Towards Efficient Intrusion Detection Systems Based on Machine Learning Techniques.

Mariano Vallés[1], Carlos Catania[1] and Carlos García Garino[1,2]

[1] Instituto para las Tecnologías de la Información y las Comunicaciones (ITIC)
[2] Facultad de Ingeniería
Universidad Nacional de Cuyo, Mendoza, Argentina
mariano.valles@fcm.uncu.edu.ar
{ccatania,cgarcia}@itu.uncu.edu.ar

Abstract. Intrusion Detection Systems (IDS) have been the key in the network manager daily fight against continuous attacks. However, with the Internet growth, network security issues have become more difficult to handle. Jointly, Machine Learning (ML) techniques for traffic classification have been successful in terms of performance classification. Unfortunately, most of these techniques are extremely CPU time consuming, making the whole approach unsuitable for real traffic situations.
In this work, a description of a simple software architecture for ML based is presented together with the first steps towards improving algorithms efficiency in some of the proposed modules. A set of experiments on the 1998 DARPA dataset are conducted in order to evaluate two attribute selection algorithms considering not only classification performance but also the required CPU time. Preliminary results show that computational effort can be reduced by 50% maintaining similar accuracy levels, progressing towards a real world implementation of an ML based IDS.

1 Introduction

The problem of network security has become more relevant in the past years. In the beginning of Internet, protocols supporting it worked well. With the fast growth of the Internet, more and more attacks against data confidentiality, authenticity and availability were performed using underlying vulnerabilities in protocols such as ARP, TCP, TELNET, SMTP and FTP. Although most of these faults have been fixed, new ways of attacking networks are discovered everyday.

Therefore, network managers in charge of preventing attacks are in a continuously changing environment. Intrusion Detection Systems (IDS) are a remarkable tool to aid in this task. There are two main approaches to IDS [1]. The first one is misuse detection IDS, which works by representing attacks in a signature or pattern, and based on this pattern detects attacks by using a large set of rules that describe every known attack. Its main disadvantage is the fact that it is very difficult to detect unknown attacks. The second approach is called anomaly detection IDS which builds an statistical model to describe normal traffic and

any deviation from this normal traffic is considered an anomaly and classified as an attack. This approach is capable of detecting unknown attacks but due to the difficulty in classifying normal traffic, many times this is followed by an important false positive rate.

Although the mentioned approaches are commonly used in real world IDS their disadvantages are an issue to improve. To open a new perspective in the implementation of an IDS, Machine Learning (ML) algorithms have been used and have proved to achieve good accuracy results when classifying attacks over network traffic [2]. Many of these techniques, however, imply a great use of CPU time in their process, something that may not be suitable for a real network environment.

Before classifying traffic, it is necessary to preselect a number of the *attributes* present in the instances of network traffic. The set of instances together with the selected attributes will form what is known in ML as a *dataset*. This preselection stage is very important. In this case, it is the starting point to succeeding in the classification of further instances as attacks or normal traffic. A good way of deciding which attributes to consider is to use others experience under this field [3]. Among these attributes source IP addresses, protocol type, connections flags or port numbers can be found.

As mentioned in the previous paragraph, the correct attributes selection is the key for obtaining high values in classification. Although, it is important to notice that the more attributes used, the more CPU time and loss of efficiency in the implementation. For this reason, the selection of a small group of attributes that offer enough information for attack detection is a fundamental issue in the development of efficient IDS.

The aim of this work is to carry out a study of the minimum required attributes to efficiently implement a ML based IDS by decreasing CPU time but not the algorithms' accuracy in classification of network traffic. To do so, Consistency Based (CON)[4] and Correlation Feature Selection (CFS)[5] , two of the most frequently used attribute selection algorithms have been applied to the original set of attributes present in the datasets and then, ran over some of the most representative classification algorithms to compare its results before and after the selection of attributes. Evaluation is carried out using traffic instances extracted from the 1998 DARPA dataset [6], widely used by IDS researchers since it was introduced in the context of 1999 KDD Cup.

The rest of this paper is organized as follows. Section 2 provides a description of the modules involved in the development of IDS based on ML with special focus on the attributes selection process. A deep evaluation of representative ML algorithms and their attributes selection needs are presented in section 3. Finally, in section 4 concluding remarks and future work are commented.

