Sensing attacks in Computers Networks with Hidden Markov Models

Davide Ariu, Giorgio Giacinto, and Roberto Perdisci

Department of Electrical and Electronic Engineering, University of Cagliari,
Piazza d'Armi, 09123 Cagliari, Italy
{davide.ariu, giacinto, roberto.perdisci} @diee.unica.it

Abstract. In this work, we propose an Intrusion Detection model for computer newtorks based on Hidden Markov Models. While stateful techniques are widely used to detect intrusion at the operating system level, by tracing the sequences of system calls, this issue has been rarely researched for the analysis of network traffic. The proposed model aims at detecting intrusions by analysing the sequences of commands that flow between hosts in a network for a particular service (e.g., an ftp session). First the system must be trained in order to learn the typical sequences of commands related to innocuous connections. Then, intrusion detection is performed by indentifying anomalous sequences. To harden the proposed system, we propose some techniques to combine HMM. Reported results attained on the traffic acquired from a European ISP shows the effectiveness of the proposed approach.

1 Introduction

The widespread diffusion of information systems in an increasing number of businesses, as well as for social and government services, requires incresing level of security. Very often, information resources are the core business of an organisation, or at least consitute one of the principal assets. The internal flow of information, and the external flow to customers and providers need to be deployed as a lightweigth service in order to be effective. As a consequence, information resources need to be easily reacheble and accessible, and the risk of misuse is increasing [18]. The adoption of best practices in the configuration and management of all the devices in the network is the first step to protect the information. Very often reported incidents in computer networks are related to the misconfiguration of:

– operating systems and the applications running on the hosts inside the network;
– routers and switches, which are the devices connecting the hosts in a local network, and the local network to the Internet;
– firewalls, which are the first line of defence used to protect a network from attempts of intrusions.

However, no matter how cleverly the network has been configured and managed, an intruder may find his path through the inevitable bugs and errors that are always present in software or may exploit legitimate services as security requires setting a trade-off bewteen protection of resources and their usability. As a consequence network analysis tools are needed to detect anomalous or intrusive traffic. These tools used to protect the network and its resources are called *Intrusion Detection Systems* (IDS). An IDS includes a set of tools that can be used to detect and stop attempts of intrusion. We can distinguish Intrusion Detection Systems between anomaly-based and misuse-based Systems. The anomaly-based approach has been the first to be developed as in principle this approach is able to detect intrusions never seen before [6]. These kind of IDS are based on a description of the normal behaviour. Starting from this description, the system classifies as anomalous all the behaviors that are different from the normal ones. Anomalous behaviors are typically related to intrusions, but they may also be related to normal activities as the definition of a good model of normal activities is far from being perfect. As a consequence, anomaly based systems may generate a very high percentage of false alarms. For this reason, the most widely used IDS model are based on misuse detection. Misuse-based systems perform a pattern matching between a set of rules (called signatures), which describes well known attacks, and currently observed patterns. If this process detects a matching between the observed behaviors and those encoded in the signatures, the system labels the observed patterns as an attempt of intrusion. It is easy to see that misuse based IDS can precisely detect known intrusions, but if the traits of attacks are only slightly modified the matching process is likely to fail. This is the case of so-called "polimorphic" attacks, where the code of the attack is changed in order to evade misuse-based IDS while retaining their malicious effect. The increasing number of these kind of attacks in recent years motivates a renewed interest on anomaly based IDS [19].

In this work, we propose an anomaly based IDS that analyzes sequences of commands exchanged between two hosts through a certain protocol (e.g., FTP, SMTP, HTTP, etc.), and produces an output score that is used to assess if the analyzed sequences are normal or anomalous. To model normal network traffic, we use Hidden Markov Models (HMM). After a sequence of commands is analysied, the HMM assigns a probability value that can be interpreted as the likelihood that the sequence is normal. By setting a threshold on this probability value, it is possible to flag anomalous traffic. However, the performances of HMM depend on the choice of the learning parameters as well as on the number of hiddden states. Thus, it may be difficult to design a model that meets the requirement of high detection rate and low false alarm rates. To solve this problem, we propose to use an ensemble of HMM created by using multiple training sets and multiple learning parameters. Experimental results show the effectiveness of the ensemble approach with respect to the use of a single model.

The paper is organised as follows. A review of the related works on stateful approaches to intrusion detection is reported in section 2. Section 3 summarises the basic concepts of Hidden Markov Models. The proposed IDS model is described in section 4, where the techniques used to design the ensemble of HMM are also reported. Experimental results related to the analysis of the FTP traffic of a European Internet Service Provider are reported in Section 5. Conclusions are drawn in Section 6.

