Neural networks vs. decision trees for intrusion detection

Yacine Bouzida
Mitsubishi Electric ITE-TCL
1, allée de Beaulieu CS 10806
35708, Rennes, France
Bouzida@tcl.ite.mee.com

Frédéric Cuppens
Département RSM GET/ENST Bretagne
2, rue de la Châtaigneraie F-35576, Cesson Sévigné, France
Frederic.Cuppens@enst-bretagne.fr

ABSTRACT
Signature based intrusion detection systems cannot detect new attacks. These systems are the most used and developed ones. Current anomaly based intrusion detection systems are also unable to detect all kinds of new attacks because they are designed to restricted applications on limited environment. Current hackers are using new attacks where neither preventive techniques mainly based on access control nor current intrusion detection systems can prevent the devastating results of these attacks against information systems. We enhance the notion of anomaly detection and we use both neural networks and decision trees for intrusion detection. Since these techniques are mainly applicable to misuse detection, we use our anomaly detection enhancement and improve these techniques for anomaly detection. Experimental results demonstrate that while neural networks are highly successful in detecting known attacks, decision trees are more interesting to detect new attacks. The proposed methods outperform previous work in detecting both known and new attacks.

KEY WORDS
Intrusion Detection, Anomaly Detection, Neural Networks, Decision Trees.

I. Introduction

Anomaly intrusion detection systems are not well studied or explored as misuse detection ones. Misuse detection consists in using patterns of well known intrusions to match and identify known labels for unlabeled datasets. In fact, many commercial and open source intrusion detection systems (IDSs) are misuse based ones. Recently, attackers have explored serious break-ins to many commercial and government sites where serious damages have occurred. The different intrusions that have been used were new. This situation was foreseeable because the attackers are attempting to develop new attacks forms where neither misuse detection tools nor access control tools (such as firewalls) installed in our networks may detect or stop these new attacks forms.

Anomaly detection, on the other hand, consists in building profiles of normal behaviors then detecting any deviation of a new behavior from the learned normal profiles. This definition (of anomaly detection) is restrictive because only one class which corresponds to the normal behavior is learned.

In this paper, we extend the definition of anomaly detection to not only take into account normal profiles but also handle known attacks and explore supervised machine learning techniques, particularly neural networks and decision trees for intrusion detection. In fact, decision trees induction algorithm has proven its efficiency in predicting the different classes of the unlabeled data in the test data set for the KDD 99 intrusion detection contest [9]. Since machine learning techniques, generally, cannot find boundaries between known and unknown classes, an extension of neural networks and decision trees is introduced to deal with new unknown anomalies.

The rest of the paper is organized as the following. Section II presents the state of the art of current anomaly detection method in general and the limitations of the current anomaly detection tools that only learn normal behaviors and flag suspicion when deviation, from the established normal behavior, is observed. Based on this state, we enhance the anomaly detection notion for detecting novel attacks. Section III briefly presents background of neural networks and an improvement of this technique to handle new attacks with the different results obtained for both the standard multilayer neural network and its enhancement for new attacks detection. In Section IV, we present decision trees induction algorithm with an improvement for anomaly detection and the corresponding results. Finally, Section V concludes the paper.

II. Problem statement and motivation

Anomaly intrusion detection is the first intrusion detection method that was introduced to monitor computer systems by Anderson [1] in 1980 and then improved by Denning [6] in 1987. At that time, intrusion detection was immature since only user behavior and some system events were taken into account. In fact, this approach consisted in establishing normal behavior profile for user and system activity and observing significant deviations of the actual user activity with respect to the established habitual profile. Significant deviations are flagged as anomalous and should raise suspicion. This definition did not take into account the expert knowledge of known vulnerabilities and known attacks. This is why we enhance the notion of anomaly detection not only by considering normal

profiles but also by taking into account abnormal behaviors that are extracted from known attacks.

Since we have knowledge about known vulnerabilities and their corresponding attacks, we may enhance the anomaly detection by adding to the learning step the abnormal behavior corresponding to known attacks. Therefore anomaly detection would consists in learning all known *normal* and *attack* profiles. Based on this learnt knowledge, anomaly detection has then to detect whether a new observed profile is normal or abnormal and its corresponding known attack is determined or the observed profile is new and therefore it is considered as a novel unknown behavior. Thereafter, we suggest that a diagnosis should be done on the observed traffic that has caused the detection of the new anomaly in order to find out the reason of this new observation. Thus, if it corresponds to a new activity that was not seen before it is flagged either as a normal profile or as a new attack. The new observations with their real classification would then be considered for further investigation. We note that the diagnosis of the new observed behaviors is not our main objective here.

In our knowledge, all the efforts done by different researchers for detecting new attacks in the KDD 99 either consider unrealistic hypotheses or obtain uninteresting results. In the following, we recall the different measures that are used to rank the different proposed methods for this task. We use this measure to assess our results discussed in Sections III and IV.