2 Machine Learning Based IDS

When considering the study of an IDS based on Machine Learning techniques, an overview from a software architectural point of view seems necessary. Spe-

cial emphasis is put on modules responsible for handling the network data and putting those modules relevant for classification aside.

2.1 Components of a ML based IDS Architecture

A simple architecture for an IDS based on ML techniques can be organized in four modules[7].

The first module has the responsibility for the traffic capture process. The *Traffic Capture Module* intends to acquire all the data available on the wire in raw mode. Then the *Preprocessing Module* has the responsibility to select, extract and rearrange the data in a way needed by the algorithms on the ML module. The *Learning Module* uses the information from the *Preprocess Module* to build a model which, in turn, will be used for classifying new traffic instances. Finally, the *Evaluation Module* uses the model obtained from the *Learning Module* to classify new traffic instances.

During the last years, a considerable number of ML algorithms have been proposed for being implemented inside the *Learning Module*. In that sense, Algorithms such as K-nearest neighbor (k-NN)[8], Naive Bayes (NB), Neural networks [9] and decision trees[10], just to mention a few, have exhibited high attack classification performance and seem to be ready for practical uses.

Despite the algorithm implemented, the *Learning Module* requires a number of values that measure different aspects of a traffic instance. These values are usually referred as *attributes* in the ML context.

One possibility consists of using as *attributes* all the data in raw mode acquired from the *Traffic Capture Module*. However, the application of ML algorithms on such data would require considerable computational effort and could turn the whole ML approach infeasible. More appropriate is the idea of using only a carefully selected set of *attributes* which can be representative enough for applying ML algorithms with less computing effort and maintaining a reasonably good performance. As mentioned in previous paragraphs this selection is performed by the *Preprocessing Module*.

2.2 Network Traffic Attributes

On the *Preprocessing Module* the set of selected *attribute*s for describing the traffic data consists of a number of fields available from a network traffic instance as well as other high level attributes obtained after some network packet preprocessing.

Table 1 shows a total of fifteen fields related to a TCP connection commonly used as *attributes* by previous works on network intrusion detection [11,12]. In this case only *protocol*, *tcp.srcport*, *tcp.dstport*, *ip.src* and *ip.dst* are easily obtained from an individual TCP connection. Remaining ones are higher level attributes which provide information related to connection time and data transferred.

Table 1: Basic attributes of individual traffic connections.

Attribute Name	Description
connection.time	Time of the connection in hours, minutes and seconds
protocol	Type of protocol, e.g ssh,http,ftp
tcp.srcport	TCP source port
tcp.dstport	TCP destination port
ip.src	IP source address
ip.dst	IP destination address
tcp.len	Number of bytes transferred
num.pkts.src.dst	Number of packets from src IP to dst IP
num.pkts.dst.src	Number of packets from dst IP to src IP
num.ack.src.dst	Number of packet with ACK flag active from src to dst
num.ack.dst.src	Number of packet with ACK flag active from src to dst
num.syn.src.dst	Number of packet with SYN flag active from src to dst
num.syn.dst.src	Number of packet with SYN flag active from dst to src
num.bytes.src.dst	Number of bytes from src to dst
num.bytes.dst.src	Number of bytes from dst to src

Attributes involving many connections are shown in Table 2. These high level *attributes* show information involving the number of connections computed using a five-second time window as well as information about the last 20 connections.

Note that the 32 *attributes* shown in Table 1 and Table 2 are far from being a complete *attribute* list, but they have been considered useful in other works in the IDS field.

2.3 Attributes Selection

For many ML algorithms the use of a large set of *attributes* like the ones mentioned in Subsection 2.2 can lead to high classification performance results. However, in some cases, the good performance is observed only at the expense of a considerable computing effort, which could lead to the unviability of the ML approach. Therefore, it seems that selecting the minimal set of attributes required by the *Learning Module* could be the first step on the development process of an IDS based on ML.

Consistency Based (CON)[4] and Correlation Feature Selection (CFS)[5] are two of the most frequently used algorithms for attribute selection. The final idea behind these algorithms is finding an appropriate subset of attributes while maintaining a high classification rate.