2 Related Works

HMM have been successfully used in a numer of pattern recognition applications in the past years (e.g. Speech recognition, Motion recognition, etc.) . HMM have also been used for Intrusion Detection thanks to their ability to model time-series using a stateful approach where the role and meaning of the internal states are "hidden". In an Intrusions Detection problem these series may be sequences of events, commands or function running on a single host, or sequences of packets in a Network. The vast majority of studies that proposed HMM to implement IDS are related to host-based systems, i.e., IDS that analyzes the actions performed on a single host to detect attempts of intrusion[4][10][12][24]. The simplest way to detect an attempt of intrusion in a single host, is to analyze the log files that contain the traces of the system calls. In fact, when the goal of an intruder is to gain control of the operating system, typically it can be detected by analysing the sequence of system calls and comparing them to typical sequences observed during normal system usage [15].

The user's behavior can be described using different mechanisms of auditing. At the lower level the behavior of users is represented by the sequences of input characters, while, at higher levels, the behavior can be characterised by the sequence of input commands or by the characteristics of different work sessions (with a work session being usually defined as the set of simple operations that a user performs to carry out a more complex operation [5]).

When a sequence is evaluated by HMM, a value can be associated to the sequence, which denotes the probability that the sequence is produced by the process modeled by the HMM [21]. Also, the most likely sequence of states that generate the observed sequence of symbols can be compute. In this latter case, a database of normal sequences is needed to perform intrusion detection by the direct comparison of the sequence of states output by the HMM and the normal ones that are stored into the database [25].

To the best of our knowledge, only few works have proposed the use of HMM to analyse Network traffic [14] [11]. In addition, these works represent the traffic at the packet level using features as the source and destination ports, the values of flags, and the content of the message. Thus, according to these works, a probability value is assigned to each packet. On the other hand, in this work we propose a state model at the application level, where the traffic is characterised by the commands exchanged between hosts in the Internet.

3 Hidden Markov Models

Hidden Markov Models represent a very useful tool to model time-series, and to capture the underling structure of a set of strings of symbols. HMM is a stateful model, where the states are not observable (hidden). A probability density function is associated to each hidden state that provides the probability that a given symbol is emitted from that state. A Hidden Markov Model $\lambda = (S, V, A, B)$ is defined as (see figure 1):

- **S** $= \{S_1, ..., S_N\}$, the set of N hidden states in the model.
- **V** $= \{V_1, ..., V_M\}$, the set of M distinct observation symbols emitted from each state.
- **A** $= \{a_{i,j}\}$, a NxN matrix of transition probabilities between states, where $\{a_{i,j}\}$ is the probability of being in the state j at time $t+1$ given that we were in state i at time t

  $$\mathrm{a}_{i,j} = \mathrm{P}(\mathrm{q}_{t+1} = \mathrm{S}_j \mid \mathrm{q}_t = \mathrm{S}_i),\ 1 \leq \mathrm{i,j} \leq \mathrm{N},\ \text{where } \mathrm{q}_t \text{ is the state at time t.}$$

- The probability density function that describes the probability to emit symbols from each state of the HMM.

  $$\mathrm{b}_j(k) = \mathrm{P}(\mathrm{V}_k \mid \mathrm{q}_t = \mathrm{S}_j)$$
  $$1 \leq \mathrm{j} \leq \mathrm{N},\ 1 \leq \mathrm{k} \leq \mathrm{M}$$

- The probability of being in the state i at the beginning of the process (i.e., t=1) $\pi = \{\pi_i\}$.

  $$\pi_i = \mathrm{P}(\mathrm{q}_1 = \mathrm{S}_i)\ ,\ 1 \leq \mathrm{i} \leq \mathrm{N}$$

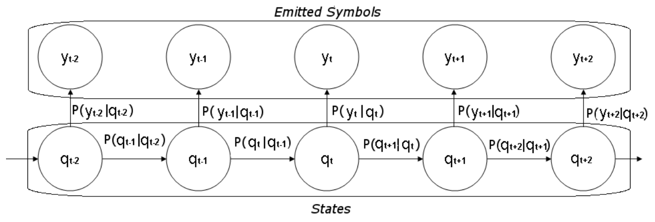


Fig. 1. The basic structure of HMM.