To rank the different results a cost matrix C is defined [7]. Given the cost matrix illustrated in Table I and the confusion matrix obtained subsequent to an empirical testing process, a cost per test (CPT) is calculated using the formula given in Equation (1).

	Normal	**Probing**	**DoS**	**U2R**	**R2L**
Normal	0	1	2	2	2
Probing	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

TABLE I

THE COST PER TEST MATRIX.

$$CPT = \frac{1}{N} \sum_{i=1}^{5} \sum_{j=1}^{5} C_{i,j} * CM_{i,j} \qquad (1)$$

where C corresponds to the cost matrix, N is the number of instances in the test data set and CM corresponds to the confusion matrix obtained subsequent to the method that is used in the classification task.

The accuracy of each experiment is based on the percentage of successful prediction (PSP) on the test data set.

$$PSP = \frac{number\ of\ successful\ instance\ classification}{number\ of\ instances\ in\ the\ test\ set} \qquad (2)$$

The different techniques that are applied to the KDD 99 data sets did not detect unknown attacks as new ones because the methods used for this task are not anomaly based techniques. However, they consisted in learning the signatures of the connections (attacks or normal traffic) using the 41 attributes composing the different connections. Then the new connections are compared to the learning data set using the model constructed during the training phase. We call this technique: *misuse detection by learning* because it differs from other signature based techniques, such as snort, bro, etc., where only attack signatures are used. On the other hand, it is the role of the expert to write the different rules using the corresponding known vulnerabilities.

Based on the above considerations and limitations and the need to explain the failure of supervised techniques to detect U2R and R2L attacks, we investigate two supervised techniques, namely neural networks and decision trees in order to explain the failure of machine learning techniques in the KDD 99 contest.

In our experiments, we use the KDD 99 data sets without altering any sample or considering any new sample as described in Table II. The attacks that have any occurrence in the learning set should be detected as known attacks and others —those that are absent in the training set and are present in the test set— are considered as anomalies and should be predicted as new attacks.

The default supervised algorithms do not deal with unknown classes. They are interesting since they can generate alarms in real time at the end of a connection by contrast to unsupervised techniques that remain unusable for real time intrusion detection. Separate modules for anomaly and misuse detection may not be as efficient as a single module with the two techniques in the same time. These observations have motivated us to enhance supervised anomaly detection techniques as presented in the following sections.

In the following, we motivate our research with neural networks to compare its results with the best entries. We consider them also as a cross validation technique with decision trees for the task of detecting new attacks. While the successful detection rate of the new U2R attacks is increased, in our experiments, the R2L attack remains very low. This suggested us to focus on the transformation done by the MADAM/ID tool and prove that this transformation is not an appropriate one. The low detection rate of new R2L attacks is due to this transformation and not to the enhanced machine learning algorithms particularly the decision trees induction algorithm.

III. NEURAL NETWORKS

A. Backpropagation technique for intrusion detection

Backpropagation is a neural network learning algorithm. A neural network is a set of connected units following a particular topology. Each neuron is described by a unit that has an input and an output. Two neurons are connected if the output of one of them is connected to the input of the other. Each connection in a neural network has a weight associated to it. The topology of the neural network, the training methodology for weights' adjustment and the connections between the different neurons define the type of the corresponding neural

Probing $(4, 107; 4, 166)$	DoS$(391, 458; 229, 853)$
ipsweep$(1, 247; 306)$, mscan$(0; 1, 053)$, nmap$(231; 84)$, portsweep$(1, 040; 364)$, saint$(0; 736)$, satan$(1, 589; 1, 633)$.	apache2$(0; 794)$, back$(2, 203; 1.098)$, land$(21; 9)$, mailbomb$(0; 5, 000)$, neptune$(107, 201; 58, 001)$, pod$(264; 87)$, processtable$(0; 759)$, smurf$(280, 790; 164, 091)$, teardrop$(979; 12)$, udpstorm$(0; 2)$.
U2R$(52; 228)$	R2L$(1, 126; 16, 189)$
buffer_overflow$(30, 22)$, httptunnel$(0; 158)$, guess_passwd$(53; 4, 367)$, loadmodule$(9; 2)$, perl$(3; 2)$, perl$(3; 2)$, ps$(0; 16)$, rootkit$(10; 13)$, sqlattack$(0; 2)$, xterm$(0; 13)$.	ftp_write$(8; 3)$, imap$(12; 1)$, multihop$(7; 18)$, named$(0; 17)$, phf$(4; 2)$, sendmail$(0; 17)$, snmpgetattack$(0; 7, 741)$, snmpguess$(0; 2, 406)$, spy$(2; 0)$, warezclient$(1, 020; 0)$, warezmaster$(20; 1, 602)$, worm$(0; 2)$, xlock$(0; 9)$, xsnoop$(0; 4)$.