In this work both attribute selection algorithms are evaluated in order to get a reduced subset of the original 32 attributes shown in Subsection 2.2. Evaluation should be conducted observing not only classification rate but also CPU time required for building the model and the time required for evaluating new traffic instances.

Table 2: High level attributes involving many connections.

Attribute Name	Description
Information about the connections in the last five seconds	
count.time.src.	Number of connections from the same address as the current connection source address
count.time.dst	Number of connections to the same IP address as the current connection destination IP address
count.time.srv.src	Number of connections from the same service as the current connection
count.time.srv.dst	Number of connections to the same service as the current connection
Information about the last 20 connections	
count.src	Number of connections from the same address as the current connection source address
count.dst	Number of connections from to the same address as the current connection destination address
count.srv.src	Number of connections from the same service as the current connection
count.srv.dst	Number of connections to the same service as the current connection

3 Experiments

The experiments of this section focus on comparing a set of representative ML algorithms in terms of classification performance and CPU time required when datasets have been filtered using different *Attribute Selection* algorithms.

The ML algorithms chosen are examples of four of the major algorithm families present in WEKA software[13] , widely used in the ML community. As an example of the decision trees, *J48* is proposed, *Naive Bayes (NB)* as part of the statistics based, *IBk1* as an example of the instance based, and *Multilayer Perceptron (MLP)* as an example of the neural networks.

Experiments are carried out on an Intel Core 2 Duo E8400 with 4GB DDR2 RAM memory running WEKA version 3.6.1 over Debian Lenny 5.0 GNU/Linux.

3.1 Performance Metrics for IDS evaluation

Accuracy, a commonly used metric in the ML field, is used for evaluating the performance classification. On the other hand, CPU time is evaluated considering both *model build* time and *test* time.

Accuracy is computed as the ratio between the number of correctly classified traffic instances and the total number of traffic instances. *Model build* time refers to the CPU time the ML algorithm needs for building the model while *test* time refers to the time required by the algorithm for classifying new instances.

3.2 Dataset description

Evaluation is performed using an extract of five weeks from the 1998 DARPA dataset. The attack instances are distributed differently among 9 datasets. Varying from a dataset with 10% of instances labeled as attacks and 90% as normal traffic, to one with 90% of attacks and 10% of normal traffic. Each dataset consists of 11639 instances of network traffic. A set of 32 attributes are selected as mentioned in Subsection 2.2.

It is important to note that Cross folding validation, with 10 folds per dataset, is used to perform the experiments[13].

3.3 Evaluation of the complete attribute set

Figure 1 shows average classification accuracy for the nine datasets when the complete *attribute* set is used. All algorithms show near optimal accuracy over the different attack distributions. *NB* algorithm is the one showing lowest accuracy of the selected four. Although, the value shown is 99,11%, which can be considered extremely good.

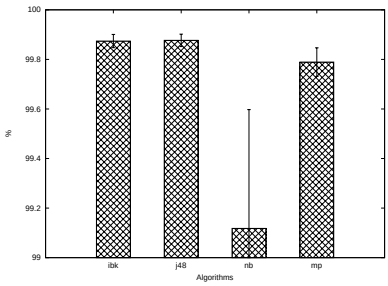


Fig. 1: Average classification accuracy using the complete attribute set

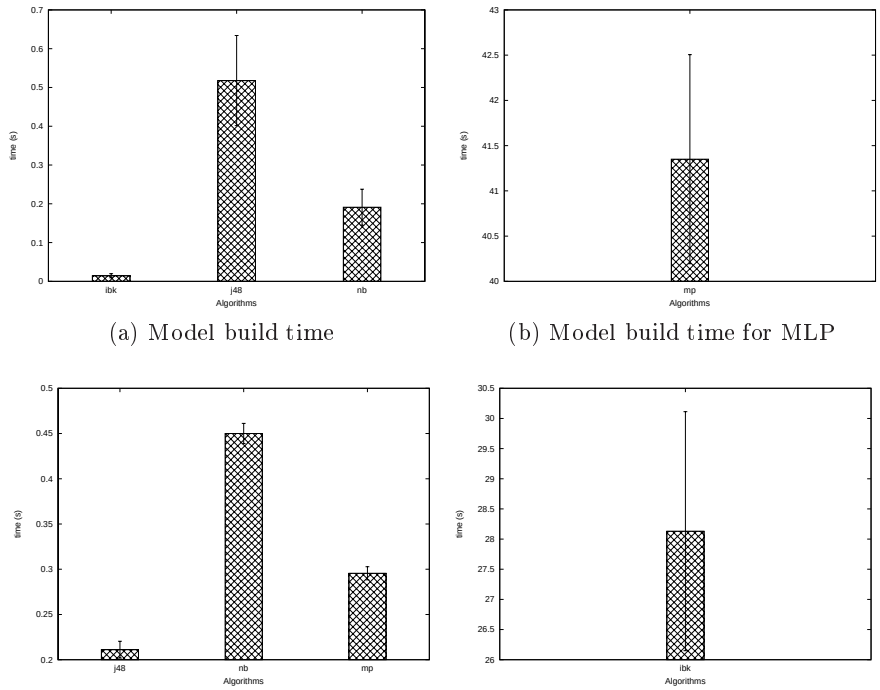
Figures 2(a) and 2(b) show average time needed for building the model. Due to the fact that MLP exhibits an important difference in y-axis magnitude, its results are detached and shown in Figure 2(b).

