Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

ELSEVIER

Computer Networks 45 (2004) 175–193

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

# Measuring normality in HTTP traffic for anomaly-based intrusion detection

Juan M. Estévez-Tapiador *, Pedro García-Teodoro, Jesús E. Díaz-Verdejo

*Department of Electronics and Computer Technology, University of Granada, E.T.S. Ingeniería Informática, C/Daniel Saucedo Aranda, S/N, 18071 Granada, Spain*

Received 16 July 2003; received in revised form 26 November 2003; accepted 29 December 2003

Responsible Editor: J. Solé-Pareta

**Abstract**

In this paper, the problem of measuring normality in HTTP traffic for the purpose of anomaly-based network intrusion detection is addressed. The work carried out is expressed in two steps: first, some statistical analysis of both normal and hostile traffic is presented. The experimental results of this study reveal that certain features extracted from HTTP requests can be used to distinguish anomalous (and, therefore, suspicious) traffic from that corresponding to correct, normal connections. The second part of the paper presents a new anomaly-based approach to detect attacks carried out over HTTP traffic. The technique introduced is statistical and makes use of Markov chains to model HTTP network traffic. The incoming HTTP traffic is parameterised for evaluation on a packet payload basis. Thus, the payload of each HTTP request is segmented into a certain number of contiguous blocks, which are subsequently quantized according to a previously trained scalar codebook. Finally, the temporal sequence of the symbols obtained is evaluated by means of a Markov model derived during a training phase. The detection results provided by our approach show important improvements, both in detection ratio and regarding false alarms, in comparison with those obtained using other current techniques.

*Keywords:* Anomaly detection; Application-level intrusion detection; HTTP attacks; Computer and network security

## 1. Introduction

The limitations of current formal methods and misconfigured control access policies in computer and network security (mainly aimed at *preventing* security failures), together with the increasing number of exploitable bugs in network software, have made it necessary to develop security-oriented monitors designed to supervise system activity in search of security violations. The goal, therefore, of these intrusion detection systems (henceforth referred to as IDS) is apparently a simple one: to detect intrusions in a computer network environment. Such a detection is basically performed through an analysis of system activities, in search of events of interest from a security point of view.

* Corresponding author. Tel.: +34-958-242305; fax: +34-958-240831.

*E-mail addresses:* tapiador@ugr.es (J.M. Estévez-Tapiador), pgteodor@ugr.es (P. García-Teodoro), jedv@ugr.es (J.E. Díaz-Verdejo).

Traditionally two main approaches have been adopted to the problem of intrusion detection, according to how the data collected from the supervised system are analysed: *misuse detection* and *anomaly detection*. In misuse detection, each known attack is described through the construction of a specific model, known as the attack "signature". Incoming activities that match a pattern in the library of signatures raise an alarm. In the second approach, the basic principle behind any anomaly detection system is the hypothesis that every anomalous event is suspicious. Note that "suspicious" is not strictly equal to "intrusive" or "malicious". An event can be catalogued as anomalous because its degree of deviation with respect to the profile of characteristic behaviour of the system exceeds a given threshold. Such a situation may occur because the observed event does not adequately fit the model derived from the analysis made for the creation of the system profile.

Regardless of the method used for detecting attacks, an IDS can be classified as either *host based* or *network based* depending on its source of input data. This feature imposes a division in current systems between those that use the events taking place within a given host (e.g., programs executed, user activity, etc.), and those that resort to network traffic as the main source of data. For example, Denning proposed a scheme to construct patterns related to login times and resources consumed by users and programs [1]. On the other hand, SNORT [2], BRO [3] and NETSTAT [4] are well-known examples of network-based IDSs.

Current IDSs mainly rely on misuse detection mechanisms. Anomaly-based detection is often said to be a more powerful mechanism, due to its theoretical potential for addressing novel or unforeseen attacks. Nevertheless, although these detection techniques have proliferated rapidly and detection systems have evolved significantly in recent years, this goal is still far from being satisfactorily achieved. Interested readers can find good surveys of IDSs in [5,6].

In the context of anomaly detection in a system, the need to define the notion of "normal state of operation" of the entity being supervised is, without doubt, the most crucial challenge. Without loss of generality, we may say the most significant issue concerning anomaly detection is the construction of the system model, and, to be precise, identifying which variables contain the most sensitive information for this purpose. Current computer networks are highly complex and the number of parameters involved in their dynamics is intractable. The collection of complete, reliable data is a complex task in itself, and determining what events to log and where to place the sensors is still an open question.

