Improving traffic transformation function to detect novel attacks

Amine Bsila*, Sylvain Gombault*, Abdelfateh Belghith**

*ENST Bretagne 2, rue de la Châtaigneraie F-35576, Cesson-Sévigné, FRANCE
Amine.Bsila@enst-bretagne.fr
Sylvain.Gombault@enst-bretagne.fr

**Ecole Nationale des Sciences de l’Informatique, Campus Universitaire Manouba, Manouba 2010,
TUNISIE
Abdelfateh.Belghith@ensi.rnu.tn

Abstract: Most current intrusion detection systems are signature based. The major limitation of this technique is its incapacity to detect new attacks, which by definition cannot be in the database of signatures. It is important to be able to detect this type of attack, because they mean that the attacker has new means to bypass information system protections. Only, the implementation of several anomaly detection methods make possible, in theory, the detection of these new attacks. Numerous researches worked on the transformation of the DARPA 98 traffic into KDD 99 intrusion detection data set. These researches revealed many limitations of this transformation. We extend recent work that proposed the most efficient machine-learning algorithm based on decision trees and suggest an improvement of the transformation to discover known and unknown attacks. Experimental results prove that the suggested method succeeded in the detection of new attacks and exceeded previous work.

Key words: Intrusion Detection, Decision Tree, KDD 99, and traffic transformation function.

INTRODUCTION

Intrusion Detection Systems (IDSs) have become a major focus of computer scientists and practitioners as computer attacks have become an increasing threat to commercial business as well as our daily lives. Researches have developed two main approaches for intrusion detection: misuse and anomaly intrusion detection. Misuse consists of representing the specific patterns of intrusions that exploit known system vulnerabilities or violate system security policies. Then monitoring activities of such patterns, and report the matches. In fact most commercial and open source intrusion detection systems are misuse based ones.

On the other side, anomaly detection assumes that all intrusive activities are necessarily anomalous. This means that if we could establish a normal activity profile for a system, we could, in theory, flag all system states varying from the established profile as intrusion attempts.

These two kinds of systems have their own strengths and weaknesses. The former can detect known attacks with a very high accuracy via pattern matching on known signatures, but cannot detect novel attacks because their signatures are not yet available for pattern matching. The latter can detect novel attacks but in general for most such existing systems, have a high false alarm rate because it is difficult to generate practical normal behavior profiles for protected systems. In this paper, we only consider anomaly detection systems, 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, and particularly decision trees. These techniques have proven their efficiency in predicting the different classes of the unlabeled data in the test data set for the KDD99 intrusion detection contest. Since machine-learning techniques, generally, cannot detect classes not previously seen in training
Specifically, "a connection is a sequence of TCP packets" in the TCPdump file is summarized into connections. [Hettich and Bay 99]. Therefore, packet information in the improved decision trees algorithm and the modified algorithm. 

The rest of the paper is organized as follows. Section 1 presents the KDD 99 intrusion detection cup dataset. Section 2 introduces machine-learning techniques, particularly decision trees, to handle new instances that are not considered in all current supervised machine learning techniques. Using the improved decision trees presented in Section 2, Section 3 describes the experimental results obtained, using the decision trees algorithm and the modified algorithm. This results obtained with the enhanced algorithm over KDD99 do not correspond to what we expect. This is due, in reality, to the transformation of DARPA 98 to KDD 99. Section 4 explains the problems in the transformation and suggests a solution to the problem.

1. KDD dataset

The KDD 99 intrusion detection datasets are based on the 1998 DARPA initiative, which provides designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies [MIT.L.L 98]. To do so, a simulation is made of a fictitious military network consisting of three 'target' machines running various operating systems and services. Additional three machines are then used to spoof different IP addresses to generate traffic. Finally, there is a sniffer that records all network traffic using the TCP dump format. The total simulated period is seven weeks. Normal connections are created to profile that expected in a military network and attacks fall into one of four categories: User to Root; Remote to Local; Denial of Service; and Probe [MIT.L.L 98].