Timing difference exists between the different algorithms, especially for *MLP* which takes approximately 80 times more , when compared to the next slowest one *J48*.

Test time can be observed in Figure 2(c) and 2(d). As*IBk1* takes as least 62 times more to test instances against its model than the rest, it is shown separately in Figure 2(d).

3.4 Evaluation of the CON and CFS Selected Attribute Set

As shown by Figure 3(a), in most cases, datasets with an attribute set reduced by CON or CFS algorithms, take half the model build time, compared to the



(a) Model build time (b) Model build time for MLP

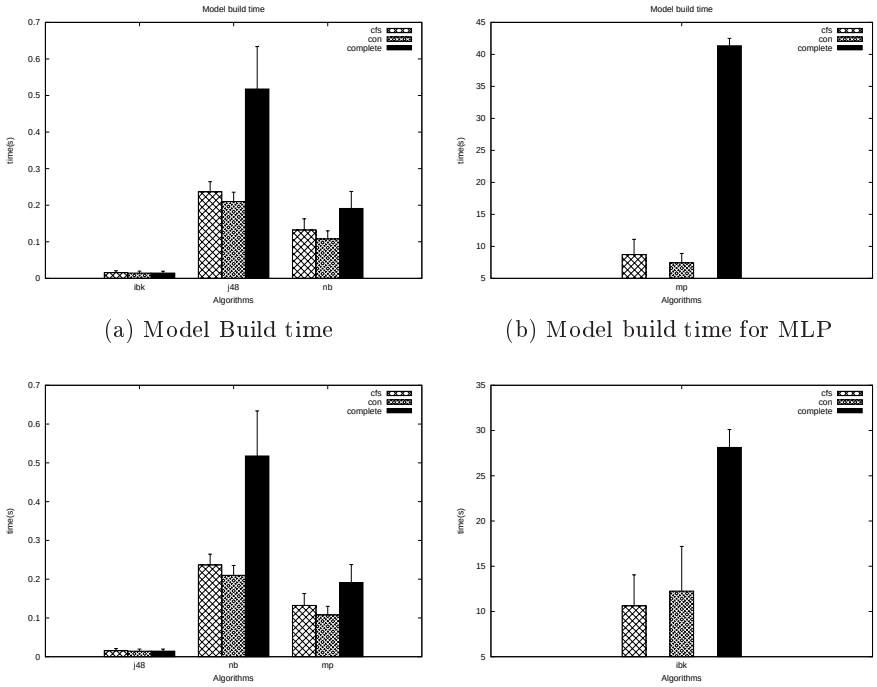
(c) Model test time (d) Model test time for IBk1

Fig. 2: Model Build time and Test time using the complete attribute set

dataset using a complete set. One Exception is *MLP*, shown in Figure 3(b) which takes at the most, a fourth of the time when the attributes have been reduced using CON or CFS and due to a difference of scale in the y-axis magnitude, is shown separately.

Figure 3(c) shows the average time in seconds for the algorithms to test datasets with a reduced attribute set. The relation is that a reduced attribute set takes approximately half the time to be tested than the complete one. Also, *IBk1* algorithm takes an excessive amount of time for testing, making it impossible to be shown in Figure 3(c) and shown in Figure 3(d).