HMM are based on the Markov's property whereby the probability of being in a state q_{t+1} at time t+1 depends only on the state q_t at time t. Accordingly, the joint probability of observable (emitted symbols y_i) and unobservable (hidden state q_i) variables can be expressed as:

$$\mathrm{P}(\mathrm{y}_1^T,\mathrm{q}_1^T) = \mathrm{P}(\mathrm{q}_1) \prod_{t=1}^{T-1} \mathrm{P}(\mathrm{q}_{t+1} \mid \mathrm{q}_t) \prod_{t=1}^{T} \mathrm{P}(\mathrm{y}_t \mid \mathrm{q}_t)$$

The joint probability distribution is thus fully specified by: i) The initial state probability P(q_1); ii) the transition probabilities P(q_{t+1} | q_t); iii) the emission probabilities P(y_t | q_t).

3.1 Basic HMM problems

Three basic problems can be solved by Hidden Markov Models: the *Decoding* problem, the *Training* problem, and the *Evaluation* problem [22]. The *Decoding* problem is formulated as follows: given a sequence **O**, and a model λ, find the most likely sequence of states of λ that generated **O**. As this problem is not addressed in this paper, we will not provide details about it. On the other hand, we provide details of the *Training* and *Evaluation* procedures in the following subsections. Let us first describe the so-called Forward-Backward procedure, because forward and backward variables are used in the training and evaluation problems.

Forward-Backward Procedure. Let us consider the variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = S_i|\lambda),$$

This variable represents the probability of observing the sequence $\{O_1, O_2, ..., O_t\}$, given the model λ, and that the state variables at time t is $q_t = S_i$. The procedure of estimation of $P(O|\lambda)$ is made up of three steps:

1. *Initialization* $\alpha_1(i) = \pi_i b_i(O_1)$, $1 \leq i \leq N$. This step initializes the forward probability α as the joint probability of the state S_i and the initial observation O_1.
2. *Induction* $\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}] \cdot b_j(O_{t+1})$
$1 \leq t \leq T-1, \quad 1 \leq j \leq N$.
3. *Conclusion* $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

The backward probability is computed in a similar way. The backward probability is defined as the probability that the last symbol of a sequence O_T is preceded by the sequence of symbols O_{T-1}, O_{T-2}, until the symbol O_{t+1}. The backward variable is defined as

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T, q_t = S_i|\lambda)$$

and describes the probability of a subsequence of symbols within time t+1 and time T .
$\beta_t(i)$ can be calcultated by induction:

1. *Initialization* $\beta_T(i) = 1$, $1 \leq i \leq N$.
2. *Induction* $\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(O_{t+1})\beta_{t+1}(j)$,
t = *{T-1,T-2,...,1}*, $1 \leq i \leq N$.

3.2 Evaluation

Given a model λ, and a sequence of symbols $\mathbf{O} = \{O_1, ..., O_T\}$, we want to compute the probability $P(O_||\lambda)$ that the sequence of symbols is emitted by the model. This probability provides a "matching value" between the model, and the sequence. This problem can be solved using the forward variables, because $P(O|\lambda)$ can be expressed as the sum of the terminal forward variables $\alpha_T(i)$:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

3.3 Training

Given a set of sequences $\{O_|\}$, we need to calculate the model λ which maximises $P(O_|| \lambda)$. In this case, the problem is to find the set of parameters (A, B, π) that maximise the Emission Probabilities $P(\{O_t\}_| | \lambda)$ of a given set of sequences $O_|$. The solution of the problem can be find through an iterative procedure aimed at finding a local maximization of $P(O|\lambda)$. One of the most widely used training procedure for HMM is the *Baum-Welch* algorithm [2], which is an *Expectation-Maximization* algorithm that computes the parameters of the model by maximizing the log-likelihood $\lambda = arg\,max\,log(P(\{O_t\}_| | \lambda))$. At each iteration, a new estimation of the parameters is performed using the probability density functions estimated at the preceding iteration. Typically the initial values of the parameters are randomly chosen. More details on the Baum-Welch algorithm can be found in [2].

4 The Proposed IDS Model

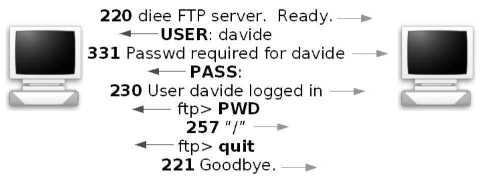
The proposed IDS aims at analysing sequences of commands exchanged between pairs of hosts, in order to assess if the sequences represent attempts of intrusion or not. To perform this analysis, three problems must be addressed:

– the length of the sequences is not known in advance.
– the correlations between the elements in the sequences are not known in advance, so that we cannot use a window of fixed length to capture correlated elements.
– the internal state of the machine responding to the commands is unknown.

