An Evaluation of Size-based Traffic Feature for Intrusion Detection

Ming Ye

G. Premkumar

Dan Zhu

College of Business
Iowa State University
Ames, IA 50011

Abstract

Network attacks have become a significant threat to organizations and effective intrusion detection systems have to be developed detect these attacks before they inflict harm to the internal network infrastructure. Denial of service (DoS) and probing attacks are the most common attacks. While time-based traffic features provide information to identify attacks, size-based traffic features enhance the identification accuracy. In this study, we add a size-based feature to an existing time-based feature intrusion detection system. The system is tested on a data set that includes both normal traffic and attack traffic from different types of attacks. The results indicate that size-based feature increases the accuracy of prediction. We also used meta-classification schemes such as bagging and boosting to examine if they improve the perfor-

mance. The improvement in accuracy was only marginal compared to the combined model that includes both time-based and size-based features.

Keywords: Intrusion detection, network security, data mining, network attacks, induction tree algorithm

Introduction

Information technology has dramatically changed our lives, but has also made us very dependent on a technology that is vulnerable to attacks. Network security has become a key issue for most organizations as extensive waves of attacks and intrusions on servers and client machines are becoming common place (Durst et al. 1999). Intrusions are any set of actions that attempt to compromise the integrity, confidentiality, or availability of network resource. Organizations take various actions to prevent or reduce intrusions into their networks without compromising on the basic objective of networks, i.e. facilitating communication among users. Various techniques such as user authentication, information protection through encryption, physical security, defensive programming techniques, training of users on basic security awareness, and intrusion detection are used to create various defense mechanisms to reduce the impact of attacks on network resources.

An important tool available to the network administrator is an intrusion detection system (IDS) that acts as an early warning system that detects intrusions. They serve an important function in the network infrastructure acting as gatekeepers and checking for suspicious activity. They need to be adaptive and learn about new attack techniques as they occur and respond to them. Also, IDS have to be accurate since they have to detect attacks without raising many false alarms that could become an impediment to the actual functioning of the network.

Intrusion detection systems (IDS) can be located on the host or on the network (Lippmann et al., 2000a, 2000b). Network based intrusion detection systems, unlike host based systems, examine all the packets traveling the network to identify possible intrusions. The typical attacks detected by network IDS are denial of service (DOS) and probing attacks. These attacks are normally synchronized and therefore have certain unique time-based features. Lee and Stolfo (2000) describe the various time-based features of network traffic that can be monitored to detect an attack. They have a threshold "count" of number

of connections in a fixed time period (2 secs) to determine if an attack is imminent. However, the normal traffic can widely vary during different times of the day and during different days of the week, to raise some false alarms. A size-based feature that incorporates specific character-istics of the attack packets can help to improve the prediction accuracy. Hence, the major objectives of this study are:

- To develop an IDS with "size-based" traffic feature
- To compare the effectiveness of time-based and size-based IDS on actual intrusion data

The paper is organized in eight sections. Section two provides a brief background of research in IDS; sections three and four describe the time-based features and size based features respectively; section five describes the data mining algorithms; section six discusses the research experiment and section seven discusses the results; the last section presents a summary of our findings, limitations, and research implications.

Background

Research on IDS has been extensive starting from a seminal paper on intrusion detection by Denning (1987). Much of the research on IDS is based on tools that incorporate new ideas for detection based on the attack characteristics. An extensive survey of research on IDS is available in Allen et al., (2000). Intrusion detection, based on their detection method, can be broadly divided into two categories: misuse detection and anomaly detection. Misuse detection system detects attacks based on well-known vulnerabilities and intrusions stored in a database (a.k.a. signatures), while anomaly detection system detects deviations in activity from normal profiles. Misuse detection systems such as EMERALD (Porras & Neumann, 1997), TIM (Teng et al. 1990), IDIOT (Kumar and Spafford, 1995), use patterns or signatures of well-known attacks to match and identify intrusions. Various techniques including rule-based expert systems, model-based reasoning systems, state transition analysis, genetic algorithm, and fuzzy logic have been used for detection (Carrettoni et al., 1991, Garvey & Lunt, 1991, Ilgun et al. 1995, Porras & Valdes, 1998). A good example would be failure of account login for more than 3 times due to wrong password. The intrusion signature (login fails three times) is matched with the audit data, and then the detector flags intrusion and sets off an alarm. Although misuse detection allows sensors to collect a more tightly targeted set of system data, thereby reducing the system overhead, its

inability to detect novel attacks, which do not have matched patterns or signatures in IDS's database constrains its effectiveness.