We present interesting results concerning the development of anomaly-based detectors for network services. Most of the research carried out in anomaly-based network IDSs can be catalogued as *network models*, whose main purpose is to model the behaviour of network traffic at the lower layers (see [7]). We propose an approach that should be included in the context of modelling the normal behaviour of specific application traffic, for instance, HTTP or DNS (see [8] for an earlier paper in this field). In this framework, we introduce:

(a) an experimental analysis of several features related to application traffic, of interest concerning anomaly detection;
(b) a new anomaly-based intrusion detection approach that uses knowledge related to the application-layer protocol and improves on the detection results obtained by other current techniques. Our approach consists of two stages, training and recognition, in each of which a mapping of every HTTP payload into a variable-length sequence of symbols is considered. In the training process, a Markov chain is derived from a predefined set of such sequences, thus modelling the "normal" behaviour of HTTP traffic. The recognition (or detection) phase seeks to identify intrusive actions by evaluating incoming HTTP traffic with the Markov model described above.

This paper is organized as follows: Section 2 provides a description of the data sets used for experimentation, and in Section 3, preliminary analytical results and empirical considerations on the nature of HTTP traffic are described. Our approach to application traffic modelling is described in detail in Section 4, detailing the work developed for HTTP and the results obtained from our experiments. Further comments concerning the proposed scheme and its application in real environments are provided in Section 5. Finally, Section 6 summarizes the paper, presenting its main conclusions and future research objectives, as well as the advantages and limitations of the approach described. Concerning the statistical tools used in this study, a briefing on the Kruskal–Wallis and Kolmogorov–Smirnov tests, as well as on Markov chains and their use in sequence recognition, is presented in Appendix A.

## 2. Data set description and evaluation framework

A major problem concerning current IDS technology is the lack of common, well-established techniques for assessing the detection capabilities of such systems. An ad-hoc methodology is found in most evaluation frameworks, but it is difficult to compare different systems and approaches [9]. The use of the same data sets is one of the most critical elements in the evaluation of a new proposal, and hence, in its subsequent comparison with pre-existing systems. The following subsection describes the data sets of network traffic used in our experiments.

Despite the absence of common evaluation frameworks, there exists a broad range of analytical tools that can be used to test specific aspects of current detection systems. For instance, Receiver Operating Characteristic (ROC) curves have been widely used to measure the performance of a detector [10]. A ROC curve plots the attack detection rate (i.e., true positives) against the false positive probability, thus providing a way of visualizing the trade-offs between detection and false positive rates. The method used to obtain such a plot may be substantially different from one system to another, although the rates can nearly always be computed by varying the detection threshold or an equivalent parameter. In addition to ROC curves, a number of analytical tools are used in the present paper. Appendix A provides a brief background concerning such statistical tools.

### *2.1. Data sets and attacks*

As stated above, regardless of the testing methodology used, similar data sets are required in order to compare the behaviour exhibited by different systems under the same conditions and events. With this aim in mind, DARPA sponsored an IDS Evaluation Program performed at MIT Lincoln Labs during 1998 and repeated in 1999 [11]. Although this framework is not without drawbacks (see [12] for an excellent critique), it is undeniably a remarkable advance.

The 1999 DARPA Evaluation Framework is composed of several off-line test sets consisting of traffic captured during 5 weeks on a network with hundreds of hosts and a connection to Internet. The data comprise *tcpdump* files for traffic captured on various network segments, audit logs recorded in the hosts, and *BSM* audit data for Solaris machines. The training data provided to adapt the systems consist of the first 3 weeks. Weeks 1 and 3 are attack-free, intended to construct anomaly-based detectors that rely on normal traffic, whilst week 2 includes both attack instances and attack-free records. Finally, the test data provided to evaluate the detection capabilities consist of weeks 4 and 5, which include several attack instances interleaved with background traffic.

In our approach to the problem of anomaly detection at the application layer, complete data sets of both normal incoming traffic as well as anomalous connections are required. To carry out our experiments, we collected normal incoming HTTP traffic from the DARPA'99 IDS Evaluation data sets, specifically from weeks 1 and 3. Since our purpose is to study HTTP traffic, the packets extracted are those corresponding to this service and destined to the two different HTTP servers existing in the DARPA'99 data sets: `hume` (NT Server with IP address `172.16.112.100`) and `marx` (Linux Server with IP address `172.16.114.50`). The total number of payloads extracted for these hosts is 32435 for `hume` and 41648 for `marx`. Each payload concerns one or more IP packets and it is reassembled, if needed, in the same way that the final server to which it is destined would do it. This task was performed using the facilities provided by the Snort intrusion detection system [2]. After this preprocessing stage, every payload is represented in our data set as a variable-length sequence of characters (bytes).

Concerning hostile traffic, the HTTP-related attacks contained in the DARPA'99 data sets do not satisfy our requirements for the evaluation, for two main reasons:

- First, attack technology has evolved significantly since 1999, especially concerning Web attacks. An analysis based exclusively on these relatively old attacks could be unpractical in current environments, which have to deal with more sophisticated threats.
- The DARPA'99 was intended to be a general-purpose evaluation framework. Although the attack instances are representative of broad categories of threats, they do not exhaustively cover HTTP attacks, as is necessary for our purposes. Thus, within the range of attacks included in the data sets, there are only two different HTTP-related attacks. It is unrealistic to perform an analysis based on this scarcely representative volume of attacks.