TABLE II

THE DIFFERENT ATTACK TYPES AND THEIR CORRESPONDING OCCURRENCE NUMBER RESPECTIVELY IN THE TRAINING AND TEST DATA SETS.

network. In our study, we are interested in the multilayer neural networks using the backpropagation learning algorithm [14]. In a multilayer neural network, there are three kinds of layers. Each layer contains a set of neurons. The first layer, called input layer, sets the activation of its neurons according to the provided pattern in question. The output layer provides the answer of the network. A multilayer network may contain one or many hidden layers although in practice, usually one is used.

Like any supervised learning technique, a multilayer neural network has two phases. The learning phase where the network learns by adjusting the weights so as to be able to predict the correct class label of the new input patterns during the test phase.

Before the training process, one should define the number of hidden layers (if more than one) and the number of neurons on each layer. The number of neurons on the input layer corresponds to the number of attributes that represent a sample. However, input values should be numerical to perform the backpropagation algorithm. Therefore, the discrete values are transformed into a vector as it is explained in the following. For each different discrete value of an attribute is assigned a neuron on the input layer. For example, for the protocol type *(tcp, udp, icmp)*, there are three inputs, say I_0, I_1, I_2 assigned to this attribute. Each unit is initialized to 0. If the protocol type of the current connection is *tcp* (resp. *udp, icmp*) then I_0 is set to 1 (resp. I_1 is set to 1 and so on). One output unit, on the output layer, may be used to represent exactly one class. So, if the output of a neuron on the output layer is equal to 1 then the corresponding class is designed as the predicted class. The number of hidden layers and the number of units on each hidden layer is established by experience during the training phase since there are no clear rules as to set the *best* number of hidden layer units.

The use of neural networks in intrusion detection is not new because there are at least two works that were developed during the last decades. The first model is used in Hyperview [5] for a user behavior modeling. The second one is that discussed in [4]. This latter was used as a misuse detection tool where only packet header attributes are considered for analysis to detecting denial of service and port scan attacks. While these works used neural networks for either user anomaly detection or misuse detection, we use them here for both network misuse and anomaly detection particularly over the different KDD 99 data sets [9].

B. Experimental methodology and results

Some parameters of the neural network are known a priori from the provided problem. The number of neurons on the input layer, in our example using the KDD 99 data sets, is equal to 125 units because the discrete attributes among the 41 attributes are translated into continuous ones. The number of neurons on the output layer is equal to the number of the total classes corresponding to the five classes considered in the KDD 99 contest *(normal, probing, DoS, U2R* and *R2L respectively)*.

Other parameters such as the number of hidden layers, the number of neurons in the hidden layers, the momentum, the learning rate and the number of iterations are determined by experience.

In the following, the number of hidden layers we consider in our neural network architecture is limited to only one hidden layer. After performing different tests using one hidden layer, two then three hidden layers we did not obtain a significant improvement in comparison with using only one hidden layer. The momentum is fixed to 0.60 after many experiments where this parameter varied over the interval $[0.20, 0.90]$. The learning rate is fixed to 0.20 after varying it over the interval $[0.10, 0.50]$. However, the weights values of the different connections in the whole network are randomly initialized in the interval $[-0.50, 0.50]$.

Since each neuron on the output layer corresponds to one class, the neuron with the highest value defines the predicted class. Using this technique, every sample will be assigned a class among the five classes defined a priori.

Table III presents the confusion matrix related to the best percentage of successful prediction obtained after combining the best parameters of the neural network.

Predicted / Actual	%Normal	%Probing	%DoS	%U2R	%R2L
Normal(60,593)	**97.87**	0.75	1.20	0.00	0.18
Probing (4,166)	10.68	**71.63**	15.34	0.00	2.35
DoS (229,853)	2.62	0.36	**97.00**	0.00	0.02
U2R (228)	86.84	7.02	3.95	**0.00**	2.19
R2L (16,189)	73.20	0.06	0.06	0.00	**26.68**
$PSP = 93.10\%$, $CPT = 0.2072$					

TABLE III

CONFUSION MATRIX WHEN USING THE BACKPROPAGATION TECHNIQUE WITH THE BEST PARAMETERS.

We mention, from Table III, that the prediction ratio $PSP = 93.10\%$ and the cost per test $CPT = 0.2072$ outperform all the results of the previous works done over KDD 99. However, the U2R class is undetectable. This is because the number of samples of the U2R class during the learning step

IF (all neurons on the output layer are less than a threshold) THEN
The corresponding connection is new
A diagnosis should be performed
ELSE
Let the k^{th} neuron be the most activated one,
This connection corresponds to the k^{th} class.
FI

Fig. 1. Classification process using a threshold.

is the lowest one (52 instances over $494,021$). Therefore, it is difficult to learn this category using neural networks. Our first goal does not consist in outperforming previous work done over KDD 99 intrusion detection contest. However, we want to understand why all these algorithms fail to detect the last two attack classes, namely U2R and R2L. We also note that the two attacks U2R and R2L are often detected as a normal traffic (86.84% for U2R and 73.20% for R2L) in almost all the techniques that are used for this purpose. There are many R2L and U2R instances that are new (see Table II) in the test data sets since their corresponding attack type is not present in the learning data set.