The first problem does not allow designing the IDS as a deterministic finite state machine, as for these state machines we must fix the initial and final states as well as the transitions between states. On the other hand, HMM is suitable for this purpose.

The basic idea of the proposed IDS is represented in figure 2.

The sequence of events we are interested in is the sequence of commands (**USER**, **PASS**, **PWD**, etc.) and numeric codes (**220**, **231**, **257**, etc.) exchanged between hosts. In particular, we are not interested in any argument



```
220 diee FTP server.  Ready.
USER: davide
331 Passwd required for davide
PASS:
230 User davide logged in
ftp> PWD
257 "/"
ftp> quit
221 Goodbye.
```

Fig. 2. Commands exchanged between hosts

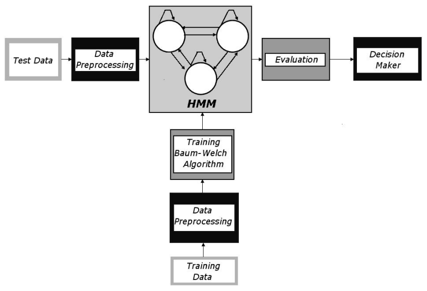


Fig. 3. The basic scheme of the proposed IDS

asssociated to the command (e.g., the command "STORE xxx" is considered as "STORE"). In order to explain the characteristics of the proposed system, we will refer to the scheme reported in Figure 3 [25].

The scheme can be explained as follows:

1. The first component, called *Data-Preprocessing*, is a module that performs a number of preliminary operations on the sequence in order to make it suitable for the HMM. During the training phase, this module extracts the dictionary of symbols to be used by the HMM. Once the dictionary is created, all the test sequences are preprocessed in order to contain only the symbols that are in the dictionary. We will go into the details of the creation of dictionaries later on.
2. HMM are built using the Baum-Welch procedure, using a set of training sequences.
3. Once the model is built, its performances are assessed by a set of test sequences, using the *Evaluation Procedure*.
4. For each test sequence, the HMM outputs a probability value stating how likely the sequence is anomalous. By setting a decision threshold, the sequence can be labeled as normal or anomalous (i.e., potentially intrusive).

4.1 Creation of the Dictionaries of Symbols

To train and test HMM, we need to create the dictionaries of symbols. Such symbols are related to the commands exchanged by hosts for a given Internet

service. Typically the number of commands defined by the RFC (i.e., the rules that define the protocols associated to services) is very large, compared to the number of commands that are actually used by applications. As a consequence, the use of a dictionary comprising all the possible commands would be ineffective as a large number of emission probabilities would be zero, and the computational load would be high. In addition, if a new command is added to the protocol, the HMM must be re-trained to take into account the new symbol. On the other hand, if we build the dictionary by using only the set of symbols of the sequences in the training set, some action must be perfomed for those test sequences that contain symbols that are not in the dictionary.
In this work, we propose to build the dictionaries by using only the symbols in the training sequences. We present two alternatives solutions for processing the test sequences that we call *Large Dictionary* and *Small Dictionary.*

Large Dictionary. The dictionary D=$\{S_1, S_2, ..., S_n, ``NaS''\}$ contains all the symbols $S_1 ... S_n$ the are present in the training sequences, plus a special symbol "NaS" (Not a Symbol). Unknown symbols in the test set are managed by replacing them with "NaS". Of course this symbol is not a command defined by the RFC, but is simply used to replace all the symbols in the test sequences that don't belong to the dictionary of commands learned from the training sequences. As an example, if the HMM is trained using the dictionary of symbols *"a","b","c","d"*, the sequence *[a-b-d-g-c-a]* cannot be analysed because there is the unknown symbol "g". If the dictionary is enlarged with the "NaS" symbol, so that the HMM is trained using the dictionary {"a","b","c","d","NaS"} , the symbol "g" into the test sequence can be replaced with "NaS": and the resulting sequence *[a-b-d-NaS-c-a]* can be analysed by the HMM.

Small Dictionary . In this second solution, we discard from the test sequences all the symbols that don't belong to the dictionary. Thus, the test sequence of the previous example *[a-b-d-g-c-a]* becomes *[a-b-d-c-a]*.

If we compare the two solutions we can observe that in the case of Large Dictionaries, all the test sequences that contain an unknown symbol are anomalous, as the symbol NaS is never encountered in the training set, and its associated probability of emission is 0. On the other hand, if the solution using a Small Dictionary is used, intrusions that contain unknown symbols cannot be detected if the sequence resulting after discarding the unknown symbols are similar to the normal ones. We can conclude that the fewer the number of erased symbols compared to the length of the sequence, the smaller the impact of the Small Dictionary solution. The use of Large Dictionaries on the other hand, allows producing an alert for each new symbol encountered. If the training set is highly representative, then the presence of an unknown symbol in a test sequence can be certainly related to some kind of anomaly.