• Denial of Service (DoS): Attacker tries to prevent legitimate users from using a service.

• Remote to Local (R2L): Attacker does not have an account on the victim machine, hence tries to gain access.

• User to Root (U2R): Attacker has local access to the victim machine and tries to gain super user privileges.

• Probe: Attacker tries to gain information about the target host.

In 1999, the original TCPdump files were pre-processed for utilization in the Intrusion Detection System benchmark of the International Knowledge Discovery and Data Mining Tools Competition [Hettich and Bay 99]. To do so, packet information in the TCPdump file is summarized into connections. Specifically, “a connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol” [Hettich and Bay 99]. This process is completed using the Bro IDS [Paxon 99], resulting in 41 features for each connection. Features are grouped into four categories:

• Basic Features: Basic features can be derived from packet headers without inspecting the payload. This includes features such as duration or protocol name;

• Content Features: Domain knowledge is used to assess the payload of the original TCP packets. This includes features such as the number of failed login attempts;

• Time-based Traffic Features: These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval;

• Host-based Traffic Features: Utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

The KDD 99 intrusion detection benchmark consists of three components, which are detailed in Table 2. In the International Knowledge Discovery and Data Mining Tools Competition, only “10% KDD” dataset is employed for the purpose of training [Hettich and Bay 99]. This dataset contains 22 attack types and is a more concise version of the “Whole KDD” dataset. It contains more examples of attacks than normal connections and the attack types are not represented equally. Because of their nature, denial of service attacks account for the majority of the dataset. On the other hand the “Corrected KDD” dataset provides a dataset with different statistical distributions than either “10% KDD” or “Whole KDD” and contains 14 additional attacks. The list of class labels and their corresponding categories for “10% KDD” are detailed in Table 2. Since “10% KDD” is employed as the training set in the original competition, we performed our analysis on the “10% KDD” dataset.

The task was to predict the value of each connection (normal or one of the above attacks categories) for each of the connection record of the test data set containing 311,029 connections.

It is important to note from Table 2, that:

The test data set has not the same probability distribution as the training data set;

The test data includes some specific attack types that are not present in the training data. There are 22 different attacks types out of 39 present in the training data set. The remaining attacks are present in the test dataset with different rates towards their corresponding categories.
Table 1. Basic characteristics of the KDD 99 intrusion detection datasets

<table>
<thead>
<tr>
<th>Data set</th>
<th>DoS</th>
<th>Probe</th>
<th>U2r</th>
<th>R2L</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>“10% KDD”</td>
<td>391458</td>
<td>4107</td>
<td>52</td>
<td>1126</td>
<td>97277</td>
</tr>
<tr>
<td>“KDD Corrected”</td>
<td>229853</td>
<td>4166</td>
<td>70</td>
<td>16347</td>
<td>60593</td>
</tr>
<tr>
<td>“Whole KDD”</td>
<td>3883370</td>
<td>41102</td>
<td>52</td>
<td>1126</td>
<td>972780</td>
</tr>
</tbody>
</table>

Table 2. The different attack types and their corresponding occurrence number respectively in the training and test data sets

<table>
<thead>
<tr>
<th>Probing (4,107;4,166)</th>
<th>Dos (391,458;229,853)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ipsweep (1,247;306), mscan (0; 1,053), nmap (231; 84), portsweep (1,040; 364), saint (1,589; 1,633).</td>
<td>Apache2 (0; 794), back (2,203; 1,098), land (21; 9), mail bomb (0; 5,000), Neptune 107,201; 58,001, pod (264; 87), process table (0; 759), smurf (280, 790; 164,091), teardrop (979; 12), udpstrom (0; 2).</td>
</tr>
<tr>
<td>U2R (52;228)</td>
<td>R2L (1,126; 16,189)</td>
</tr>
<tr>
<td>Buffer_overflow (30,22), httptunnel (0; 158), loadmodule (9,2), perl (3,2), root kit (10,13), pHs(0;16), Sqlattack (0,2), xterm (0;13).</td>
<td>ftp write (8,3), guess passwd (53,4,367), 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), xclock (0,9), xsnoop (0,4).</td>
</tr>
</tbody>
</table>