In order to detect these new attacks we improve the classification process of the neural networks as the following. A threshold θ is defined. Therefore, if the value of the highest output neuron is below this threshold, the corresponding connection is considered momentarily anomalous however a diagnosis should be performed for further investigation. The diagnosis is not a goal here. Figure 1 presents this algorithm.

Figure 2 shows the variation of the percentage of successful versus the variation of the a priori fixed threshold θ.

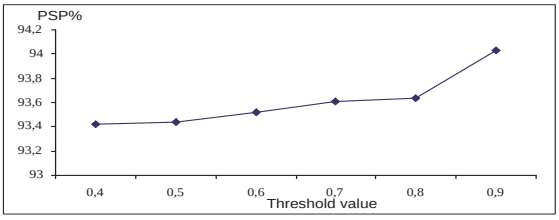

Fig. 2. PSP variation according to the considered threshold value.

The results shown in Figure 2 are performed over the same neural network using the best parameters. We mention that the attacks instances that are predicted as new attacks are considered as a successful prediction ratio.

While the whole successful prediction ratio increases, the corresponding prediction ratio of each class decreases respectively. Figure 3 shows the different prediction ratios of each class[1] when varying the threshold.

According to Figure 3, even if the threshold is set to 0.90 the two classes DoS and Normal remain detectable in their actual classes. This means that the neural network has correctly learned these two classes. However, the probing and the R2L

[1]The U2R class is not presented since it is not detected even before considering the threshold.

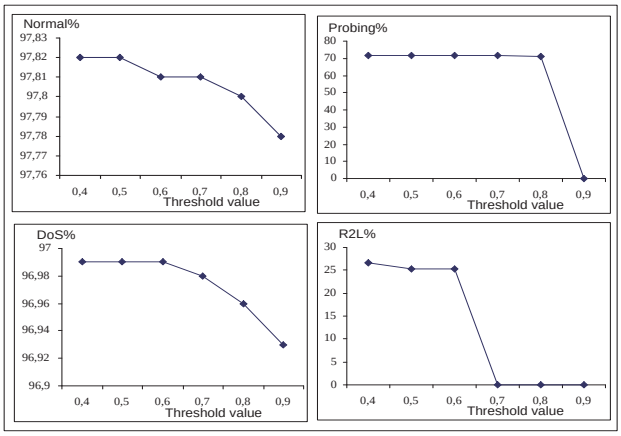

Fig. 3. Different classes PSP variation according to the considered threshold value.

classes are not predicted in their corresponding actual class when considering the threshold equal to 0.70 for R2L and 0.90 for the Probing attack class. This means that the instances of the test corresponding to these two classes are not well learned or are not close to their corresponding instances in the learning data set.

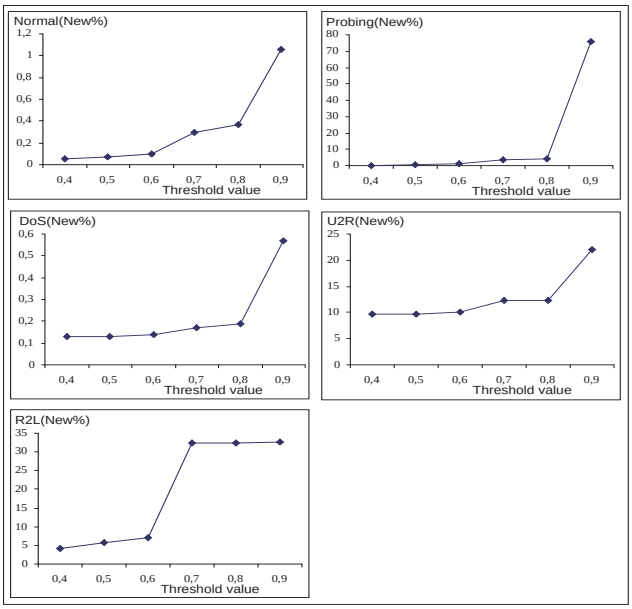

Fig. 4. Different classes ratios detected as a new class.

We report in Figure 4 the prediction ratios of the different classes that are detected as new ones. Figure 4 shows that while increasing the threshold value the two attack classes R2L and Probing are detected as new ones. This means that they are moving from their actual class when no threshold was considered to a new class as if they differ from their real class. However, Figure 5 presents the different attack classes that are detected as a normal class while increasing the value of the threshold. It is interesting to note that the prediction ratio of

these attacks as a normal one remains respectively stable for all of them even if the value of this threshold is equal to 0.90. The two classes U2R and R2L are always detected as normal with rates exceeding 76.75% for U2R and 64.4% for the R2L class. This means that the new instances of these two classes seemingly are close to the instances of the normal connection present in the training set.

