a 23% false positive rate compared to the next-closest rate of 51.5% -- although both figures are still unacceptable for a production system.

Although it cannot be seen by looking exclusively at the classification algorithm results, the detection rate and false positive rate are in general inversely proportional, as can be discerned by comparing the results of the K-Means clustering algorithm with those of the classification algorithms. This means that such an algorithm could be useful for a certain production systems, where a lower detection rate might be acceptable if it also decreases the false positive rate. For example, a university network may consider the lower false positive rate to be an acceptable tradeoff, whereas the tradeoff may not be acceptable for an FBI or military network that has strong security requirements.

For bag sys calls often a small number of attributes – often only a single one – may be used to perform the majority of classification while the network connection attributes technique uses a much larger number of attributes for classification. This may be one of the reasons why the connection attributes approach was more stable in testing, as more variations between attributes are required in order to result in a misclassified instance.

6.2 Evaluation of IDS Techniques

Although the Bag of System Calls approach appears promising for anomaly detection, there are issues that must be dealt with before it can be incorporated into a production IDS. As mentioned previously, one of the reasons the Network Connection Attributes approach had fewer false positives than the Bag of System calls approach is that it considers other connections to the host. Thus the approach is better able to detect attack bursts. The Bag of System calls features would likely be more effective for anomaly detection if they were augmented with additional features such as the traffic-based features from the KDD data set.

An attack that was not considered in the evaluation was that of root kits. Essentially these techniques would replace system processes with those that have been modified to hide the actions of an attacker. It is likely that the bag of system calls approach would be effective against this type of attack since a patched process would likely have a different signature of system calls than the original process. This is an area for further evaluation, as it would clearly demonstrate an effective and novel contribution of this approach.

Also, with only a few thousand instances in our data sets, there may not have been enough data to properly evaluate the Bag of System Calls method. It would be worthwhile to repeat the evaluation of this approach using larger datasets. Ideally, two versions of these data sets would be created – one with only the system calls and another augmented with select network connection features. An evaluation of both of these data sets could lead to a more robust approach to anomaly detection. Also, by pruning the attributes from such a set, the IDS might be able to function effectively while examining a smaller subset of the system calls and network connection attributes of existing connections.

Given the successful results demonstrated by both the bag of system calls and the network connection attributes approaches, it is likely that each approach could be successfully integrated into a real world IDS. However, more testing is required in order to determine the performance of either approach on real network data. It is unlikely that either technique would approach the significant detection / false positive rates that they achieved on the DARPA and UNM datasets.

Nevertheless, it is clear that the Bag of System Calls technique is highly effective for attack detection over most data sets used for our evaluation. Given the simplicity of the approach, this is significant as these features should be able to be incorporated into IDS without significant implementation effort. In the end, this would help to make a production IDS more robust to newer types of attacks, both local and remote.

Classification Results

Data Set	Algorithm	Correctly Classified (%)	Attack Detection Rate (%)	False Positive Rate (%)
UNM LPR	J48	99.910	100.000	0.200
UNM LPR	Decision Table	99.910	100.000	0.200
UNM LPR	JRip	99.906	99.992	0.200
UNM LPR	Nearest Neighbor	100.000	100.000	0.000
UNM LPR	LWL	99.910	100.000	0.200
UNM LPR (MIT)	J48	99.873	99.863	0.100
UNM LPR (MIT)	Decision Table	99.919	99.963	0.200
UNM LPR (MIT)	JRip	99.906	99.919	0.100
UNM LPR (MIT)	Nearest Neighbor	99.922	99.967	0.200
UNM LPR (MIT)	LWL	99.457	99.331	0.200
UNM Synthetic Sendmail	J48	95.904	99.396	51.700
UNM Synthetic Sendmail	Decision Table	96.145	99.568	51.500
UNM Synthetic Sendmail	JRip	94.746	98.871	62.000
UNM Synthetic Sendmail	Nearest Neighbor	95.982	97.371	23.000
UNM Synthetic Sendmail	LWL	95.312	99.971	69.200
UNM Synthetic Sendmail CERT	J48	97.527	99.286	17.900
UNM Synthetic Sendmail CERT	Decision Table	96.314	98.538	23.200
UNM Synthetic Sendmail CERT	JRip	96.678	98.568	19.800
UNM Synthetic Sendmail CERT	Nearest Neighbor	97.775	98.843	11.500
UNM Synthetic Sendmail CERT	LWL	93.509	99.591	58.600
UNM Stide	J48	99.973	99.979	0.900
UNM Stide	Decision Table	99.979	99.988	1.200
UNM Stide	JRip	99.975	99.980	0.600
UNM Stide	Nearest Neighbor	99.980	99.993	1.700
UNM Stide	LWL	99.970	99.969	0.000
DARPA 98, Week 4, Monday	J48	100.000	100.000	0.000
DARPA 98, Week 4, Monday	Decision Table	100.000	100.000	0.000
DARPA 98, Week 4, Monday	JRip	100.000	100.000	0.000
DARPA 98, Week 4, Monday	Nearest Neighbor	100.000	100.000	0.000
DARPA 98, Week 4, Monday	LWL	100.000	100.000	0.000
DARPA 98, Week 4, Tuesday	J48	99.551	100.000	1.400
DARPA 98, Week 4, Tuesday	Decision Table	99.551	100.000	1.400
DARPA 98, Week 4, Tuesday	JRip	99.551	100.000	1.400
DARPA 98, Week 4, Tuesday	Nearest Neighbor	99.262	99.567	1.400

