THE USE OF ARTIFICIAL INTELLIGENCE FOR ENHANCED NETWORK DEFENSE

Michael Knight[(a)], Kortney Raulston[(b)], Kennard Laviers[(c)], Kenneth Hopkinson[(d)]

[(a)(b)(c)(d)]Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433-7765

[(a)]Michael.knight@afit.edu, [(b)]kortney.raulston@afit.edu, [(c)]Kennard.laviers@afit.edu, [(d)] kenneth.hopkinson@afit.edu

ABSTRACT

Even after a network intrusion system (IDS) has identified a cyber-attack, network administrators are still faced with the difficult challenge of assessing network health and status in order to appropriately take action to mitigate damage caused by such an attack due to the large amount of data available from the network components. This paper explores the use of auto-clustering to abstract network meta-data to form high-level units of information that are more comprehensible for a network administrator or an AI Agent to understand and act on. We perform an empirical analysis to evaluate our approach using the NSL-KDD99 dataset for both abstraction of network log data and attack family classification. By auto-clustering, we significantly increase the classification speed without greatly increasing the error.

Keywords: Classification, KDD99, IDS, C4.5, K-Means

DISCLAIMER

The views expressed in this document are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

1. INTRODUCTION

A significant problem with computer networks today is the ability to defend against cyber-attacks in real-time. Defending attacks becomes more difficult when there are an overwhelming amount of network features and samples to analyze. The goal of this research is to detect an attack and formulate a plan to counter the attack as it is happening. The primary focus of this paper will be on the detection portion of the problem. Without the detection of a cyber-attack, the process to create and carry out a plan to mitigate its effects cannot be developed. This plan should include information about the type of attack that is likely occurring in order to produce a set of actions or procedures to combat it. By clustering and classifying network traffic, personnel or monitoring agents can more easily determine if the traffic is originating from an attack or otherwise normal activity. This classifier can be trained on a given network to determine what normal traffic is. Furthermore, if significant features are highlighted before the classifier is created, some of the complexity can be reduced so attacks can be detected more easily. Clustering is the grouping of similar data items into clusters (Fung 2001). In this application, clustering is performed on the individual features of the network trace data so that the important/common features of attack strategies can be highlighted. The data set used is the well-known KDD 99 which is network data produced from a simulated Air Force network experiencing different forms of network attacks. WEKA is a software workbench design to support the application of machine learning technology to real world data sets (Garner 1995). WEKA is useful because of its versatility in allowing data to be presented in its own format, such as the KDD 99 data set, and it is designed to encompass all learning algorithms under a common interface. The algorithms used for this research are ReliefF, clustering by K-Means, and C4.5.

2. RELATED WORK

There has been a great deal of research in the area of learning feature representations from unlabeled data sets for high-level tasks such as classification. Much of this research has shown great progress on benchmark data sets like NORB and CIFAR by making use of complex unsupervised learning algorithms (Coates, Lee, and Ng 2011).

The WEKA system is traditionally used with agricultural data sets and these data sets tend to be larger and of lesser quality than those in machine learning data sets (Garner 1995). Applying the WEKA system to machine learning data sets allows us to answer questions like “Do these changes in certain features indicate a cyber attack?” as well as gain some insight into how to apply learning algorithms to existing real world data sets.

Previously, the classification of network traffic was performed through the use of port-based or payload-based analysis. This has become increasingly more difficult as peer-to-peer networks (P2P) adapt to using dynamic port numbers, masquerading techniques, and encryption to avoid detection (Erman, Arlitt, and Manhanti 2006). To combat these cyber-attack adaptations, an alternative approach for classifying network traffic is introduced by exploiting and extracting common or distinct attack strategy characteristics.

Authors of another related paper (McGregor, Hall, Lorier, and Brunskill 2004) describe their efforts using the Expectation Maximization (EM) algorithm to cluster network flows into different application types with a fixed set of attributes (features). The EM algorithm separates the network traffic into a few basic classes but the accuracy and quality of the clustering is limited due to nature of its fixed attributes. Another technique uses the sequential forward selection (SFS) algorithm (Zander, Nguyen, and Armitage 2005) to find the best feature set to avoid an expensive, exhaustive search. Some other data sets, similar to KDD 99, used with the SFS feature selection algorithm include the Auckland-VI, NZIXII and Leipzig-II traces.