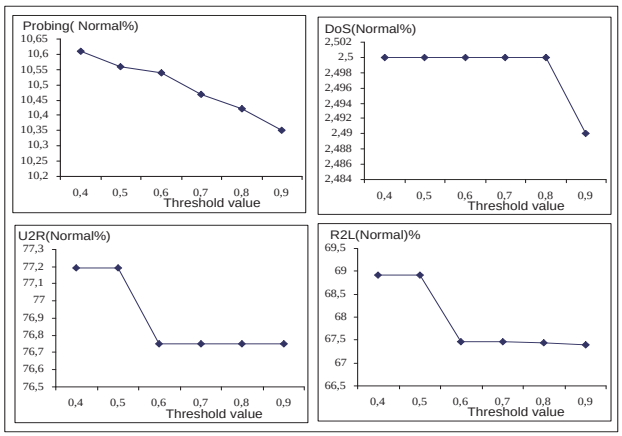

Fig. 5. Ratios of the different attack classes detected as a normal class.

Although the neural network outperforms all previous works done over KDD 99 intrusion detection data sets, it failed to detect the attacks that are not present with low number presence in the training data set particularly the U2R attack instances that are almost not predicted in the whole experiments.

While the neural networks transform the discrete values of the different attributes into numerical values, the decision trees algorithm works not only with numerical attributes values but also with discrete values. In Section IV, we investigate the decision tree induction algorithm to test whether it is possible to detect new attacks, especially the two classes R2L and U2R that remain undetectable. Our goal consists in detecting the last two classes as attacks rather than improving the percentage of the successful prediction ratio of the whole test data. If this is not the case, we should give the reasons why they are always detected as normal connections.

IV. Decision trees

A. Background

Decision trees learners trace their origins back to the work of Hunt and others in the late $1950s$ [8]. At least two seminal works are to be mentioned, those by Quinlan [12] and by Breiman et al. [3]. A decision tree is a tree that has three main components: nodes, arcs, and leaves. Each node is labeled with a feature attribute which is most informative among the attributes not yet considered in the path from the root, each arc out of a node is labeled with a feature value for the node's feature and each leaf is labeled with a category or class.

Decision trees classifiers are based on the *"divide and conquer"* strategy to construct an appropriate tree from a given learning set containing a finite and not empty set of labeled instances.

The decision tree is constructed during the learning phase, it is then used to predict the classes of new instances.

Most of the decision trees algorithms use a top down strategy; i.e from the root to the leaves. Two main processes are necessary to use the decision tree: the building process and the classification process.

Besides the construction and classification steps, many decision trees algorithms use another optional step. This step consists in removing some edges that are considered useless for improving the performance of the tree in the classification step. Pruning trees simplifies the tree since many useless edges are removed rendering complex trees more comprehensive for interpretation. In addition, a tree that is already built is pruned only when it gives better classification results than before pruning [11].

In practice, one successful method that is used for finding high accuracy hypotheses is based on pruning the rules issued from the tree constructed during the learning phase. This method is used in the C4.5rules [12] that is a companion program to C4.5.

After the building process, each attribute test along the path from the root to the leaf becomes a rule antecedent (precondition) and the classification at the leaf node becomes the rule consequence (postcondition). To illustrate the rule post pruning, let us consider the following rule generated from the tree:

`IF (protocol_type = icmp)` $\wedge$ `(count> 87) THEN` $class = smurf$

This rule is pruned by removing any antecedent whose removal does not worsen its estimated accuracy.

In addition to the advantages cited by Mitchell [11], the pruned rules have many advantages in intrusion detection. Since the rules have the "`IF ... THEN ...`" format, they can be used as a model for a rule based intrusion detection. The different C4.5 rules that are generated are concise and intuitive. Therefore, they can be checked and inspected by a security expert for further investigation. We notice that C4.5rules has interesting properties for intrusion detection since it generates a good generalization accuracy. New intrusions may appear after the building process whose forms are quite similar to known attacks that are considered a priori. Using the generalization accuracy of the rules, new attacks variations could then be detected using the different rules. Real time IDSs require short rules for efficiency. Post pruning the rules generates accurate conditions hence improves the execution time for a real time use of decision in intrusion detection.

B. Improving the classification process

While the rules are efficient for detecting intrusions and their variants, they remain limited to known attacks and normal traffic. This is because the decision trees C4.5 algorithm written by Quinlan [12] presents a drawback towards the set of instances that are not covered by any of the rules generated from the decision tree. He proposed a default class for those

instances. The default class is defined as that with most items not covered by any rule. In the case of conflict, ties are resolved in favor of the most frequent class. An example of such a classification is illustrated in Table IV.

C4.5 rule	**Meaning**
duration $<= 2$, num_failed_logins> 5 $->$ class guess_passwd	If the duration of the connection is less or equal to 2 seconds and the number of failed logins is greater than 5 then this connection (*telnet* or *rsh*) is a guessing password attack.
protocol_type $=$ icmp, src_bytes> 333 $->$ class smurf	If the protocol type is icmp and the length of the packets coming from the source is greater than 333 bytes then this connection is a smurf attack.
. . .	. . .
Default: Normal	If none of the rules matches then the current connection corresponds to a normal one.