2. Decision trees

Decision tree induction has been studied in details in both areas of pattern recognition and machine learning. In the vast area concerning decision trees, also known as classification trees or hierarchical classifiers, at least two seminal works are to be mentioned, those by Quinlan [Quinlan 86] and those by Breiman and al. [Breiman and all 84]. The former synthesizes the experience gained by people working in the area of machine learning and describes a computer program called ID3, which has evolved in a new system, named C4.5 [Quinlan 93]. The latter originated in the field of statistical pattern recognition and describes a system, named CART (Classification And Regression Trees), which has mainly been applied to medical diagnosis. 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. 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 trees

2.1. Building process

It consists of building the tree by using the labeled training dataset. An attribute is selected for each node based on how it is more informative than others. Leaves are also assigned to their corresponding class during this process.

We use the C4.5 algorithm [Quinlan 93] to construct the decision trees where Shannon Entropy is used to measure how informative is a node. The selection of the best attribute node is based on the gain ratio (1) where (S) is a set of records and (A) a non-categorical attribute. This gain defines the expected reduction in entropy due to the partition of the data on A. It is calculated as the following [Mitchell 97]:

\[
Gain (S, A) = \text{Entropy} (S) - \sum_{v \in \text{Value} (A)} \frac{|S_v|}{|S|} \text{Entropy} (S_v)
\]

In general, if we are given a probability distribution P = (p₁, p₂, pₙ) then the information conveyed by this distribution, which is called the Entropy of P is:

\[
Entropy (P) = - \sum_{i=1}^{n} p_i \log_2 p_i
\]

If we consider only Gain (A) then an attribute with many values will be automatically selected. One solution is to use Gain Ratio instead [Quinlan 86].

\[
\text{GainRatio}(S, A) = \frac{Gain(S, A)}{\text{Split Information}(S, A)}
\]
Where

\[ Split \ Information(S, A) = -\sum_{i} \left( \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \right) \] (4)

Where \( S \) is a subset of \( S_i \) for which \( A \) has a value \( v_i \).

2.2. Classification process

A decision tree is important not because it summarizes what we know, i.e. the training set, but because we hope it will classify correctly new cases. Thus, when building classification models, one should have both training data to build the model and test data to verify how well it actually works. New instances are classified by traversing the tree from up to down based on their attribute values and the node values until one leaf is reached that corresponds to the class of the new instance.

2.3. Improving the classification process

The decision trees C4.5 algorithm written by Quinlan presents a drawback toward the set of instances that are not covered by any of the rules generated from the decision tree. He proposes a default class for those instances. The default class is defined as the one with most items not covered by any rule. In the case of conflict, ties are resolved in favor of more frequent classes.

Using this principle, a default class from the learning data set is assigned to any observed instance that may be normal, known or unknown attack.

The default class is assigned to any new instance, which is not covered by any rule generated from the training data set. This classification is useful only in the case of an exclusive classification; i.e. there is a class for any given instance and the assigned class has at least one instance in the learning data set. Since we are interested in detecting novel attacks this classification kind 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 solve this problem, Bouzida [Bouzida & al, 05] introduced the following principle: a default class denoted new class is assigned to any uncovered or unseen instance. These results are published in [Quinlan 93].

The new C4.5 algorithm has increased the detection rate of the U2R class by 60.96%, which decreases the false negative rate of this class from 82.89% to 21.93%. The detection rate of 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, hence as a new attack.