Anomaly detection system detects deviations from normal behavior, and based on a threshold value determines if it is normal or abnormal behavior. Although the knowledge of the pattern of a specific attack is not required, it needs to establish a normal usage profile to compare. For example, the normal usage profile may contain the average frequency of certain system call a user uses in a session. If the frequency is either too high or too low then an anomaly alarm is triggered. However an anomaly detection system tends to generate more false alarms, because it cannot correctly classify a new normal behavior. There are various approaches for anomaly detection including statistical analysis, sequence analysis, neural networks, machine learning, and artificial immune system (Bonifacio et al. 1997; Lee et al., 1998) that tries to identify patterns from large volumes of data to detect attacks. Since intrusions are normally based on a series of activities, sequence analysis, which examines a sequence of calls or activities rather than a single call, works better in detecting intrusions (Hofmeyr et al., 1998). Modern intrusion detection systems combine both approaches to identify intrusions (Anderson et al., 1995).

Since IDS is essentially a warning system, accuracy is critical to ensure all intrusions are identified without raising too many false alarms. Extensibility and adaptability are also critical in today's network computing environment. There are multiple "penetration points" that can be used to attack a network system including network devices such as routers and switches, servers, clients, and the general network. For example, at the network level, a malicious IP packet can crash a host; at the host level, vulnerabilities in system software can be exploited to yield an illegal root shell. Since activities at different levels are usually recorded in different audit data, an IDS often needs to be extended to include modules that incorporate the specialized knowledge of intrusion on all these attack points. They also need to be adaptable so that new rules and features are easily added as new attacks are identified. Building an effective IDS is an enormous knowledge engineering task. Although modern techniques like data mining can help to generate rules and create patterns in intrusion detection, system builders' intuition and experience are still important. Experts first

analyze and categorize attack scenarios and system vulnerabilities, and then hand-code the corresponding rules and patterns.

Time based traffic features

Intrusion detection involves analyzing audit or traffic data to recognize patterns that relate to intrusive behavior. Much of research on anomaly detection in IDS have focused on improving the accuracy of prediction by incorporating features that better describe the attacks and efficient algorithms to examine the large volumes of data. Lee and Stolfo (2000) describe an IDS that uses a time-based traffic feature to recognize attacks. Their study examines the TCPDUMP data for patterns to identify potential attacks. For example, in SYN flooding many machines with spoofed addresses attack a host, generating similar packets to the same destination. Denial of Service (DoS) and Probing attacks establish multiple connections to the same host and/or same port in a very short period of time and therefore exhibit packet traffic patterns that can be easily identified. A count of the number of packets within a certain time interval would be able to differentiate a normal traffic pattern of clients requesting connection to a server versus an attack. The time based feature provides warning signs for various attacks including SYN flooding, denial of service (DOS) attack, and probing attack. It had been extensively tested by Lee and Stolfo (2000). As our study builds on these features we present a brief summary of their ideas. The attacks are classified by them into 4 categories:

- DoS - denial-of-service. e.g.: teardrop, back, smurf, etc.;
- R2L - remote-to-local attacks, which involve unauthorized access from a remote machine. e.g: guess_passwd, rootkit, spy, etc.;
- U2R - user-top-root attacks, which involve unauthorized access to local super user privileges by a local unprivileged user. e.g: buffer_overflow, loadmodule, perl;
- Probing, surveillance and probing. e.g.: satan, nmap, ipsweep.

DOS and Probing attacks have sequential patterns for short time spans and therefore ideally suited for detection using "time-based traffic" feature. R2L and U2R do not have sequential patterns that are different from normal traffic. These attacks are embedded in the data portion of the packets and normally carried out on a single connection. Hence, pure time-based traffic feature may not be adequate and other approaches are required to detect such attacks. Lee and Stolfo (2000) developed "content-based traffic" features to detect R2L and U2R attacks.

TCPDUMP data provides packet level information that needs to be pre-processed for data mining. Lee and Stolfo (2000) explain how the raw TCPDUMP data is processed into network connection records using a pre-processing program that captures the important features of the connection (Paxson 1998). Network connection records that contain information on source and destination host *(src-host, dst_host)*, source and destination ports *(src_port, dst_port)*, data bytes sent and received *(src_bytes, dst_bytes)*, length of connection (duration), type of protocol *(protocol type),* type of service *(service)*, and an indicator of error status (flag).