TABLE IV

CLASSIFICATION USING THE POST PRUNED RULES.

Using this principle, a default class from the learning data set is assigned to any observed instance that may be a normal connection, known or unknown attack. This classification is useful only if it is exclusive. Since we are interested in detecting novel attacks this classification would not be able to detect new attacks that normally are not covered by any rule from the tree built during the learning step.

To overcome this problem, instances that do not have a corresponding class in the training data set are assigned to a default class denoted *new class*. Therefore, if any new instance does not match any of the rules generated by the decision tree then it is classified as a new class instead of assigning it to a default class. Let us call this algorithm enhanced C4.5.

To illustrate the effectiveness of this new classification, we conduct, in Section IV-C, our experiments on the KDD 99 database since it contains many new attacks in the test data set that are not present in the training data set as shown in Table II. On the other hand, we applied this technique to a real traffic in our laboratory network. This traffic contains some new attacks that were not available when DARPA98 was built such as the slammer worm and the different DDoS attacks. These experiments shown the effectiveness of our algorithm. We do not present them here because of space limitation (for more details, see [2], Chapter 5).

This proposal may be generalized to any problem similar to the KDD 99 contest that seeks to find new instances in the test data set where some classes should be detected as new ones but not as one of the categories listed in the training data set. The fact that new attacks are not considered is one of the reasons that does not enable the different methods applied to KDD 99 contest to predict any new attack.

C. Experimental Analysis of KDD 99

We first present the different experiments and results obtained when using the different rules generated from the standard C4.5 algorithm. Applying this algorithm, a default class from the known classes in the training data set is automatically assigned to any new instance that may not be covered by any of the different rules. In the second step, the enhanced C4.5 algorithm, as explained in Section IV-B is used to handle new instances.

Table V presents the confusion matrix for the 5 classes when using the rules from the decision trees generated by the standard C4.5rules algorithm of Quinlan [13].

Predicted Actual	%Normal	%Probing	%DoS	%U2R	%R2L
Normal(60,593)	**99.47**	0.40	0.12	0.01	0.00
Probing (4,166)	18.24	**72.73**	2.45	0.00	6.58
DoS (229,853)	2.62	0.06	**97.14**	0.00	0.18
U2R (228)	82.89	4.39	0.44	**7.02**	5.26
R2L (16,189)	81.60	14.85	0.00	0.70	**2.85**
$PSP = 92.30\%$, $CPT = 0.2342$					

TABLE V

CONFUSION MATRIX RELATIVE USING THE RULES GENERATED BY THE STANDARD C4.5RULES ALGORITHM.

From Table V, the two classes R2L and U2R are badly predicted. On the other hand, many probing and DoS instances are misclassified within the normal category. Most misclassified instances are predicted as normal. This is due to the supervised C4.5rules algorithm that assigns a default class among known classes as explained in Section IV-B. We note that the class that has the highest number of uncovered instances according to the different pruned rules in the learning data set is the normal class corresponding to the normal traffic.

Hence, if a new instance is presented that is `different` (see for instance definition 4.1 below) from all other known normal or abnormal instances in the learning step, it is automatically classified as the default class *normal*.

Definition 4.1: An instance *A* is different from all other instances present in the training data set, according to the different generated rules, if none of the rules matches this instance.

The confusion matrix obtained when we use the enhanced C4.5rules algorithm that considers the default class as a new instance is presented in Table VI.

Predicted Actual	%Normal	%Probing	%DoS	%U2R	%R2L	**%New**
Normal(60,593)	**99.43**	0.40	0.12	0.01	0.00	**0.04**
Probing (4,166)	8.19	**72.73**	2.45	0.00	6,58	**10.06**
DoS (229,853)	2.26	0.06	**97.14**	0.00	0.18	**0.36**
U2R (228)	21.93	4.39	0.44	**7.02**	5.26	**60.96**
R2L (16,189)	79.41	14.85	0.00	0.70	**2.85**	**2.20**
$PSP = (92.30 + 0.57)\%$, $CPT = 0.2228$						

TABLE VI

CONFUSION MATRIX WHEN USING THE GENERATED RULES FROM THE ENHANCED C4.5 ALGORITHM.

By using the enhanced C4.5 algorithm, the detection rate of the $U2R$ class is increased by 60.96% (corresponding to the *httptunnel* attack) which decreases the false negative rate of this class from 82.89% $(189/228)$ to $21,93\%$ $(50/228)$. The detection rate of the Probing class is also enhanced by $10,06\%$

corresponding to 413 instances which are not classified as a normal traffic but as a new class. We note that the different ratios presented in Table VI are the same as those in Table V except the normal column where the corresponding ratios have decreased from Table V to VI. This is expected since the normal class is the default class, whereas in the second experiment all the instances that are classified using the default class are classified in the new class.