3. Experiments on KDD 99

We present the different results and experiment obtained when directly applying the C4.5 algorithm and the methods discussed in section 2.3 on the different KDD 99 cup data sets.

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

\[ PSP = \frac{number \ of \ successful \ instance \ classification}{number \ of \ instances \ in \ the \ test \ set} \] (5)

Table 3 presents the confusion matrix for the 5 classes when using the rules from the decision trees generated by standard C4.5 algorithm of Quinlan [Quinlan 93].

From table 3, the two classes R2L and U2R are badly predicted. On the other hand many Probing and DoS instances are misclassified within the normal class. Most misclassified instances are predicted as normal. This is due to the supervised C4.5 algorithm that assigns a default class among known classes as explained in Section 2.

To illustrate the effectiveness of this new idea, in section 3, we present experiment done on the KDD99 database since it contains many new attacks in the test data set that are not present in the training data.

<table>
<thead>
<tr>
<th>Predicted as Actual</th>
<th>%Normal</th>
<th>%Probing</th>
<th>%DoS</th>
<th>%U2R</th>
<th>%R2L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>99.47</td>
<td>0.40</td>
<td>0.12</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>Probing</td>
<td>18.24</td>
<td>72.73</td>
<td>2.45</td>
<td>0.00</td>
<td>6.58</td>
</tr>
<tr>
<td>DoS</td>
<td>2.62</td>
<td>0.06</td>
<td>97.14</td>
<td>0.00</td>
<td>0.18</td>
</tr>
<tr>
<td>U2R</td>
<td>82.89</td>
<td>4.39</td>
<td>0.44</td>
<td>7.02</td>
<td>5.26</td>
</tr>
<tr>
<td>R2L</td>
<td>81.60</td>
<td>14.85</td>
<td>0.00</td>
<td>0.70</td>
<td>2.85</td>
</tr>
</tbody>
</table>

PSP = 92.30 %

Table 3. Confusion matrix relative to five classes using C4.5 algorithm

Consequently, if a new instance is presented, it is automatically classified as the default class normal since it has the highest number of uncovered instances.

Table 4 shows the confusion matrix obtained when using the modified C4.5 algorithm that affect a class labeled new to any uncovered or unseen instance. These results are published in [Bouzida & al, 05].
Table 4. Confusion matrix when using the generated rules from the enhanced C4.5 algorithm

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 this approach this attack is detected as an abnormal traffic with a detection rate of 67.98%.

However, the false negative rate of the R2L class remains stable. Most R2L instances are predicted as normal connections. In the following, we explain why this class is misclassified in the normal class. The main reason is the transformation done over DARPA 98 to obtain KDD 99 where most attacks of type R2L in the test data set are not different from many normal connections in the training data set.

Table 5. Confusion matrix relative to new R2L attacks using the enhanced C4.5 algorithm.

In the following we examine in details the classification of the new instances belonging to the R2L class presented in Table 2; namely {named, sendmail, snmpgetattack, snmpguess, worm, xlock, xsnoop}. Table 5 presents the confusion matrix corresponding to these new R2L attacks in the test data set.

From Table 5, there is only one instance of type xsnoop that is classified properly as R2L attack 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 modified 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. These results show that these new R2L connections are not distinct from the normal connections issued after transformation.

4. Improving the KDD 99 transformation

In this section we investigate the two attacks snmpguess and snmpgetattack and show why they are similar to the normal traffic.

The new two attacks snmpguess and snmpgetattack that are present only during the two test weeks correspond in reality to an attack scenario [Kendall 99]. In this scenario an attacker guesses the SNMP [Case ET all. 90] community password and then remotely monitors router activity. The SNMP password is set to public by default, and is often never changed from this default value.

