Applying Kernel Methods to Anomaly Based Intrusion Detection Systems

Karim Ali
David R. Cheriton
School of Computer Science
University of Waterloo
Waterloo, Ontario
karim@uwaterloo.ca

Raouf Boutaba
David R. Cheriton
School of Computer Science
University of Waterloo
Waterloo, Ontario
rboutaba@uwaterloo.ca

***Abstract*—Intrusion detection systems constitute a crucial cornerstone in securing computer networks especially after the recent advancements in attacking techniques. IDSes can be categorized according to the nature of detection into two major categories: signature-based and anomaly-based. In this paper we present KBIDS, a kernel-based method for an anomaly-based IDS that tries to cluster the training data to be able to classify the test data correctly. The method depends on the K-Means algorithm that is used for clustering. Our experiments show that the accuracy of detection of KBIDS increases exponentially with the number of clusters. However, the time taken to classify the given test data increase linearly with the number of clusters. It can be derived from the results that 16 clusters are sufficient to achieve an acceptable error rate while keeping the detection delay in bounds.**

***Index Terms*—Kernel Methods, Machine Learning, Intrusion Detection Systems.**

I. Introduction

The vast ongoing achievements in computer technology brought about new schemes of attacks as well as pushing the technology forward. An attacker can now infiltrate a network through a vulnerable host then use it to launch attacks on other hosts. Moreover, an attacker can use the collaborative power of infiltrated hosts to perform a Distributed Denial of Service (DDoS) attack [1]. IDSes are an example of the tools used to safeguard computer systems against such types of attacks. The main goal of an IDS is to detect malicious/unauthorized use of resources. There are two types of IDSes: signature-based (e.g. Snort [2]) and anomaly-based (e.g. Bro [3]). The former cannot detect novel intrusions. However, the latter serves well in detecting anomalies that deviate from the normal behaviour of the system whether this abnormality was encountered before or not. Anomaly-based IDS mostly depend on statistical modeling [4] to define a system profile reflecting its normal behaviour. Other anomaly-based approaches [5], [6] rely on predicting the future behaviour of the system given its history. Although these approaches are more successful in capturing temporal and multiple-variable correlations, they require more time for training the model and, in some cases, their application can be infeasible because of the size of data sets involved [7]. In this paper, we will discuss different kernel methods that were applied to IDSes and propose an algorithm that tries to build k clusters in the feature space based on a defined kernel method. The rest of the paper is structured as follows. Section II discusses related work. The motivation of our work is explained in section III. Implementation details are explained in section IV. Section V explains the process of evaluating the system with the results presented in section VI. Finally, section VII states some concluding remarks and possible future work.

II. Related Work

A. Intrusion Detection Systems

1) Snort: Snort [2] is an open source cross-platform, lightweight, rule-based network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of malicious network traffic. It provides system administrators with enough data helping them to take informed decisions on the proper action to take towards a suspicious activity. Moreover, Snort can plug potential holes in a network's security coverage, such as when a new attack emerges and commercial security vendors are slow to release new attack recognition signatures.

Snort is available under the GNU General Public License [8], and is free for use in any environment.

978-1-4244-4624-7/09/$25.00 ©2009 IEEE

Therefore, Snort is useful when it is not cost efficient to deploy commercial NIDS sensors.

Snort developers lately added very useful extensions to its basic infrastructure. One of the extensions provides the functionality of Intrusion Prevention System (IPS) allowing system administrators to provide Snort with the appropriate actions to take if some incidents occur.

2) Bro: Bro [3] is an open source stand-alone high-speed monitoring system for detecting network intruders in real-time. In Bro, there is a clear distinction between mechanism, policy and extensibility by dividing it into three components. An *event engine* that reduces a kernel-filtered network traffic stream into a series of higherlevel events. A *policy script interpreter* that interprets event handlers written in Bro scripting language used to express a site's security policy. Finally, *event handlers* can update state information, synthesize new events, record information to disk, and generate real-time notifications via syslog.

B. Kernel-based Intrusion Detection Systems

1) Kernel-ART: Kernel-ART [9] is an adaptive intrusion detection algorithm which combines the Adaptive Resonance Theory(ART) with the Concept Vector and the Mercer-Kernel. Kernel-ART can detect unknown types of intrusions in an on-line fashion by generating clusters incrementally. The authors use the Concept Vector which is the weight vector of each cluster normalized to the mean vector of the clusters. Therefore, the learning rate parameter in updating the weight vectors is implied which will improve the speed of the execution. They also improve the data separability by mapping the input data vectors to a feature space using the Mercer Kernel.