4.2 Combination of HMM

As the performances of HMM are sensitive to the training set, and to the initial values of the parameters, in this work we explored the performances attained by combining an ensemble of HMM in order to attain low false alarms and high detection rates. To this end, we used three techniques for combining the outputs of HMM, namely:

- Arithmetic Mean
- Geometric Mean
- Decision Templates

The first two techniques simply combine the outputs by computing the average of the outputs:

$$P_{arithm}(O|\lambda) = 1/L \cdot \sum_{i=1}^{L} P(O|\lambda_i)$$

or the product of the outputs:

$$P_{geom}(O|\lambda) = \sqrt[L]{\prod_{i=1}^{L} P(O|\lambda_i)}$$

where $P(O|\lambda_i)$ is the probability that the sequence O has been emitted by the i-th HMM, and L represents the number of combined HMM.
The combination by *Decision Templates* is a more complex technique that has been first proposed in [16], and that has been used to combine HMM outputs [3] [7].

The *Decision Templates* method is based on a similarity measure between two vectors, called Decision Profile and Decision Template. The Decision Template is a vector whose elements represent the mean support given by each classifier to the N training sequences of each class. So, as in this case we are interested in modeling only one class, i.e. the normal class, the decision template represents the average of the emission probabilities of training sequences for each HMM. Let us define $dt_i(Z)$ as the average emission probability of the *i-th* HMM for the N sequences in the training set Z:

$$dt_i(Z) = 1/N \cdot \sum_{j \in Z} P(O_j/\lambda_i)$$

The Decision Template is thus defined as follows:

$$\mathbf{DT}(Z) = [dt_1(Z)...dt_k(Z)...dt_L(Z)]$$

Analougously the Decision Profile for a test sequence O_{test} is defined as follows:

$$\mathbf{DP}(O_{test}) = [P(O_{test}|\lambda_1)...P(O_{test}|\lambda_k)...P(O_{test}|\lambda_L)]$$

A soft label is then assigned to the test sequence O_{test} by means of a similarity measure between DP(O_{test}) and $DT(Z)$. We compute this similarity by the *Squared Euclidean Distance*:

$$Sim(DT(Z), DP(O_{test})) = 1 - \frac{1}{L} \sum_{i=1}^{L} (dt_i(Z) - P(O_{test}|\lambda_i))^2. \qquad (1)$$

5 Experimental Results

5.1 Dataset

The dataset used to test the proposed solution is made up of a set of sequences of FTP commands exchanged betwen a server and many clients. These sequence are extracted from the FTP traffic that is generated by the users that upload and download their resources on their own Web Space. The data were extracted from the network of the European ISP *Tiscali SpA*. The sequences of commands have been extracted by the live traffic using SNORT, a very popular open source IDS [23]. In order to filter out potentially intrusive sequences, we discarded all the sequences for which SNORT raised an alarm. The resulting dataset is made up of 40,000 sequences that have been used to train and test the HMM. First, we randomly extracted a training set made up of 80% of the traffic, the remaining 20% being used for testing. To avoid a bias in the evaluation, we repeated this subdivision five times. Thus, we created 5 different training sets, each one made up of 32,000 sequences, and 5 different test sets, each one made up of 8,000 sequences. Each of the 5 training set is further subdivided into 10 subsets (without replacement) of 3,200 sequences. Each of this sequences has been used to train distinct HMM. As a result, 50 different training set are availble for training HMMs. The main drawback of Hidden Markov Models is the computational cost of the training process, the larger the training set, the longer the training time. On the other hand, the training sets used to build Anomaly Based IDS are typically very large, so that normal activities overwhelm those anomalous events that can be present in the training traffic. In addition, the set of parameters used of a HMM trained on a large training set may not capture the structure of data. For this reason, it can be more effective to split the training set into a number of smaller subsets, and to use each subset to train different HMM. The outputs of these HMM can be then combined using the techniques outlined in the previous section, thus exploiting the information in the training set.
In order to create attack sequences, we used the simulator *IDS-Informer* [26] and added 22 attack sequences to the test set. It is worth nothing that the generation of attack sequences is not an easy task, because typically for each service a very small number of vulnerabilities can be actually exploited. This can be explained by the fact that software vendors and developers update frequently their products to correct known vulnerabilities. In addition the traffic sniffed in a network tipically contains a very small percentage of attacks. Thus, this experimental setup allows simulating a real network scenario.