The network connection records are then analyzed and summarized. First the "same host" feature examines the connections in the past 2 seconds that have the same destination host as the current connection, and various statistics are calculated including a count of connections, percentage of connections that have the same service and different service as current one, percentage of SYN errors, and percentage of REJ errors. Next, the 'same service' feature examines connections in the past 2 seconds that have the same service as the current connection and all statistics are calculated. The time-based traffic features that are calculated by Lee and Stolfo (2000) are shown in Table 1.

Feature	Description	Value Type
Count	Number of connections to the same host as the current connections in the past 2 seconds	Continuous
	The following features refer to these same-host connections	
serror_%	% of connections that have "SYN" errors	Continuous
rerror_%	% of connections that have "REJ" errors	Continuous
same_srv_%	% of connections to the same service	Continuous
Diff_srv_%	% of connections to different services	Continuous
Srv_count	Number of connections to the same service as the current connection in the past 2 seconds	Continuous
	the following features refer to same-service connections	
Srv_serror_%	% of connections that have "SYN" errors	Continuous
Srv_rerror_%	% of connections that have "REJ" errors	Continuous
Srv_diff_host_%	% of connections to different hosts	Continuous

Table 1. Traffic Features of Network Connection Records
(Lee and Stolfo, 2000)

Packet size based traffic feature

A careful examination of raw TCPDUMP records during DoS and probing attacks reveal that these attacks have two common characteristics. They are:

-        In a sequence of connections records, the connections involving a DoS or Probing attack are continuous;
-        The connection records in a DoS or Probing attack have the same packet size for outgoing and incoming frames.

Typically, DoS and Probing attacks send packets of the same length to the victim host, and receive packets of the same length from the victim. These attack connections are more likely to appear together in a sequence without the interference of the normal connections. On the other hand, normal connections usually do not have the same packet size in a long connection sequence. Hence, the "size-based traffic" feature can be added to the "time-based traffic" feature to improve the accuracy of intrusion detection. If a sequence of connections has the same packet size for packets received and sent (src_bytes and dst_bytes), it could indicate with a high probability of an occurrence of a DoS or Probing attack.

Determination of Window Size

Since the system needs to assess a sequence of records to determine if there is an attack, a sliding window is used to identify a set of connection records for analysis. The window size is defined by the number of connection records. In a continuous sequence, if there are n or more connections that have the same src_bytes and dst_bytes, then all these connections are labeled as 1 - "possible attack". Otherwise, these connections are labeled as 0 - "possible normal". For illustration, consider a continuous sequence of connection records, as shown in Table 2.

In this example records 3 through 20 have the same src_bytes and dst_bytes. The algorithm for a window size of 3 (n = 3) is as follows. First compare record 1 and record 2. Since these two records have different src_bytes and/or dst_bytes, label record 1 as 0, but keep record 2 unlabeled. Then compare record 2 and record 3. Label record 2 as 0, and keep record 3 unlabeled. In the next step, we find record 3 and record 4 have the same src_bytes and dst_bytes. At this time, since the window size is 3, we keep both record 3 and record 4 unlabeled and go to compare record 4 and record 5. Since record 4 and record 5 have

the same src_bytes and dst_bytes, we can label all these 3 records as 1. Then we check record 6, since it has the same byte feature as record 5, it should also be labeled as 1. Similarly, all the following connections through record 20 should be also labeled as 1. After that, record 21 should be labeled as 0 again. If record 5 has different src_bytes and/or dst_bytes, record 3 and record 4 should also be labeled as 0. Since there are no previous connections to compare with record 1, the label for record 1 may not be correct. Hence, this connection record is not considered for further analysis.

serial #	…	…	src_bytes	dst_bytes	…	Label
1	…	…	100	200	…	0
2	…	…	200	300	…	0
3	…	…	200	400	…	1
4	…	…	200	400	…	1
5	…	…	200	400	…	1
…	…	…	200	400	…	1
20	…	…	200	400	…	1
21	…	…	120	150	…	0
…	…	…	…	…	…	…

Table 2. A Sequence of Sample Network Connection Records

There are two traffic scenarios that need to be considered in determining the appropriate window size. Since we are collecting data in a network environment it is possible that some normal connections interfere in a sequence of continuous attack connections, and divide the attack sequence into several sub-sequences. In each sub-sequence, attack connections are still continuous. There are two possible cases in this scenario.