2) Fuzzy Kernel with Multiple Hyperspheres: [10] presents a novel classification algorithm that is based on fuzzy kernel with multiple hyperspheres (FKMH) that tries to cover all the training samples of each class during the training process. The ultimate goal is to make each hypersphere cover as many samples of the input as possible using a greedy algorithm. The greedy method used depends on moving the centre of the hypersphere so that it encompasses as many data points as possible. Moreover, FKMH defines a fuzzy membership function that is used to label a testing sample as one pattern class according to the membership values.

III. Motivation

Applying Kernel methods has proved itself in pattern analysis-related problems that are very critical like drug industry. On the other hand, not much effort was done in prior work to relate kernel methods to IDS, especially anomaly-based IDS. That's why it seems to us that applying kernel methods to that field of computer security is very interesting. In addition, the collected data for an IDS can produce nicely shaped clusters such that each cluster represent the normal behaviour of a set of users. In other words, clusters could stand for the normal behaviour of system administrators, normal users, superusers or guest users. Moreover, the attacks that can be performed against a network are themselves categorizable by nature. Therefore, developing a clustering algorithm for anomaly-based IDS looks like an interesting problem to study in depth.

IV. KBIDS

A. Input Data

IDSes can be categorized according to the type of event data they analyze. An IDS that monitors traffic flowing in a network is usually called *network-based*, while an IDS that analyzes data produced locally at a host is often referred to as *host-based*. The input data that we will work with is gathered from a network, so the algorithm used is for a *network-based* IDS. The data represents the normal behaviour of the network only. This makes the system adaptive to future changes in any malicious or abnormal behaviour that it encounters. In other words, KBIDS will recognize the normal behaviour of the system marking any other behaviour as abnormal. Therefore, we do not define a set of predefined abnormal behaviours which makes the system more robust against new attacks. This approach is referred to *whitelisting* as opposed to *blacklisting*.

B. Clustering

KBIDS will try to cluster the input data images in a feature space having two goals in mind. The first goal is to have the least number of possible clusters. This goal is very important because it will reduce the number of clusters a test point has to be checked against during the operation of the IDS. This will also help boost the performance of the IDS which is important because some attacks need real time or *pseudo-real* time notification to successfully defend against them. The second goal is to choose the clusters such that as many input data points as possible are covered by the chosen clusters. This will help reduce the false positive (FP) rate of KBIDS.

The algorithm that we use to create the clusters in the feature space is the *K-Means* clustering algorithm [11]. *K-Means* tries to create K clusters. The distance between the points is measured using a kernel function that can

be changed but remains constant each time the algorithm runs.

C. Detection

After the system is trained to generalize from the given input data, it will be used in order to detect intrusions that are present in the test dataset. If a test data input vector does not lie on any of the clusters obtained through the training phase, it will be considered an intrusion. Therefore, an alarm will be raised to notify the system administrator that there has been a malicious behaviour going on. The detection algorithm is given in algorithm 1.

Algorithm 1 Detect

```
for each point p in the test data do
    for every cluster c do
        if p lies within c then continue checking
        else raise alarm
        end if
    end for
end for
```

V. Evaluation

A. KDD CUP 99

The latest publicly available dataset for testing IDS algorithms from MIT-Lincoln Labs is the DARPA 2000 [12] dataset. However, the latest form of this data that is compatible with machine learning algorithms was achieved in 1999 in the Third International Knowledge Discovery and Data Mining Tools Competition. Although the KDD-99 dataset is old, it will give a good indication of the accuracy of the proposed algorithm. In general, testing datasets is considered one of the major problems in IDS research.

B. Data format

A typical line in the KDD 99 dataset is represented by a set of features that was monitored by the data collection tool. Each line ends with its designated class. Some of the features have numeric datatypes, while others (vis. protocol type, service, flag and class) are enumerated data. The set of features and their possible values are given in the KDD-99 dataset.

C. Data Preprocessing

The KDD 99 dataset had to be reformed to the Attribute-Relation File Format (ARFF) [13] in order to be used with Weka [14] machine learning tool. The data preprocessing phase involves many steps and more than one pass over the input data. The dataset was divided into two parts: test data and training data. The training data is 10 times the test data and is about 45 MB while the test data is almost 4 MB. The pseudocode for preprocessing is given in algorithm 2.

Algorithm 2 PreProcessData

```
loop                                        ▷ over input dataset
    Add protocol to the list of protocols
    Add service to the list of services
    Add packet flag to the list of packet flags
    Add packet class to the list of packet classes
end loop
Build headers for train.arff and test.arff
Add data rows to both files
```

D. Parameter Selection

There are some parameters required for the K-Means algorithm to run. The algorithm is initialized with a random seed that is used in the random selection of the initial k centroids. This random seed is derived from the current time stamp of the machine. Another parameter is the number of clusters that should be generated. This parameter is variable and a group of values is used as it will be explained in the results.