Data Set	Algorithm	Correctly Classified (%)	Attack Detection Rate (%)	False Positive Rate (%)
DARPA 98, Week 4, Tuesday	LWL	99.416	100.000	1.800
DARPA 98, Week 4, Thursday	J48	99.733	99.580	0.000
DARPA 98, Week 4, Thursday	Decision Table	99.733	99.580	0.000
DARPA 98, Week 4, Thursday	JRip	99.733	99.580	0.000
DARPA 98, Week 4, Thursday	Nearest Neighbor	99.733	99.580	0.000
DARPA 98, Week 4, Thursday	LWL	99.733	99.580	0.000
DARPA 98, Week 4, Friday	J48	98.806	100.000	10.800
DARPA 98, Week 4, Friday	Decision Table	98.806	100.000	10.800
DARPA 98, Week 4, Friday	JRip	98.806	100.000	10.800
DARPA 98, Week 4, Friday	Nearest Neighbor	96.898	97.789	10.000
DARPA 98, Week 4, Friday	LWL	98.806	100.000	10.800

Table 6: Classification Results for Bag of System Calls

Data Set	Algorithm	Correctly Classified (%)	Attack Detection Rate (%)	False Positive Rate (%)
DARPA 99	J48	99.889	99.917	0.0063
DARPA 99	Decision Table	99.898	99.833	0.0010
DARPA 99	JRip	99.924	99.917	0.0016
DARPA 99	Nearest Neighbor*	99.918	99.625	0.0134
DARPA 99	LWL*	95.036	99.107	9.0000

Table 7: Classification Results for Network Connection Attributes

Data Set	Number of Clusters	Correctly Classified (%)	Detection Rate (%)	False Positive Rate (%)
UNM LPR	2	99.955	99.900	0.000
UNM LPR MIT	6	55.951	99.800	0.407
UNM Synthetic Sendmail	4	36.388	40.000	16.760
UNM Synthetic Sendmail CERT	3	62.500	23.529	17.007
UNM STIDE	6	57.060	100.000	0.350
DARPA 1998 Monday	2	100.000	100.000	0.000
DARPA 1998 Tuesday	9	56.405	75.524	0.331
DARPA 1998 Thursday	4	67.467	100.000	2.110
DARPA 1998 Friday	3	67.331	89.286	0.000

Data Set	Number of Clusters	Correctly Classified (%)	Detection Rate (%)	False Positive Rate (%)
KDD Network Connection Attributes	-	-	-	-

Table 8: Classification Results for K-Means Clustering Algorithm

Data Set	Selected Attributes
UNM LPR	fstat2 sigvec sigblock sigsetmask
UNM LPR MIT	read close creat fstat2 sigvec sigblock sigsetmask
UNM Synthetic Sendmail	close creat unlink fstat1 mmap getdents
UNM Synthetic Sendmail CERT	open link execve accept sigvec
UNM STIDE	close mmap munmap fstat
DARPA 1998 Week 4	audit

Table 9: Selected Attributes for Bag of System Calls Data Sets

References

1. Gaurav Tandon and Philip Chan, Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL 32901, Learning Rules from System Call Arguments and Sequences for Anomaly Detection, http://www.cs.fit.edu/~pkc/papers/dmsec03tandon.pdf, 2003.

2. Dae-Ki Kang, Doug Fuller, and Vasant Honavar, Artificial Intelligence Lab, Department of Computer Science, Iowa State University, Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation, http://archives.cs.iastate.edu/documents/disk0/00/00/03/59/00000359-00/isi05.pdf , 2005.

3. Dae-Ki Kang, Benchmark Datasets for Intrusion Detection System in Bag of System Calls Representation, http://www.cs.iastate.edu/~dkkang/IDS_Bag/, March 7, 2005.

4. MIT Lincoln Laboratory, DARPA Intrusion Detection Evaluation Data Sets, http://www.ll.mit.edu/IST/ideval/data/data_index.html, September 18, 2001.

5. Aleksandar Lazarevic, Aysel Ozgur, Levent Ertoz, Jaideep Srivastava, and Vipin Kumar, Department of Computer Science, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455, A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection, http://www.cs.fit.edu/~pkc/id/related/lazarevic03sdm.pdf, 2003.

6. Bren School of Information and Computer Sciences, University of California, Irvine, The UCI KDD Archive, KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 28, 1999.

7. University of Waikato, Weka 3 - Data Mining with Open Source Machine Learning Software in Java, http://www.cs.waikato.ac.nz/ml/weka/, October 15, 2005.

8. Ian H. Witten & Eibe Frank, "Data Mining: Practical Machine Learning Tools and Techniques", 2nd Edition, Morgan Kaufmann Publishers, San Francisco, 2005.

Project Contributions

Member	Contributions	Percentage
Justin Ethier	• Submitted Project Proposal • Performed research into the IDS topic • Obtained DARPA, KDD, and Bag of System Calls data • Converted KDD '99 (Network Connection Attributes) data set to ARFF Format • Ran Weka Experiments • Wrote first draft of the Project Paper • Edited all sections of the project paper • Planning to participate in final presentation	40%
Sanjay Srivastava	• Performed research into the IDS topic • Organized Group Meetings • Set Deliverable Dates • Ran Weka Experiments • Assisted in first draft of project paper • Edited all sections of the project paper • Created Presentation Slides • Planning to participate in final presentation	40%
Theodore Sadler	• Ran Weka Experiments • Edited sections of the project paper, added abstract and organization sections • Planning to participate in final presentation	20%