This paper will use the popular and well-known KDD 99 data set for experimentation and evaluation purposes. A modification of the Relief algorithm is used for feature selection and clustering is performed via the K-means algorithm. Additionally, the C4.5 algorithm is used to build a decision tree based on the feature selection results.

3. METHOD

3.1. Data Set

The data set that is used to evaluate the approach introduced in this article is the KDD99 data set. This data set was introduced for the 1999 KDD Cup challenge to accurately classify network data to a given class of attack or normal traffic (KDD Cup 1999). The data set consists of both normal and attack traffic classes, with the attack classes making up the majority of the cases. In total, there are 23 classes and 41 features in the original KDD99 data set (Tavallaee, Bagheri, Lu, and Ghorbani 2009). Later, this data set was transformed into to NSL-KDD99 which solved some issues regarding redundant samples and the over training of classifiers. Dimensions of this data set include measures such as the protocol type, service, server error rates, and byte counts. The values types in the data are nominal, discrete, and real. Attack types include back dos, buffer_overflow u2r, ftp_writer2l, guess_passwd r2l, imap r2l, ipsweep probe, land dos, loadmodule u2r, multihop r2l, neptune dos, nmap probe, perl u2r, phf r2l, pod dos, portsweep probe, rootkit u2r, satan probe, smurf dos, spy r2l, teardrop dos, warezclient r2l, and warezmaster r2l.

Table 1 shows the breakdown of the sample sizes per class in the NSL-KDD99 data set. In this article, only classes with 20 or more samples will be considered due to the difficulty that classifiers have in distinguishing between small sample sizes because they cannot be easily trained to do so. After this modification, there are 14 classes to process and 125,901 samples in all. This is a relatively small decrease as the original sample size was 125,973.

3.2. Feature Selection

Before a classifier is created, only the NSL-KDD99 data set's significant features will be selected. The goal here is to reduce the number of features to be classified so that there are fewer features to process and yet the accuracy is not diminished too greatly. The ReliefF algorithm (Garner 1995) was used to select the best data set features for use by the classifier.

The Relief algorithm, shown in Algorithm 1, is the original algorithm that ReliefF is built upon. The Relief algorithm requires an input n for the number of instances to randomly select and updates the weight values associated with each feature based on choosing the two nearest neighbors. One of these two nearest neighbors is selected inside the class of the randomly selected instance while the other is the nearest outside the class. The ReliefF algorithm slightly modifies the original Relief algorithm by using the nearest k neighbors, where k is specified as an input, in and outside the class. Instead of choosing one nearest neighbor, it chooses k neighbors. This modification reduces the number of random selections of instances needed. ReliefF also allows for any number of classes.

Table 1: Number of Samples, NSL-KDD99 Classes

Item	Type	Count
1	normal	67343
2	neptune	41214
3	werezclient	890
4	ipsweep	3599
5	portsweep	2931
6	teardrop	892
7	nmap	1493
8	satan	3633
9	smurf	2646
10	pod	201
11	back	956
12	guess_passwd	53
13	ftp_write	8
14	multihop	7
15	rootkit	10
16	buffer_overflow	30
17	imap	11
18	warezmaster	20
19	phf	4
20	land	18
21	loadmodule	9
22	spy	2
23	perl	3

The Relief algorithm, shown in Algorithm 1, is the original algorithm that ReliefF is built upon. The Relief algorithm requires an input n for the number of instances to randomly select and updates the weight values associated with each feature based on choosing the two nearest neighbors. One of these two nearest neighbors is selected inside the class of the randomly

selected instance while the other is the nearest outside the class. The ReliefF algorithm slightly modifies the original Relief algorithm by using the nearest k neighbors, where k is specified as an input, in and outside the class. Instead of choosing one nearest neighbor, it chooses k neighbors. This modification reduces the number of random selections of instances needed. ReliefF also allows for any number of classes.

```
set all weights W[A]:=0.0
for i ← 1 to n do
  begin
    select instance of R randomly
    H ← nearest hit
    M ← nearest miss
    for A ← 1 to cardinality(all_attributes) do
      W[A]←W[A]–diff(A,R,H)/n + diff(A,R,M)/n
  end
```