VI. Results

In order to evaluate KBIDS, we performed several experiments in order to see the performance of KBIDS in three aspects: false positives (measure as sum of squared errors), time to classify test data and number of iterations of *K-Means* take to cluster the data. Figure 1 shows that the accuracy of KBIDS increase exponentially with the increase in the number of clusters used. However, Figure 2 shows that the time taken to classify the test data increases in a linear fashion with the number of clusters. Figure 3 shows the random relation between the number of iterations of K-Means and the number of clusters. This is attributed to the fact that the initial centroids are selected randomly so the number of iterations will depend on how good the initial centroids are. Our results suggest that 16 clusters can be considered an optimal number of clusters for the given dataset. This number of clusters will cause the sum of squared errors to be

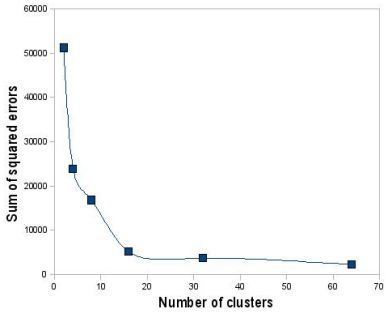

Fig. 1: Relation between number of clusters and sum of squared errors

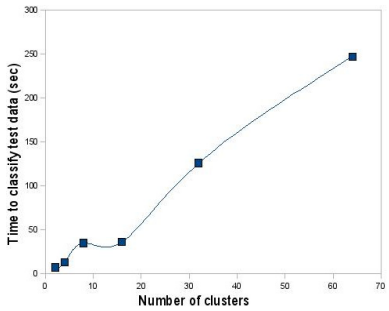

Fig. 2: Relation between number of clusters and time taken to classify test data

5111 with 48 seconds delay of detection which is still acceptable compared to the size of the test data.

VII. Conclusion

Intrusion detection systems represent an important cornerstone in securing computer networks. They are considered on of the early defences against malicious behaviour and attacks that might threaten the secured system. IDSes can be categorized according to the nature of detection into two major categories: signature-based and anomaly-based. In this paper we presented a kernel-based method for an anomaly-based IDS that tries to cluster the training data to be able to classify the test data correctly. The method depends on the K-Means algorithm that is used for clustering. The results of our experiments show that the number of clusters have a great effect on the accuracy of the algorithm and on the time taken to classify the test data. However, the results show that 16 clusters can be considered an optimal number of clusters for the given dataset.

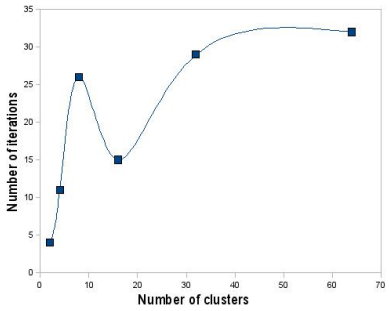

Fig. 3: Relation between number of clusters and number of iterations of K-Means algorithm

KBIDS can be further extended in order to decrease the delay of classifying the test data since in some attack incidents real-time response is very crucial. Moreover, it would be a good feature if re-generating the clusters can be done dynamically as the input data changes. This will enable the system to be trained using the new incoming data while it is using previous clusters to classify the test data.

References

[1] J. Mölsä, "Mitigating denial of service attacks: a tutorial," *Journal of Computer Security*, vol. 13, no. 6, pp. 807–837, 2005.

[2] M. Roesch, "Snort - lightweight intrusion detection for networks," in *LISA '99: Proceedings of the 13th USENIX conference on System administration*. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238.

[3] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 1998, pp. 3–3.

[4] D. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[5] P. D'haeseleer, S. Forrest, and P. Helman, "An immunological approach to change detection: algorithms, analysis and implications," May 1996, pp. 110–119.

[6] T. Lane and C. E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *ACM Transactions on Information System Security*, vol. 2, no. 3, pp. 295–331, 1999.

[7] D. Dasgupta and F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 281–291, Jun. 2002.

[8] R. Stallman, "Gnu general public license," 1989. [Online]. Available: http://www.gnu.org/copyleft/gpl.txt

[9] H. Lee, Y. Chung, and D. Park, *Advances in Knowledge Discovery and Data Mining*. Springer Berlin / Heidelberg, 2006, ch. An Adaptive Intrusion Detection Algorithm Based on Clustering and Kernel-Method, pp. 603–610.

[10] G. Lei, W. Hui-zhong, and X. Liang, "A novel classification algorithm based on fuzzy kernel multiple hyperspheres," vol. 2, Aug. 2007, pp. 114–118.

[11] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[12] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, 2000.

[13] "Attribute-relation file format," 2008. [Online]. Available: http://www.cs.waikato.ac.nz/ ml/weka/arff.html

[14] "Waikato environment for knowledge analysis." [Online]. Available: http://www.cs.waikato.ac.nz/ ml/