Case 1: The size of each sub-sequence is big enough - window size is smaller than or equal to the size of the attack sub-sequence, but bigger than the normal sequence. In this case the "interference" of normal sequences will not affect the accuracy of intrusion detection as the attack can still be detected. As shown in Figure 2 a sequence of attack connections is separated by a sequence of normal connections (b) into

two sub-sequences (a) and (c). Connections in (a) and (c) can be correctly labeled if the sequence size is not less than the window size. Connections in (b) can also be correctly labeled if its sequence size is less than the window size.
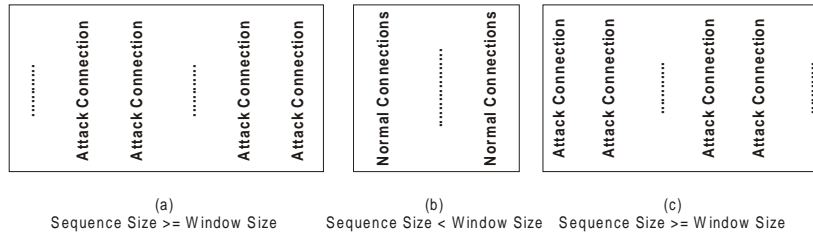

(a)
Sequence Size >= Window Size
(b)
Sequence Size < Window Size
(c)
Sequence Size >= Window Size

Figure 2 : Interference of normal connections, case 1

Alternatively, as shown in Figure 3, the normal connections are inserted into the attack sequence in such a way that splits the attack connections into many small sub-sequences that is discretely distributed in the sequence. In this case the window size may be too big for each sub-sequence leading to incorrect labeling of attacks. In this case both "time-based traffic" features and "size-based traffic" feature cannot easily detect the intrusions.
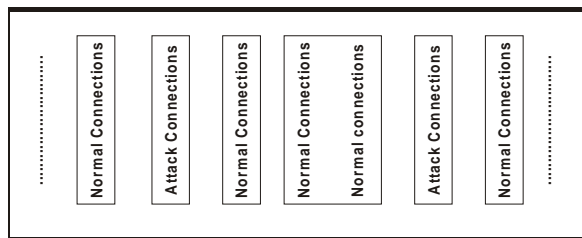

Figure 2 : Interference of normal connections, case 2

Since the connection data is collected at the network level rather than at the host level, a continuous sequence of network connections with the same connection size may be for different destinations. If the sequence size is smaller than the window size it will be correctly labeled. If the sequence size is large, either due to large number of destination and/or source sites, then mere evaluation of source and destination packet size may lead to wrong inference. It may be necessary to combine it with other "time-based traffic" features, which are calculated based on the same-host and same-service counts.

Window size is an important variable in determining the accuracy of an IDS. It is possible that by selecting a wrong window size we may label a normal connection as an attack (false positive) or an attack as a normal connection (false negative). If the window size is very small there is a greater probability of "false alarms" since some routine normal packets can be mistaken for attacks. For example, with a window size of 4, a sequence with 5 normal connections that have the same src/dst_byte will be labeled as "possible attack". On the other hand, if a window size is very big, "false negatives" are more likely to occur since more records would be considered as normal. For example, with a window size of 20, a sequence with 15 attack connections that have the same size source and destination packets (src_byte, dst_byte) will be labeled as "normal". False positive rate increases as the window size decreases, and the false negative rate increases as the window size increases. While we would like to reduce false negative it may come at the cost of some false positive alarms. Hence, to determine appropriate window size, we need to balance between false positive and false negative rate.

Data mining algorithm

Data mining algorithms provides convenient tools to analyze vast amounts of data on network traffic to identify patterns or associations that can help to detect attacks. Some of the common algorithms used in the intrusion detection field are statistical analysis, neural networks, induction tree algorithms, and rough sets (Zhu et al., 2001). Induction learning algorithms develop the knowledge structure using decision trees. Each node in the decision tree is labeled with attributes and the edge is labeled with the attribute value and the leaf is labeled with class. The algorithm performs a top-down heuristic search through a problem space and uses information gain as the criterion for selecting the branching attribute of a node. The ID3 algorithm (Quinlan, 1984) and its improvement C4.5 (Quinlan, 1993), are very popular algorithms in data mining. A detailed discussion of the algorithm can be found in Quinlan (1993).

The C4.5 algorithm constructs classification rules in the form of a decision tree, recursively starting at the root. At each node, attribute ai is selected to split the training data into subsets where ai = 0 or 1. This algorithm is then invoked recursively on the two subsets of training data until all examples in one node belong to the same class. At this point, a leaf node is created and labeled as the expected value of the

categorical attributes for the records described by the path from the root to that leaf. The scheme used in decision tree learning for selecting attributes is designed to minimize the depth of the final tree. The idea is to pick the attribute that goes as far as possible toward providing an exact classification of the dataset. All we need is a formal measure of "fairly good" and "really useless". The measure should have its maximum value when the attribute is perfect and its minimum value when the attribute is of no use of all.