5.2 Dictionaries of Symbols

In order to generate the Dictionaries of Symbols, we implemented the Large and Small Dictionaries described in the previous section. In particular, in the case of Large Dictionaries, we extracted the symbols from each of the 5 training sets made up of 32,000. Thus, in this case all the ten HMM extracted from the same training set, use the same dictionary. On the other hand, Small Dictionaries have been extracted from each of the 3,200 sequences used to train each HMM. As a result, the ten HMM extracted from the same training set, use ten different dictionaries.

5.3 Experimental setup

For each of the five Training Set, the following simulations have been performed for each of the 10 HMMs: i) HMM have been created using both the Large and the Small Dictionary; ii) three values for the number of states of HMM have been considered, namely, 10, 20, and 30; iii) two different random initializations of the initial values of the emission and transition matrixes have been performed. Thus, for a given number of states of the HMM, and for a given dictionary of symbols, 100 HMM have been created. Finally, the number of iterations for the training algorithm has been set to 100.

5.4 Performances evaluation

In order to evaluate the performances of individual HMM we decided to report the mean value and the standard deviation computed over all the 100 HMM with the same number of states and the same kind of dictionary. Combination techniques have been used to combine the 20 HMM generated for each training set. Results of combination are reported in terms of average and standard deviation computed over the five training set.

In order to evaluate the performance of the proposed IDS, we selected three measures:

- The Area under the ROC curve, where the ROC curve represents the performance of the HMM at different values of the decision threshold. In particular, the ROC curve represents the *False Acceptance Rate*, i.e., the rate of attacks classified as normal traffic Vs. the *True Positive Rate*, i.e. the rate of normal sequences classified as attacks. It is easy to see that the larger the AUC, the better the performance.
- The percentage of *real false alarms* measured on the test dataset when the *Detection Rate* is equal to 100%.
- The *Detection Rate*, when the percentage of *false alarms* measured on the training set is equal to 1%. The threshold has been calculated on the *Training Set*, so we evaluated the corresponding percentage of *false alarms* on the *Test Set*.

The second performance measure is used to assess the performance of the system in term of the number of false alarms that are produced if we wish to attain a 100% detection, while the third measure aims at assessing the performances when the false alarm rate is limited to 1%. This value typically represent an upper bound for the tolerable false positives for an IDS.

5.5 Nomenclature

Let us define some acronyms that are used in the tables where results are reported: i) **LD** and **SD** are used to denote respectively the use of a *Large* or *Small Dictionary*; ii) *10s, 20s, and 30s* are used to specify the number of states of the HMM; iii) DR is used for the *Detection Rate*; iv) FA is used for the *False Alarms* rate.

5.6 Experimental Results

Experimental results pointed out that the use of Small Dictionaries provides significantly lower performances with respect to the use of Large Dictionaries. Thus, for the experiments relates to the use of Small Dictionaries, we decided to report only the best results attained by varying the number of states of the HMM. This result has been attained by setting the number of states of HMM to 30. Table 1 shows the performances of this configuration. Reported results clearly show that high vaues of AUC can be attained by combining the HMM using the Decision Template technique. Thus, as far as the AUC is concerned, combining an ensemble of HMM allows improving the performances. However, if we analyse the False Alarm rate produced when the decision threshold is set to have a 100% Detection Rate, we can easily see that these values cannot be accepted in a real working scenario, as more than 90% of normal sequences have been classified as intrusives. In addition, the combination of HMM provides less reliable results than those provided on average by individual HMM. On the hand, if we set the decision threshold (on the training set) so that the False Alarm rate is equal to 1%, we see that the performances of combination techniques are higher than those of individual HMM, the best performance being attained by the Geometric Mean. In the following we will see that the use of Large Dictionaries allows attaining higher performances. On the other hand, as far as the training time is concerned, the use of Small Dictionaries require a shorter training time than that needed when using Large Dictionaries.

Table 1. Simulations Small Dictionary 30 States

30 States SD		DR 100%	FA1%	
	AUC	FA(real%)	DR %	FA(real%)
	mean(σ)	mean(σ)	mean(σ)	mean(σ)
Mean 100 HMM	0.873 (0.006)	89.77 (4.14)	58.72 (2.52)	0.72 (0.23)
Arithmetic Mean	0.874 (0.002)	95.27 (0.61)	63.63 (4.54)	0.31 (0.15)
Geometric Mean	0.876 (0.002)	96.19 (1.50)	76.36 (2.03)	0.71 (0.20)
Decision Templates	0.933 (0.004)	93.84 (8.40)	65.54 (3.80)	0.35 (0.16)

If we analyse the performances of HMM using Large Dictionaries reported in Tables 2, 3, and 4, it is easy to see that performances are quite superior to those attained using Small Dictionaries. In particular, performances improved significantly by increasing the number of states from 10 to 20. On the other hand a further increase in the number of states from 20 to 30 does not provide significant improvements in performance, except for the standard deviation which is smaller than that of HMM with 20 states. The values of AUC in the three cases are larger than 0.95, the combination by the arithmetic and geometric means providing the highest performances.