In the DARPA98 data sets, the SNMP community password remains by default "public". In the first day of the first test week, there is an attack against an internal router by sending SNMP request to that router using different consecutive passwords until receiving a response from that router indicating that the password is correct. This attack is similar to the dictionary attack for password guessing. We should mention that the attacker sent more than 30,000 SNMP requests, in the DARPA 98 tcpdump traffic, to find out the correct password. This attack corresponds to snmpguess that is considered as an R2L attack representing 26.75% (4,367/16,189) connections in the R2L class in the KDD 99 test data set. Once the attacker has guessed the password, he may easily monitor the router without being detected. Moreover, this attacker came back many times to monitor this community. The attacker monitoring traffic corresponds to the R2L snmpgetattack in the KDD 99 database that has 47.82% (7,741/16,189) of the whole R2L connections in this test data set. All instances of snmpgetattack are predicted as normal this result is expected. In fact, this traffic is recognized as normal because the attacker logs in as if he were a non malicious user since he has guessed the password. However, the snmpguess category should be recognized as a new attack or as a dictionary attack. Unfortunately, there is no 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. 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. This situation suggested us to find attributes that can distinguish between the two traffics. In the following we suggest a solution to the problem.

As explained in section 1 the transformation of DARPA 98 into KDD 99 contains 41 features for each connection. Features are grouped into four categories. For UDP, a connection begins when host A sends a packet to host B for the first time; this is termed a request, even if in fact the application protocol being used is not based on request and replies. If B sends a packet back, then that packet is termed a reply. By consequence the connection is considered as finished and the duration of the connection is considered as null.

In the case of the snmpguess attack, the attacker sends infinity of SNMP request with various community name and the victim replies for each one, by sending an empty SNMP message. Each couple of request reply is considered as a SNMP connection independently from the others. We introduce the notion of an SNMP session that makes the link between these SNMP connections. SNMP session is identified by:

- IP source address
- IP destination address
- Destination port (161)

To differentiate the SNMP attack traffic from that normal we used the two attributes “num_failed_login” and "logged_in" which belong to the 41 attributes of the transformation function. The first one, “num_failed_login”, count the number of failed login in a session and the second one, "logged_in", indicate that the user of the session in progress presented the good password or not. These two attributes were only used for rlogin, ftp and telnet session and have zero value for other session types. In addition these two attributes belong to the second class attributes that always have zero value for UDP traffic.

Our contribution consists in modifying the values of these two attributes for the SNMP traffic. To do this we have to modify the transformation function. Precisely we have to introduce new events, in the Bro event system, representing SNMP messages.

For the “num_failed_login” parameter that counts the number of times an attacker gives a bad password. The value is incremented when the attacker gives a bad community name this is detected when the victim answers by an empty SNMP message (empty SNMP response event).

For the "logged_in" parameter the value is set at 1 when the attacker gives the good community name, i.e., when we detect that the victim answers by a non-empty SNMP message (SNMP response event).

Other events, for example New SNMP session, SNMP request, etc., are added in order to detect SNMP messages.

After having found the right community name, the attacker will observe the community. SNMP traffic generated by the attacker will then be regarded as pertaining to the same SNMP session where the attacker guessed the good password. SNMP records will have by consequence the same value of the number of failure login attribute.

Figures 1 and 2 show the modification done on the transformation function. The two figures illustrate the list of the 41 attributes resulting from the application of the two transformation functions, the original one for figure 1 and the modified for figure 2, on Snmpguess attack traffic where the attacker tries 641 times an incorrect community name then succeeded in guessing the good one. Figure 1 show that with the original transformation function the two attributes “num_failed_login” and "logged_in" have zero value. But with the modified transformation function the value are (641,0) when the attacker has not yet guessed the good community name and (641,1) when he has.

Other events, for example New SNMP session, SNMP request, etc., are added in order to detect SNMP messages.

Other events, for example New SNMP session, SNMP request, etc., are added in order to detect SNMP messages.

5. Results

We present the different results and experiment obtained when directly applying the methods discussed in section 4 on the DARPA 98 traffic data set.