In general, the basic ideas are:

- In the decision tree each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf.

- In the decision tree, each node should be associated with the non-categorical attribute which is most informative among the attributes not yet considered in the path from the root.

- Entropy is used to measure how informative node is.

The data mining system used in this experiment is the Waikato Environment for Knowledge Analysis (WEKA) system, which was developed at the University of Waikato, New Zealand (Witten and Frank 2000). WEKA is a collection of machine learning algorithms that can either be applied directly to a dataset or called from your own Java code. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. WEKA provides an easy to use interface to specify a subset of attributes for the analysis from a larger set of attributes in the dataset. In this study an induction tree algorithm (J48), a version of C4.5 (Quinlan 1993), that is extensively used in data mining was employed.

Research Experiment

Data

The data set used for this experiment was obtained from the Third International Knowledge Discovery and Data Mining Tools Competition (KDD CUP), which was held in conjunction with KDD-99, the Fifth International Conference on Knowledge Discovery and Data

Mining (Hettich, and Bay 1999). The original data was obtained from a real-world military network environment with simulated intrusions. The training and test data had different distributions of classes. The raw tcpdump data was obtained from MIT Lincoln Lab and was pre-processed to include "time-based traffic" features (Lee and Stolfo, 2000). Since the full training data set is very big (743MB uncompressed), only 10% of the data was used in this study. The training data contains 494,021 connection records, of which 392,970 connections are involved in attack. The testing data set contains 311,039 connections, of which 250,436 are attacks. Besides the attacks listed in the training data, the testing data also includes some new attacks. Both training and testing connections are already correctly labeled as "normal" or "attack".

Selection of Window Size

Since window size has a significant impact on classification accuracy it is necessary to determine the optimal window size. Multiple experiments were conducted with varying window sizes and the classification rate was analyzed to determine the optimal window size. Window size was initially set at 10 connections and changed in increments of 10 connections. Based on the "size-based traffic" feature each connection was labeled as 0 (normal) or 1 (attack). Since the original data set was too large, a smaller sample was used to determine the optimal window size. Multiple rounds with different window sizes were processed using the induction tree algorithm to assess the accuracy in prediction.

Each round used the same set of 50,000 records to enable cross comparison across different window sizes. In order to check if these 50,000 records are a fair representation of the population we compared the proportion of attack-free records in the population and in the subset. It both cases it was around 20% ensuring that the sample was a good representation of the population.

It was necessary to ensure that the sample was representative of the original data set. It was not good to split the original data set into some small pieces and choose one of them for our experiment, because the connection records in each subset were still continuous, and the attack and normal features were not equally distributed. In other words, it was very possible that we could get a sub data set with the "size-based traffic" labels all set as 0 or 1. A better way to create a sample data set was to randomly choose connection records from the original data set.

First, several copies of the full sized data set were created. Each copy had the new "size-based traffic" feature generated with a different window size. After that a series of random numbers were generated and added to each connection record. Then the full sized data sets were re-ordered according to these random numbers. 50,000 records were selected from each of these full sized data sets and used in the data mining algorithm. Since all the records in the full sized data set has been re-ordered according to the same random number, the same 50,000 records were selected in each round from the full sized data sets.

Figure 3 and 4 illustrates the changes in classification rate and false alarm rate for different window sizes.
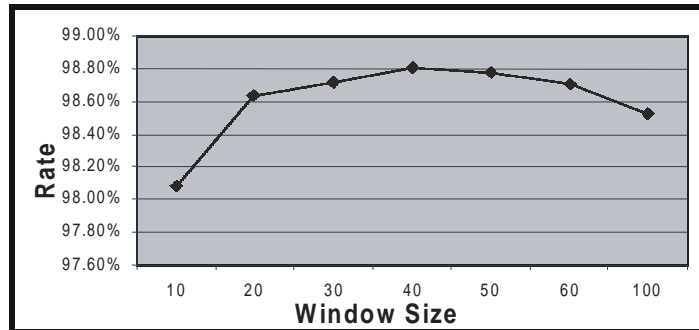