We should mention that the highest ratio for the U2R class has never exceeded 14% according to the different results available in the literature. Using our approach, this attack class is detected as an abnormal traffic with a detection rate of 67.98%. The false positive rate is increased by a small ratio corresponding to 24 instances (0.04%). However, the false negative rate of the R2L class remains stable.

We also performed two different tests to check the coherence of the learning and test databases of KDD 99.

In the first case, we use the default training data set of KDD 99 as the training data set and in the second test we use the test data set as the training set. In each test, we examine the percentage of successful prediction (PSP) using the learning data set of each test as a test set. The objective of this analysis is to help us discover whether the two data sets (learning and test data sets) are incoherent. Therefore, the different prediction ratios of the different data sets may help us to find out whether the enhanced C4.5 algorithm we proposed is inefficient or the different KDD 99 data sets present some anomalies such as incoherence.

Definition 4.2: A database is said coherent if all the training instances characterized by the same attributes' values belong to the same class. It is said incoherent if there are at least two instances having the same attributes values but different classes.

Table VII presents the confusion matrix obtained from testing the enhanced C4.5 algorithm over the training data set as a learning and a testing data set.

Predicted / Actual	%Normal	%Probing	%DoS	%U2R	%R2L	**%New**
Normal(97,278)	**99.94**	0.01	0.00	0.00	0.00	**0.05**
Probing (4,107)	0.17	**99.78**	0.00	0.00	0.00	**0.05**
DoS (391,458)	0.00	0.00	**99.99**	0.00	0.00	**0.01**
U2R (52)	1.92	1.92	0.00	**90.39**	0.00	**5.77**
R2L (1,126)	0.62	0.00	0.00	0.09	**98.93**	**0.36**
$PSP = 99.99\%$						

TABLE VII

CONFUSION MATRIX OBTAINED USING THE ENHANCED C4.5 ALGORITHM ON THE INITIAL KDD 99 LEARNING DATABASE.

We notice that the different classes are predicted with high rates using the learning database to construct the tree and to generate the different rules. The successful prediction ratio is $PSP = 99.99\%$.

In the field of supervised machine learning techniques, a method is said powerful if it learns and predicts the different instances of the training set with a low detection error and then generalizes its knowledge to predict the class of new instances. Unfortunately, the C4.5 induction algorithm has efficiently learned the different instances of the training set, according to Table VII, but could not classify new instances, for the moment, into their appropriate category according to bad results that are reported in Table V.

We also examined in details the classification of the new instances belonging to the $R2L$ class presented in Table II; namely {*named, sendmail, snmpgettattack, snmpguess, worm, xlock, xsnoop*}. Table VIII presents the confusion matrix corresponding to these new R2L attacks in the test data set.

Predicted / Actual	%Normal	%Pro-bing	%DoS	%U2R	%R2L	**%New**
named (17)	70.59	0.00	0.00	0.00	**0.00**	**29.41**
sendmail (17)	100	0.00	0.00	0.00	**0.00**	**0.00**
snmpget-attack(7,741)	100	0.00	0.00	0.00	**0.00**	**0.00**
snmpguess (2,406)	99.88	0.04	0.00	0.00	**0.00**	**0.08**
worm (2)	100	0.00	0.00	0.00	**0.00**	**0.00**
xlock (9)	100	0.00	0.00	0.00	**0.00**	**0.00**
xsnoop (4)	50.00	0.00	0.00	25.00	**25.00**	**0.00**
$PSP \simeq 0.00\%$ ($PSP \simeq 0.00\%$)						

TABLE VIII

CONFUSION MATRIX RELATIVE TO NEW R2L ATTACKS USING THE ENHANCED C4.5 ALGORITHM.

From Table VIII, there is only one instance of type *xsnoop* that is classified properly as $R2L$ attacks and another in the $U2R$ class and one instance of type *snmpguess* is classified as a probing attack and these are common results of the two algorithms standard C4.5 and enhanced C4.5. However, there are only two instances of type *snmpguess* that are classified as new attacks and five others of type *named*.

All the remaining instances concerning the new R2L attacks are predicted as normal connections, i.e $10,186$ (resp. $10,193$) using the enhanced C4.5 algorithm (resp. the standard C4.5 algorithm).

The false negative rate of the new R2L attacks present in the test data set is about 99.10% (resp. 99.97%) for the enhanced C4.5 algorithm (resp. the standard C4.5 algorithm).

These results show that these new R2L connections are not *distinct* from the normal connections issued after transformation done by MADAM/ID.

In the second test, we invert the two databases.Using the standard and the enhanced C4.5 algorithms, we obtained the confusion matrix presented in Table IX.