Figure 4, shows the average classification accuracy for each algorithm using the complete attributes set and compares it with the attributes sets obtained by CFS and CON. Only *NB* and *MLP* show an appreciable difference in accuracy when the complete attribute set is used compared with the attributes set obtained by CON and CFS in the algorithm. Nevertheless, CFS shows better results than CON for *NB* and *MLP* algorithm.



(a) Model Build time

(b) Model build time for MLP

(c) Model Test time

(d) Model test time for IBk1

Fig. 3: Model Build and Test time for the complete and the selected attributes set obtained by CON and CFS algorithms

4 Conclusions and Future Work

Classification accuracy exhibited on different attack distribution datasets confirms that ML based IDS are a viable solution for dealing with network security problems. However, the computational effort required, in some cases, seems excessive for a real world implementation.

Attribute selection appears to be a useful tool in the process of lessening these CPU usage time without a meaningful reduction of the classification accuracy. In many cases, using the attribute selection algorithms, a reduction of 50% in build and test time can be achieved, over different attack distribution datasets.

Among the different experiments the use of *J48* and *NB* algorithms offer the best overall results. Even though, both algorithms have shown good accuracy levels, when the selected attribute set has been used, *J48* has exhibited an accuracy around 99% in all the experiments and has shown build and test time suitable for an IDS implementation.

Finally, It is important to mention that the obtained accuracy levels might be biased by the fact the 1998 DARPA dataset is old and outdated. Therefore, experiments should be conducted in a real network environment.

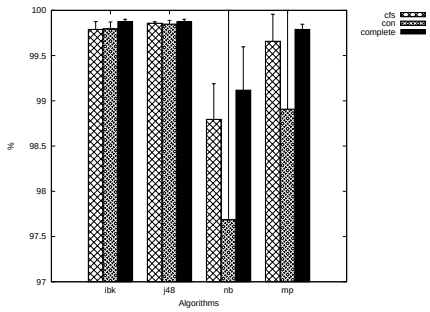


Fig. 4: Average classification accuracy for the complete and the selected attributes obtained set by CON and CFS algorithms

5 Acknowledgments

The authors would like to thank the financial support received by the projects PAE-PICT 2312 granted by ANPCyT and 06/M010 granted by UNCuyo.

References

1. Mukherjee, B., Heberline, L.T., Levitt, K.: Network instruction detection. IEEE Network **8** (1994) 26–41
2. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: A review. Expert Systems with Applications **36**(10) (2009) 11994 – 12000
3. Dimitris, G., Ioannis, T., Evangelos, D.: Feature selection for robust detection of distributed denial-of-service attacks using genetic algorithms. In: Methods and Applications of Artificial Intelligence. Volume 3025/2004. (2004) 276–281
4. Dash, M., Liu, H.: Consistency-based search in feature selection. Artificial Intelligence **151**(1-2) (2003) 155 – 176
5. Hall, M.A.: Correlation-based Feature Subset Selection for Machine Learning. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand (April 1999)
6. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings. Volume 2. (2000) 12 –26 vol.2
7. Catania, C.: Clasificacion de patrones de trafico de redes utilizando herramientas de computacion evolutiva en entornos de computacion distribuida. Master's thesis, Facultad de Ingenieria, Universidad de Mendoza, Mendoza, Argentina (November 2007)
8. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In: Applications of Data Mining in Computer Security, Kluwer (2002)
9. Ryan, J., jang Lin, M., Miikkulainen, R.: Intrusion detection with neural networks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, MIT Press (1998) 943–949

10. song Pan, Z., can Chen, S., bao Hu, G., qiang Zhang, D.: Hybrid neural network adn c4.5 for misuse detection. In: Proceedings of the Second International Conference on Machine Learning and Cybernetics. (2005)
11. Lee, W., Stolfo, S.J.: Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX Security Symposium. (1998)
12. Catania, C., García Garino, C.: Reconocimiento de patrones en el tráfico de red basado en algoritmos genéticos. Inteligencia Artificial, Revista Iberoamericana de IA **12**(37) (2008) 65–75
13. Witten, I., Frank, E.: Data Mining, Practical Machine Learning Tools and Techniques. 2nd edn. Morgan Kaufmann (2005)