Figure 3. Classification rate

Figure 3 indicates that the classification rate initially increases as the window size increases, and after it reaches a peak with a window size of 40 it begins to decrease with further increase in window size. A window size of 40 connections provides the best classification rate. In all our subsequent experiments a window size of 40 is used.
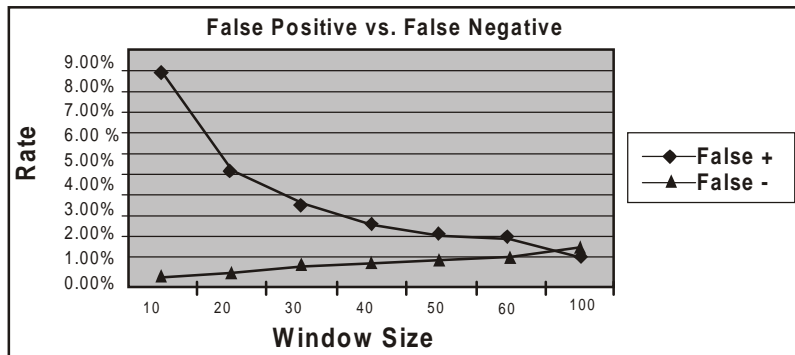

Figure 4. Comparison of false positive and false negative rate

Figure 4 shows that the false positive rate increases as the window size decreases, and the false negative rate increases as the window size increases. These two figures provide us useful information to select the appropriate window size. We notice that the change in false positive or false negative is relatively less beyond a window size of 40.

Result and Discussion

The experiment was conducted in three rounds. In the first round, "time-based traffic" feature was used in prediction; in the second round "size-based traffic" feature was used; and in the third round, the two features were combined for prediction. After these three rounds, the classification rates were compared. A window size of 40 was used for all experiments. We randomly ordered the records in the data set in order to make sure that any sample training set we retrieved from the complete data set was representative of the population.

Since WEKA has constraints on the size of the datasets, we split both the training and testing data sets into several data subsets. Each subset contained 35,000 connection records, except the last subset. We used the same training set and used the model it generated to test all the testing subsets. After we finished evaluating all testing subsets, we added up all correctly labeled connections to calculate the final classification rate. Each training/testing pair is run three times separately for the three rounds of experiment. The results of the experiments are shown in Table 3.

	"time-based feature" alone (Round 1)	"size-based feature" alone (Round 2)	"time-based feature"+ "size-based feature" (Round 3)
normal labeled as normal	59,734	59,027	59,008
normal labeled as attack	859	1,566	1,585
attack labeled as normal	56,352	25,008	23,494
attack labeled as attack	194,084	225,428	226,942
total connections	311,029	311,029	311,029
total correctly labeled	253,818	284,455	285,950
Classification rate	81.6059%	91.4561%	91.9368%

Table 3. A comparison of classification rate with/without "size-based traffic" feature.

From Table 3 we can see that "size-based traffic" feature has a better classification rate (91.4%) compared to the "time-based traffic"

feature (81.6%). Combining the two features further improves the classification rate, although only marginally (91.4% to 91.9%).

It was surprising that "size-based traffic" feature had a better classification rate compared to time-based traffic feature. It could be related to the type of attack. Most of the attack in this data set was DoS or probing attacks that have the same packet size pattern. On further analysis of the data set we found testing data subset 6 was primarily responsible for the differences in the classification rate. The testing subset 6 has a 99.6% attack rate and primarily contained neptune and smurf attacks. These attacks were much better detected by "size-based" traffic feature compared to 'time-based" traffic feature. If we analyze decision tree in the 1st round test for the attributes that are important, we find that "count" of the "time-based traffic" features plays a very important role in classifying a connection. Since in 'time-based' feature we used fixed time intervals (2 seconds) rather than window size to determine the count, it is possible that traffic levels (no. of connections/ second) may have affected the detection accuracy. A threshold count based on high traffic DOS attacks may not recognize low traffic attacks, which may have a large number of same size packets, but spread over a longer time period. On the contrary, the window size of "size-based traffic" feature defines a lower bound based on traffic, not on time, thereby making it more flexible and adaptable under different situations. For example, no matter how many attack connections come together during network congestion, if the sequence size is not less than the window size, the "size-based traffic" feature always give a correct label.

The decision tree generated by the induction algorithm provides information on how the model reaches its final prediction. The tree indicates that the first decision point tests for "size-based traffic" feature, and branches into two possible outcomes ("0-possible normal" and "1-possible attack"). On both branches the tree has a number of tests based on "time-based traffic" attributes before it reaches a leaf with a final prediction. The total number of tests (nodes) in the "possible attack" branch is significantly lesser than the total number of tests in the "possible normal" branch. This indicates that an attack with the same-packet-size pattern can be quickly detected, while an attack without this pattern will go through more decision points or nodes before it is labeled as normal or attack. The decision tree indicates that while "size-based" traffic feature is fast in identifying some attacks, "time-based traffic" features are required to detect other attacks.