Predicted / Actual	%Normal	%Probing	%DoS	%U2R	%R2L	**%New**
Normal(60,593)	**98.34**	0.02	0.03	0.01	1.50	**0.11**
Probing (4,166)	0.19	**99.35**	0.07	0.00	0.00	**0.38**
DoS (229,853)	0.01	0.00	**99.99**	0.00	0.00	**0.00**
U2R (228)	2.19	0,00	0.00	**96.93**	0.00	**0.88**
R2L (16,189)	36.40	0,02	0.01	0.05	**63.33**	**0.19**
$PSP = 97.70\%$						

TABLE IX

CONFUSION MATRIX RELATIVE TO FIVE CLASSES USING THE RULES GENERATED BY THE ENHANCED C4.5 ALGORITHM OVER THE LEARNING DATABASE OF THE SECOND TEST.

0,udp,snmp,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00, 0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,snmp,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00, 0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.

TABLE X

SNMPGETATTACK ATTACK AND NORMAL CONNECTION SIMILARITY.

Although the percentage of successful prediction rate, from confusion matrix IX, is $PSP = 97.70\%$, it is considered very low since it consists in classifying the known labeled instances of the learning data set. This rate is considered very low in the machine learning domain because it could not learn the instances whose classes are known a priori. This means that the C4.5 algorithm failed to learn instances with their appropriate labels. On the other hand, the R2L class is highly misclassified. The classifier has learned only 63.33% from all the R2L labeled instances.

Most misclassified R2L instances are predicted as normal connections. This result justifies our observation stated in the first test: i.e. after transformation, the new R2L attacks are not distinct from the normal connections.

Since there are similarities between many attack connections and many normal connections, the question one has to ask is why different attacks have the same attributes as those of the normal connections? The corresponding *tcpdump* traffic of the different attacks is similar to that of normal connections or the transformation done over these data sets is incorrect?

All instances of *snmpgetattack* are predicted as normal (within R2L class in Table VIII). Indeed, the *snmpgetattack* traffic is recognized as normal because the attacker logs in as he were a non malicious user since he has guessed the password. Table X shows that the connections corresponding to the *snmpgetattack* are the same as those of the normal traffic. However, the *snmpguess* category should be recognized as a new attack or as a dictionary attack. Unfortunately, there is not any attribute among the 41 attributes to test the SNMP community password in the SNMP request as it is the case with some attributes that verify if it is a root password or a guest password. This is considered only in the case of *telnet*, *rlogin*, etc., services. The corresponding connections of the *snmpguess* category are the same as those of the normal traffic after transformation using MADAMA/ID programs [10]. Hence, some interesting information, with which we might have distinguished the traffic, generated by the *snmpguess* attack with the normal traffic is lost after transformation. We set necessary conditions that should be satisfied by a rich transformation function to prevent these similarities (for more details, see [2]).

V. CONCLUSION

In this paper, we investigated two different techniques for anomaly intrusion namely neural networks and decision trees. These two techniques fail to detect new attacks that are not present in the training data set. We improve them for anomaly intrusion detection and test them over the KDD 99 data sets and over real network traffic in real time. While the neural networks are very interesting for generalization and very poor for new attacks attack detection, the decision trees have proven their efficiency in both generalization and new attacks detection. The results obtained with these two techniques outperform the winning entry of the KDD 99 data intrusion detection contest. Another interesting point done here is the introduction of the new class to which new instances should be classified for anomaly intrusion detection using supervised machine learning techniques. Since the different MADAM/ID programs [10] are not available and present many shortcomings, we have written the different programs that transform *tcpdump* traffic into connection records. The objective of our contribution in this paper is twofold. It first consists in extending the notion of anomaly intrusion detection by considering both normal and known intrusions during the learning step. The second is the necessity to improve machine learning methods by adding a new class into which novel instances should be classified since they should not be classified as any of the known classes present in the learning data set. As future work, we are investigating the use of this technique with explicit or semi explicit alert correlation tools. Since these tools do not deal with unknown attacks, we are currently investigating their extension to handle these new attacks generated by the new anomaly detection to integrate them in the ongoing correlation attack scenarios.

ACKNOWLEDGMENT

This work was funded by the RNRT OSCAR and ACI DADDi projects.

REFERENCES

[1] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James. P. Anderson Co., Fort Washington, Pennsylvania, 1980.

[2] Y. Bouzida. Principal Component Analysis for Intrusion Detection and Supervised Learning for New Attack Detection. PhD Thesis, March 2006.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. 1984.

[4] J. Cannady. Artificial Neural Networks for Misuse Detection. In *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*, Arlington, VA, USA, October 5-8 1998.

[5] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Symposium On Research in Computer Security and Privacy*, Oakland, CA, May 1992.

[6] D. Denning. An Intrusion Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.

[7] C. Elkan. Results of the KDD'99 Classifier Learning. *ACM SIGKDD*, 1:63–64, 2000.

[8] E. B. Hunt. *Concept Learning: An Information Processing Problem*. Wiley, 1962.

[9] KDD 99 Task. Available at: `http://kdd.ics.uci.edu/databases/kddcup99/task.html`, 1999.

[10] W. Lee. A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems. PhD Thesis, June 1999.

[11] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[12] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:1–106, 1986.

[13] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.