Table 2. Simulations Large Dictionary 10 States

10 States LD		DR 100%	FA1%	
	AUC	FA(real%)	DR%	FA(real%)
	mean(σ)	mean(σ)	mean(σ)	mean(σ)
Mean 100 HMM	0.953 (0.006)	65.15 (2.09)	85.44 (1.73)	0.80 (0.23)
Arithmetic Mean	0.958 (0.002)	76.41 (1.22)	82.72 (2.03)	2.35 (1.24)
Geometric Mean	0.961 (0.001)	74.54 (1.07)	92.72 (4.06)	2.83 (1.22)
Decision Templates	0.958 (0.002)	74.89 (2.55)	83.62 (4.06)	2.42 (1.22)

If we analyse the performances attained when the decision threshold is set so that the Detection Rate is equal to 100%, we see that the values of False Alarm rate are quite smaller than those attained using the Small Dictionaries, but still these values are not suited for a real operating environment. It is worth noting, however, that hardly any IDS is able to produce an accetable False Alarm rate when it is tuned to detect the 100% of attacks [1]. Thus the evaluation of IDS at 100% detection rate is just used to see the performances in the limit. From an operational point of view, it is more intersting to evaluate the detection rate when the false alarm rate is fixed at 1%. If we compare the values attained using 10, 20, and 30 states we can see that the detection rate increases as the number of states is increased. Again, the highest values are attained by combining HMM, reaching the value of 95.45%. If we compare this result with the false alarm rate attained at 100% detection rate, it is easy to see that a small increase in the detection rate is accompained by a very large increase in the false alarm rate. Finally, the tables also report the false alarm rate attained on the test set (FA(real%)) when the decision threshold is set to the value that produces the 1% false alarm rate on the training set. It can be seen that, apart from the case of 10 states, the false alarm rate on the test set is always smaller that 1%. Thus, the threshold estimated on the training set produces similar results on the test set.

6 Conclusions

This work proposed a novel technique to detect intrusions in computer networks, based on the analysis of sequences of commands exchanged between pairs of

Table 3. Simulations Large Dictionary 20 States

20 States LD		DR 100%	FA1%	
	AUC	FA(real%)	DR%	FA(real%)
	mean(σ)	mean(σ)	mean(σ)	mean(σ)
Mean 100 HMM	0.967 (0.004)	59.60 (4.46)	90.94 (1.27)	0.74 (0.22)
Arithmetic Mean	0.974 (0.002)	79.23 (3.02)	92.72 (6.1)	0.33 (0.16)
Geometric Mean	0.972 (0.001)	52.27 (3.06)	95.45 (0)	0.89 (0.09)
Decision Templates	0.965 (0.002)	52.34 (9.69)	95.45 (0)	0.41 (0.18)

Table 4. Simulations Large Dictionary 30 States

30 States LD		DR 100%	FA1%	
	AUC	FA(real%)	DR%	FA(real%)
	mean(σ)	mean(σ)	mean(σ)	mean(σ)
Mean 100 HMM	0.969 (0.003)	57.85 (3.57)	92.97 (0.65)	0.74 (0.16)
Arithmetic Mean	0.974 (0.0004)	55.92 (0.62)	95.45 (0)	0.53 (0.13)
Geometric Mean	0.971 (0.0008)	55.00 (1.20)	95.45 (0)	1.01 (0.10)
Decision Templates	0.962 (0.004)	86.02 (8.12)	95.45 (0)	0.62 (0.17)

hosts. In particular we modelled sequences using HMM. For each command sequence, a probability value is assigned and a decision is taken according to some predefined decision threshold. We investigated different HMM models in terms of the dictionary of symbols, number of hidden states, and different training sets. We found that good performances can be attained by using dictionary of symbols made up of all symbols in the training set, and adding a NaS (not-a-symbol) symbol in account of symbols in the test set that are not represented in the training set. Performances can be further improved by combining different HMM. As the size of training sets in an intrusion detection application is typically large, we proposed to split the training set in a number of parts, training different HMM and then combining the output probabilities by three well known combination techniques. Reported results on a real dataset extracted from the live traffic of an ISP show the effectiveness of the proposed approach.