Algorithm 1: Relief Algorithm Pseudocode (Kononenko, Simec, and Edvard 1995)

The primary parameters in WEKA for the ReliefF algorithm are the number of neighbors and the number of instances. The number of neighbors parameter is used to select a given number of neighbors to search in the algorithm. The neighbors include those samples in the randomly selected sample's class as well as the nearest neighbors in the other classes. Because of this, the parameter has to be carefully selected so the number does not go below the smallest total from amongst the classes. The number of instances parameter defines the number of instances to select. In each instance, a random sample is selected and the ReliefF algorithm updates the *weights* of each feature. The *weights* are similar to a scoring feature used to rank the best features and updated according to the equation in Equation 1.

$$W[A] := W[A] - \frac{diff(A,R,H)}{n} + \sum_{C \neq class(R)} \frac{[P(C) \times diff(A,R,M(C))]}{n} \quad (1)$$

Equation 1 is the ReliefF weight update equation. Please refer to (Kononenko, Simec, and Edvard 1995) for a full description of this derivation.

In feature selection, a greater number of instances will produce the most accurate weights due to the fact that they will approach their steady state values after enough random samples are selected. After the features are selected, a classifier will be built using those features.

3.3. Classification

The data is broken up using a 10-fold validation to test the accuracy of the proposed classification method. The classification was performed with clustering using both the K-Means (MacQueen 1967) and C4.5 algorithms. The K-means algorithm is built upon a very simple foundation: Given a set of initial clusters, assign each point to one of them and each centroid of the cluster is replaced by the mean point on their respective cluster (Fung 2001).

The pseudocode for the K-Means algorithm is shown below (MacQueen 1967):

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The K-Means classifier is chosen for this application due to its ease of implementation. The number of clusters to be chosen is based on the number of classes in the data set. The goal is to have each class mapped to a cluster using the best features selected by the ReliefF analysis. The trained classifier will then classify a set of test samples and the accuracy will be measured.

Pseudocode for the C4.5 algorithm (Quinlan 1993), also known as J48 in the open source Java implementation in WEKA, is shown below (Kotsiantis 2007):

1. Check for base cases
2. For each attribute *a*, find the normalized information gain from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recurse on the sub lists obtained by splitting on *a_best*, and add those nodes as children of *node*

4. RESULTS

4.1. Feature Selection Results

Using the WEKA ReliefF implementation, several runs/experiments were performed. We adjusted the neighbor number *k*, and number of instances, *n*. Tables 2-7 shown on the proceeding page depict the top 10 best attributes (features) across multiple values of *k* (number of classes). A feature is considered better if it has a higher correlation to the value being classified.

All instances were used in the data set. There were a total of 125,973 entries. For the number of neighbors *k*, values of two and ten were used. Two is the smallest total quantity for a class. This makes it reasonable to make two the most likely accurate weight. Ten was also used as the number of neighbors with the result having the same top 10 features except with a different order among them. These simulations were rerun with a much smaller set instances chosen at random to see if the same 10 features were selected again.

Compared to the previous three figures, there are no changes in the features displayed after changing *n* from 5000 to 1000 instances. There are, however, some

differences in the ordering of the rankings when 1000 random instances are selected, but the same features are all represented in the top ten.