7.1 Meta-Classification

Meta-learning is a mechanism for inductively learning the correlation of predictions by a number of base classifiers (Chan and Stolfo 1993, Fan et al. 2002). Meta-learning uses a learning algorithm and a fixed training set, and repeatedly applies the algorithm to different subsets of the training set or using different random choices within the algorithm in order to get a large ensemble of machines. The machines in the ensemble are then combined in some way to define the final output of the learning algorithm (e.g. classifier). The common meta-learning techniques are bagging, boosting and stacking. Bagging is used to combine classifiers generated by the same algorithm but on different data sets. Boosting also uses the same algorithm on different data sets, but it will give different weight to different classifiers. Stacking is used to combine classifiers based on the same data set but different algorithms. Unlike bagging and boosting, stacking is not widely used in practice, because if the majority of the algorithms perform poorly, the meta-result will also be poor. Another reason is that it is somewhat difficult to explain its result. Previous research has shown that an ensemble is often more accurate than any of the single classifiers in the ensemble (Optiz et al 1999).

Prior research indicates that that boosted and bagged versions of C4.5 produce more accurate classifiers than standard versions (Quinlan 1996). Bagging reduces the variance of the classification algorithms while boosting reduces both the bias and variance in the results (Bauer and Kohav 1999). Drucker et al (1994) have shown that as the size of the training set increases, the training error decreases for the boosting algorithms.

The meta-classification algorithm we used in this experiment is bagging and the base classification algorithm is C4.5. In this step we added additional training sets (10) and included other features like "host-based traffic" and "content-based traffic" features (Lee and Stolfo 2000) to further improve the classification rate and detect a wide range of attacks including U2R and R2L attacks. U2R is unauthorized access to local superuser privileges by a local unprivileged user and R2L is unauthorized access from a remote machine.

We first wrote a script that can directly call WEKA's classes, so that we can extract the classified label of each single testing record. Since there are 11 training sets, each single testing record can have 11 classified labels. Then the majority voting method is used to find out the final classified label. For example, if at least 6 training sets assign "1" to

a testing record, the final classified label will be "1"; otherwise the label would be "0". After that, this label is compared with the correct label to determine the meta-classification rate. The meta-classification rate was 93.62%. The rate was better than the rate obtained from the three rounds. However, the improvement was marginal compared to the size-based feature or the combined method.

Conclusion

Network attacks have become a significant threat to organizations and effective intrusion detection systems have to be developed to detect these attacks before they inflict harm on the internal network infrastructure. The characteristics of some common attacks and time-based features developed by Lee and Stolfo (2000) for detecting them were analyzed. The application of this new feature in some scenarios is also discussed. Since some of these attacks have similar packet size characteristics an intrusion detection system using the size-based feature was developed. The key idea is to compare the packet size in a sequence of continuous connections. After determining the optimal window size for analysis, the data set was pre-processed for the optimal window size and an induction algorithm was used to detect intrusions. The pre-labeled data was split into training and testing data set. The training data set was used to teach the system about various attacks. Three rounds of experiment were conducted with size-based feature, time-based features, and a combination of the two features. The results of the experiment indicate that size-based traffic feature performed as good or better than time-based traffic feature, and the combination of the two features improved the classification accuracy, although only marginally.

It should be noted that size-based feature is not a solution to all problems. It worked well in DoS and Probing attacks but may not work for other types of attacks. In general, "size-based traffic" feature should be combined with "time-based traffic" features in order to detect a wide range of attacks.

Reference

Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., and Stoner, E., et al. (2000), 'State of the Practice of Intrusion Detection Technologies,' Software Engineering Institute, Carnegie Mellon University, (CMU/SEI-SIM-010). Pittsburgh, PA.

Anderson, J. (1980), 'Computer Security Threat Monitoring and Surveillance,' *Fort Washington*, PA: James P. Anderson Co.

Bauer, E., and Kohav, R. (1999). 'An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants,' *Machine Learning,* 36, 105-139.

Carrettoni, F., Castano, S., Martella, G., and Samarati, P. (1991), 'RETISS: A Real Time Security System for the threat Detection Using Fuzzy Logic,'Proceedings of 25th IEEE International Carnahan Conference on Security Technology. October Taipei, Taiwan ROC.

Chan, P. K., and Stolfo, S. J. (1993), 'Toward Parallel and Distributed Learning by Meta-learning,' *AAAI Workshop in Knowledge Discovery in Database,* 227-240.

Denning, D.E. (1987), "An intrusion-detection model," *IEEE Transaction on Software Engineering*, 12(2), 222-232.

Drucker, H., Cortes, C., Jackel, L., LeCun, Y., and Vapnik, V. (1994), "Boosting and other ensemble methods," *Neural Computation*, 6(6), 1289-1301.