<table>
<thead>
<tr>
<th></th>
<th>Week 8</th>
<th>Week 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu</td>
<td>Fri</td>
<td>Mon</td>
</tr>
<tr>
<td>Snmpguess</td>
<td>2946</td>
<td>0</td>
</tr>
<tr>
<td>Snmpget-attack</td>
<td>1630</td>
<td>1630</td>
</tr>
</tbody>
</table>

Table 6. Snmpguess and snmpgetattack distribution.

The new transformation is tested over the two weeks of DARPA 98 test data set, which contain, as
indicated in section 2, the two attacks snmpguess and snmpgetattack. Table 6 presents the distribution of these attacks over the two weeks.

From table 6, the snmpguess attack is present only in the first week of tests. The number of records (2946) of this attack remains the same as in the description of KDD 99 test data set, also the same for the snmpgetattack attack which is present in the last two days of the first week and of during the second week of tests. The number of records (1630) remains the same.

To have these results, all the test traffic is regarded as a data input for the transformation, i.e. that all the file of traffic of various two weeks days are merged into only one for testing.

The classification algorithm we used for the validation of the results of the new transformation is the modified C4.5 algorithm, which is described in section 3.3. For the training data we used the KDD 99 training data set. Owing the fact that the training database does not contain snmpguess and snmpgetattack attacks, this one is not modified by the new transformation, even normal SNMP traffic of the training database is not modified by the new transformation. These reports come from tests that we carried out on the training traffic, which revealed that the recording output of both transformations are the same.

Table 7 shows the confusion matrix obtained when using the modified C4.5 algorithm that affect a class labeled new to any uncovered or unseen instance.

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Prob</th>
<th>DoS</th>
<th>U2R</th>
<th>R2L</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snmp-guess</td>
<td>0.034</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89.96</td>
<td>10</td>
</tr>
<tr>
<td>Snmp-getattack</td>
<td>6.56</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67.09</td>
<td>26.3</td>
</tr>
</tbody>
</table>

Table 7. Confusion matrix relative to snmpguess and snmpgetattack using the enhanced C4.5 algorithm.

By using the new transformation with the enhanced C4.5 algorithm, the detection rate of the snmpguess class as an R2L attack is increased by 99, 08 % which decreased the false negative rate of this class from 99,88% to 0,03%, results with the old transformation are contained in table 5. Snmpguess attacks are detected as an R2L attack at a rate of 89,96% (2650 / 2946), all these connections are detected as guess password attacks because the records generated by the new transformation are similar to those of the guess password attacks. The detection rate of the snmpgetattack is also enhanced by 93,39% which corresponds to 9134 instances which are not classified as a normal traffic but as a R2L class at rate of 67,09% and as a new class at a rate of 26.3%.

We should mention that the highest ratio for the two attacks has never exceeded 1% according to the different results available in the literature. Using our approach, these attacks are detected as an abnormal traffic with a high detection rate.

6. Conclusion

In this paper, new anomaly intrusion detection is investigated and tested over the DARPA 98 traffic database. We have proven its efficiency and its application have exceeded the winning entry of the KDD 99 data intrusion detection contest. Our contribution consists in making the transformation function of KDD99 richer in order to differentiate normal traffic from attack one. We are the first to enrich this function which was criticized for its poverty. Other work focused only on the poverty of the learning data set of KDD99, our work consists in modifying the function in order to detect SNMP attacks. This point was not taken into account during the transformation of the DARPA 98 into KDD 99 data sets. As a result these attacks traffic became identical to normal traffic after transformation. We saw that by modifying only the method of calculation of two attributes for a service we improved in a considerable way the rate of detection of the attacks on this service. We improved the transformation without adding new parameters, by using existing attributes. But in spite of these good results the transformation function has a limitation that it depends on services that are considered. As a future work, we are investigating a new universal transformation function that is more independent from services.

ACKNOWLEDGMENT

This work was funded by the French ministry of research under the ACI DADDI project.

REFERENCES