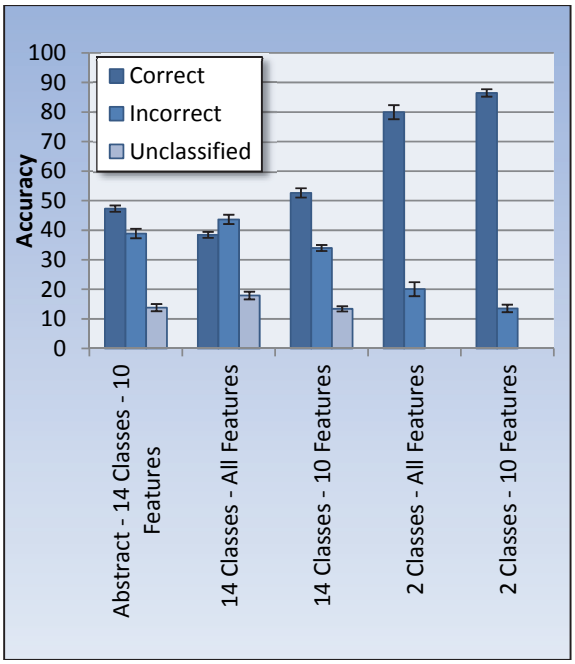

Figure 1: Results from K-Means Varying Classes and Feature Selection

Table 2: Top 10 Best Attribute from ReliefF Using WEKA with k = 10 and n = 5000

Feature Number	Feature Name
3	service
38	dst_host_serror_rate
4	flag
25	serror_rate
26	srv_serror_rate
12	logged_in
39	dst_host_srv_serror_rate
29	same_srv_rate
33	dst_host_srv_count
34	dst_host_same_srv_rate

Table 3: Top 10 Best Attribute from ReliefF Using WEKA with k = 5 and n = 5000

Feature Number	Feature Name
3	service
38	dst_host_serror_rate
4	flag
25	serror_rate
26	srv_serror_rate
29	same_srv_rate
12	logged_in
39	dst_host_srv_serror_rate
33	dst_host_srv_count
34	dst_host_same_srv_rate

Table 4: Top 10 Best Attribute from ReliefF Using WEKA with k = 2 and n = 5000

Feature Number	Feature Name
3	service
4	flag
38	dst_host_serror_rate
29	same_srv_rate
25	serror_rate
26	srv_serror_rate
33	dst_host_srv_count
12	logged_in
34	dst_host_same_srv_rate
39	dst_host_srv_serror_rate

Table 5: Top 10 best attribute from ReliefF using WEKA with k = 10 and n = 1000

Feature Number	Feature Name
3	service
38	dst_host_serror_rate
4	flag
25	serror_rate
26	srv_serror_rate
12	logged_in
39	dst_host_srv_serror_rate
29	same_srv_rate
33	dst_host_srv_count
34	dst_host_same_srv_rate

Table 6: Top 10 Best Attributes from ReliefF Using WEKA with k = 5 and n = 1000

Feature Number	Feature Name
3	service
38	dst_host_serror_rate
4	flag
25	serror_rate
26	srv_serror_rate
29	same_srv_rate
12	logged_in
39	dst_host_srv_serror_rate
33	dst_host_srv_count
34	dst_host_same_srv_rate

Table 7: Top 10 Best Attributes from ReliefF Using WEKA With k = 2 and n = 1000

Feature Number	Feature Name
3	service
4	flag
38	dst_host_serror_rate
29	same_srv_rate
25	serror_rate
26	srv_serror_rate
33	dst_host_srv_count
12	logged_in
34	dst_host_same_srv_rate
39	dst_host_srv_serror_rate

4.2. Classification Results

We use the same number of entries (125,973) for the classification results. The first classifier used for clustering was K-Means. Initially, all features were tested for classification accuracy based on clustering when 14 clusters were chosen. Afterwards, the ten features found by the previous ReliefF calculations were used as a basis for comparison. We found that using the K-Means algorithm (Fig. 1) our classification worked best across all 14 classes using 10 features without the auto-clustering (EM) employed to abstract the feature set. However, switching to the C4.5 algorithm we see the result dramatically boosted by the use of the feature abstraction.

Table 10 shows that the abstraction system classifies the attack type slightly worse on all 14 classes (95% vs. 97%) in the abstracted data-set with a significantly reduced cluster-based data-set. However, as indicated by Table 11 the classifier also acts far quicker (347% faster with K-Means and 48% faster using C4.5) on the simplified cluster-based data set which can be critical in a time-sensitive application.