Durst, R., Champion, T., Witten, B., Miller, E., Spagnuolo, L. (1999), "Testing and Evaluating: Computer Intrusion Detection Systems," *Communications of the ACM*, 42(7), 53-61

Fan, W., Wang, H., Yu, P., and Stolfo, S. (2002), 'A fully distributed framework for cost-sensitive data mining,' Proceedings of 22nd International Conference on Distributed Computing Systems. July. Vienna, Austria.

Garvey, T. D., and Lunt, T. F. (1991), 'Model Based Intrusion Detection,' Proceedings of the 14th National Computer Security Conference, October, 372-385.

Hettich, S. and Bay, S. D. (1999), 'The UCI KDD Archive,' *kdd.ics.uci.edu*, Department of Information and Computer Science, University of California, Irvine, CA.

Hofmeyr, S.A., Forrest, S., and Somayaji, A. (1998), "Intrusion Detection Using Sequences of System Calls," *Journal of Computer Security*, 6, 151-180.

Ilgun, K., Kemmerer, R.A., and Porras, P.A. (1995), "State Transition Analysis: A Rule-Based Intrusion Detection Approach," *IEEE Transactions on Software Engineering,* 21(3), March, 1-22.

Cabrera, J. B. D., Lewis, L., Qin, X., Lee, W., and Mehra, R.K. (2002), "Proactive Intrusion Detection and Distributed Denial of Service Attacks - A Case Study in Security Management," *Journal of Network and Systems Management*, 10(2).

Kumar, S., and Spafford, E. H. (1995), 'A Software Architecture to Support Miscues Intrusion Detection,' Proceeding of the 18th National Information Security Conference, 194-204.

Lee, W., Stolfo, S. J., and Chan, P. K. (1999), 'Learning Patterns from Unix Process Execution Traces for Intrusion Detection,' *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, AAAI Press, 50-56.

Lee, W., and Stolfo, S. (2000), "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Transactions on Information and System Security*, 3(4).

Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., and Das, K. (2000), "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, 34, 579-595.

Lippmann, R., and Cunningham, R. K. (2000), 'Improving intrusion detection performance using keyword selection and neural networks,' *Computer Networks*. 34, 597-603.

Lunt, T., Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P., Javitz, H, Valdes, A., and Garvey, T. (1992), 'A Real-time Intrusion Detection Expert System (IDES) – Final Technical Report,' Computer Science Laboratory, SRI International, Menlo Park, California.

NSA (2002), "Therminator to watch for cyber attacks", *Federal Computer Week*, Dec. 13, 2002.

Optiz, D, and Maclin, R. (1999), "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research,* 11, 169-198.

Paxson, V. (1998), 'Bro: A System for Detecting Network Intruders in Real-Time,' Proceedings of 7th USENIX Security Symposium. San Antonio, TX, January 1998 [online]. Available at *http://www.aciri.org/vern/papers.html>*.

Porras, P.A., and Neumann, P. G. (1997), 'EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances,' Proceedings of 20th National Information Systems Security Conference, October 7-10, Baltimore, MD.

Porras, P. A., and Valdes, A. (1998). 'Live Traffic Analysis of TCP/IP Gateways,' Proceedings of the Network and Distributed System Security Symposium, March 11-13, San Diego, CA.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Quinlan, R. (1996), 'Boosting, bagging and C4.5', Proceedings of 13 National Conference on Artificial Intelligence (AAAI 96), AAAI Press, 725-730.

Richardson, R. (2005), '10th CSI/FBI Computer Crime and Security Survey,' *www.gocsi.com/press/20050714.jhtml,* July 14, 2005.

Teng, H. S., Chen, K., and Lu, S. C. (1990), 'Security Audit Trail Analysis Using Inductively Generated Predictive Rules,' Proceedings of the 11th National Conference on Artificial Intelligence Applications*,* 24-29 March.

Wagner, D., and Dean, D. (2001), 'Intrusion Detection via Static Analysis,' Proceedings of the 2001 IEEE Symposium on Security and Privacy.

Witten, I. H., and Eibe, F. (2000), *Data Mining: Practical machine learning tools with Java implementations,* Morgan Kaufmann, San Francisco.

Huang, Y., Fan , W., Lee, W., and Yu, P.S. (2003), 'Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies,' Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS), May*,* Providence, RI.

Zhu, D., Premkumar, G., Zhang, X., and Chu, C. (2001), "Data Mining for Network Intrusion Detection A Comparison of Alternative Methods", *Decision Sciences*, 32(4), Fall, 635-660.