Future works should include the fusion of information from the proposed module, which analyses sequence of commands, with information from other modules devoted, for example, to the analysis of the arguments of commands (e.g., the name of files exchanged, subject of e-mails, etc.). In fact attacks can be reliably detected when multiple analysis are performed on the network traffic, and the partial results combined. We suspect the resulting IDS will not only produce a lower false alarm rate, but also be more robust to evasion activities, as the attacker should evade the detection capabilities of multiple modules working on different traffic characteristics.

References

1. S. Axelsson, *The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection*, Proc. of RAID, May 1999.

2. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Ann. Math. Statist., vol. 41(1), pp. 164-171, 1970.
3. Manuele Bicego, Vittorio Murino, M. Figueiredo, *Similarity-Based Clustering of Sequences Using Hidden Markov Models*, MLDM 2003, 86-95.
4. S. Cho and S. Han. Two sophisticated techniques to improve HMM-based intrusion detection systems. Proc. of RAID, 2003.
5. H. Debar, M. Becker, D. Siboni, *A Neural Network Component for an Intrusion Detection System*, Proc. of the IEEE Computer Society, Symposium on Research in Security and Privacy, 1992.
6. D.E. Denning, *An Intrusion Detection Model*, IEEE Trans. Software Eng., Vol SE-13, No. 2, Feb. 1987, pp. 222-232.
7. C. Dietrich, F. Schwenker and G. Palm, *Classification of Time Series Utilizing Temporal and Decision Fusion*, MCS 2001, LNCS 2096, pp. 378-387, 2001.
8. T. Dietterich, *Ensemble Methods in Machine Learning*, MCS 2000, LNCS 1857, pp. 1-15, 2000.
9. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, Wiley-Interscience, 2000.
10. H. H. Feng, O. M. Kolesnikov, P. Fogla, W. Lee, W. Gong. Anomaly detection using call stack information. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003.
11. F. Gao, J. Sun, Z. Wei, *The prediction role of Hidden Markov Model in Intrusion Detection*, Proc. of IEEE CCECE 2003, vol. 2, pp. 893-896, May 2003.
12. D. Gao, M. Reiter, D. Song, *Behavioral Distance Measurement Using Hidden Markov Models*, Proc. of RAID, 2006.
13. G.Giacinto, F. Roli, L. Didaci, *Fusion of multiple classifiers for intrusion in computer networks*, Pattern Recognition Letters 24 (12) (2003) 1795-1803.
14. M. Hashem, *Network Based Hidden Markov Models Intrusion Detection Systems*, IJICIS, vol.6(1), January 2006.
15. X.D. Hoang, J.Hu, *An Efficient Hidden Markov Model Training Scheme for Anomaly Intrusion Detection of Server Applications Based on System Calls*, Proc. of 12th IEEE Conference on Networks, 2004. Vol.2, pp.470-474,2004.
16. L. Kuncheva,J.C. Bezdek, R.P.W. Duin, *Decision Templates for Multiple Classifier Fusion*, Pattern Recognition, vol.34(2), pp.299-314, 2001.
17. L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley, 2004.
18. Mc Hugh J., Christie A., Allen J., 2000. *Defending yourself: The role of Intrusion Detection Systems*, IEEE Software (*Sept./Oct.*), 42-51.
19. P. Fogla, M. Sharif, R. Perdisci, O.M. Kolesnikov, W. Lee, *Polymorphic Blending Attack*, USENIX Security Symposium, 2006.
20. P.E. Proctor, *Pratical Intrusion Detection Handbook*, Prentice-Hall, Upper Saddle River, NJ, 2001.
21. Y. Qiao, X.W. Xin, Y. Bin, S. Ge, *Anomaly Intrusion Detection Method Based on HMM*, Electronic Letters, Vol.38(13), June 2002.
22. L.R. Rabiner , *A tutorial on Hidden Markov Models and selected applications in speech recognition.*, Proc. of IEEE, vol.77(2), pp.257-286, February 1989.
23. M. Roesch, *Snort - Lightweight Intrusion Detection for Networks*, Proc. of the 13th USENIX conference on System Administration, LISA '99.
24. C. Warrender, S. Forrest, B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proc. of the IEEE Symp. on Security and Privacy*, 1999.
25. X. Zhang, P. Fan, Z. Zhu, *A New Anomaly Detection Method Based on Hierarchical HMM*, Proceedings of the 4th PDCAT conference, 2003.
26. IDS-Informer, www.blade-software.com