Table 8: Results from C4.5 Using 14 Classes

	C4.5 –All Features	C4.5 –Ten Best Features
Correct	99.7967%	97.4791%
Incorrect	0.2033%	2.5703%
Unclassified	0%	0%

Table 9: Results Using C4.5and 2 Classes

	C4.5 –All Features	C4.5 –Ten Best Features
Correct	99.7817%	98.487%
Incorrect	0.2183%	1.513%
Unclassified	0%	0%

Table 10: Results Using C4.5 and 14 Classes on Abstract Data Transformed from the EM Algorithm

	C4.5 –Ten Best Features
Correct	95.0811%
Incorrect	4.9189%
Unclassified	0%

Table 11: Time Comparison Using Abstract Features Vs. Normal Feature Set

10 Features, 14 Classes	K-Means	C4.5
Abstracted	5.67	3.69
Normal (in seconds)	25.38	5.49
Improved %	447.62%	148.78%

5. CONCLUSION

This work introduces a new concept of using auto-clustering with the Expectation Maximization algorithm to significantly simplify the feature-set of network traffic along with the use of auto-feature extraction to reduce the number of features. As a result, using the K-Means algorithm our system was able to improve classification speed over 14 classes by over 347% (K-Means) and 48% (C-4.5) clearly indicating this method can be effectively used to improve classification accuracy by a significant margin.

With the methodologies described in this paper, administrators are able to assess a network's health and status more easily. By utilizing the uniqueness of various features in a network, they can be clustered and evaluated to recognize a cyber-attack. There are multiple avenues for future work using auto clustering and feature selection in networks. A more complete version of the KDD99 data set could be used to improve classification results. Success would depend on available computer resources. Experiments using a *k* value of 10% resulted in long algorithm run-times. Another interesting area for future research would be implementing a larger variety classification and feature selection algorithms. For example, the K-means algorithm is relatively simple and easy to implement, but suffers from two drawbacks. First, it is often slow and expensive when used on large datasets and can be sensitive to the initial clusters selections (Fung 2001) which is a stochastic process. Performing a comparative analysis on various classification and feature selection algorithms would provide an indication of which algorithms are more successful in detecting bad network traffic. Furthermore, there are many parameters associated with the algorithms used in this paper and a study to understand how they compare to the current results when undergoing a wider range of experiments would prove useful. Finally, a test benchmark could be constructed to create more realistic data sets than KDD99. This opens up the ability to perform simulated attacks to test the classifier. More accurate classifiers lead to better and more accurate planning and response systems.

REFERENCES

Coates, A., Lee, H., and Ng, N.A., 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning, *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, JMLR Workshop and Conference Proceedings, 15.

Erman, J., Arlitt, M., Manhanti, A., 2006. Traffic Classification Using Clustering Algorithms, *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, 281-286.

Fung, G., 2001. A Comprehensive Overview of Basic Clustering Algorithm. *Artificial Intelligence (Computer and Information Science)*, Citeseer press, 1-37.

Garner, S. R., 1995. WEKA: The Waikato environment for knowledge analysis, *Proceedings of the New Zealand Computer Science Research Students Conference*.

KDD Cup, 1999. Data, Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [accessed 13 July 2012].

Kononenko, I., Simec, E., and Edvard, I.K., 1995. Induction of decision trees using RELIEFF.

Kotsiantis, S.B., 2007. Supervised machine learning: A review of classification techniques, *Informatica,* (31), 249–268.

MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*. Available from: http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html#macqueen. [accessed 13 July 2012].

McGregor, A., Hall, M., Lorier, P., Brunskill, J., 2004. Flow Clustering Using Machine Learning Techniques, *Passive & Active Measurement Workshop*, April 19-20, France.

J. R. Quinlan, J.R., 1993. *C4.5: Programs for Machine Learning,* Available: http://books.google.com/books?id=1F1QAAAAMAAJ [accessed 13 July 2012].

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set, *IEEE Symposium on Computational Intelligence for Security and Defense Applications*.

Zander, S., Nguyen, T., and Armitage, G., 2005. Automated Traffic Classification and Application Identification using Machine Learning, *Proceeding of the IEEE Conference on Local Computer Networks 30th Anniversary*, 250-257.