In spite of these severe drawbacks, it would not be desirable to abandon the idea of using the DARPA'99 data sets, because they are widely accepted and used among the research community. In our opinion, the use of these data sets wherever possible instead of using our own traffic databases makes the experiments more reproducible.

With the aim of obtaining practical results, we compiled several well-known vulnerabilities in the HTTP service in order to generate an attack data set. For this purpose, several variants of 86 HTTP exploits listed in arachNIDS database [13] were considered. Table 1 provides a brief description of 21 of them, and specifically a subset of those related to HTTP/GET requests (although the complete set includes attacks based on other request types, like HTTP/POST or HTTP/HEAD). Most of these attacks have a cross reference to the equivalent entry in other well-known vulnerability lists such as *bugtraq* (maintained by Security-

Table 1
Some of the HTTP/GET attacks used for evaluation purposes in this work

| Attack index | ArachNIDS reference | Attack name | Attack class |
|---|---|---|---|
| A1 | IDS214 | client-netscape47-overflow-unsuccessful | CC |
| A2 | IDS248 | http-frontpage-pws-fourdots | IGA |
| A3 | IDS258 | http-cgi-get32.exe | SI, IGA |
| A4 | IDS265 | http-cgi-cgitest | SI, IGA |
| A5 | IDS268 | http-coldfusion-application.cfm | IGA |
| A6 | IDS275 | dos-http-cisco-crash | DOS |
| A7 | IDS301 | http-nessus-404-check | IGA |
| A8 | IDS310 | scanner-L3retriever-http-probe | IGA |
| A9 | IDS432 | http-iis-unicode-traversal | SI |
| A10 | IDS433 | http-iis-unicode-traversal-optyx | SI |
| A11 | IDS434 | http-iis-unicode-traversal-backslash | SI |
| A12 | IDS452 | http-iis-unicode-binary | SI |
| A13 | IDS462 | web-cgi-yabb | SI, IGA |
| A14 | IDS470 | web-cgi-webplus-version | SI, IGA |
| A15 | IDS471 | web-cgi-webplus | IGA |
| A16 | IDS549 | dos-3com_sml3com | DOS |
| A17 | IDS550 | aspseek_s.cgi_access | SI, IGA |
| A18 | IDS552 | iis-isapi-overflow-ida | SI, IGA |
| A19 | IDS553 | iis-isape-overflow-idq | SI, IGA |
| A20 | IDS554 | squid-cachemgr.cgi-connection | R, IGA |
| A21 | IDS555 | frontpage_visual_studio_rad_overflow | SI |

The last column shows the class or classes to which each attack belongs (SI: System Integrity; IGA: Information Gathering Attempt; CC: Client Compromise; DOS: Denial of Service; R: Relay).

Table 2
Summary of data sets used for experimentation

| Data set | No. of payloads |
|---|---|
| Normal requests (marx) | 41,648 |
| Normal requests (hume) | 32,435 |
| Attacks | 1500 |

Focus), *CVE* (from The Mitre Corporation) or *advICE* (maintained by Internet Security Systems), for readers more familiar with some of these databases.

The attack traffic that composes the data sets used for test purposes was generated and adapted to our framework by means of several programs that implement the corresponding exploit for each attack. Some of this software has been downloaded from hacking-related sites, while other elements were developed by the authors. For assessment purposes, a total of 1500 malicious payloads (attack instances) were generated and captured with these tools, each one corresponding to a specific variant of one of the 86 attacks considered. Table 2 shows the size of the data sets used in this paper.

## 3. Measuring normality through isolated features: a case study

Although some proposals have been made to detect anomalies in application-layer traffic (see, for example, [8,14]), these works do not provide any performance measures related to the discriminating power of the features introduced when normal and anomalous traffic is analysed. As an initial step, we study the usage of two measures described in the bibliography as adequate for anomaly-based intrusion detection at the application layer: payload length and payload histograms. The experimental results obtained are then presented.

### *3.1. Payload length*

Recent works have shown that the length of a request can be a good feature for evaluating the correctness of a payload. For instance it has been used in the case of HTTP and DNS services in [8], while in [14] it is considered as a factor in the subsequent application for detecting anomalies in mail traffic. This fact is justified by the very nature of the required service input. Thus, a specific application payload destined to HTTP or DNS applications transports several strings related to the requested service, user information, etc. Under certain types of attacks, especially those referred to as *buffer overflows*, this situation can vary dramatically.

Unfortunately, and regardless of the real discriminative power of this feature, the range of service-specific attacks includes more than buffer overflows in the local server. For instance, various attacks try to subvert certain CGI scripts, or even other programs, masking the request with format strings which are not adequately processed by the server. In those cases, the payload length may not be significantly affected by the attack code carried.

This fact is empirically illustrated in Fig. 1, which shows the payload length probability density function (pdf) of the data sets mentioned in Section 2.1. Observe that the distribution is distinct for the two hosts. This length is strongly related to the specific server, and would be likely to change if certain alterations were made in the service. For example, in the current case of HTTP requests, the addition of a new web page or the modification of the name could cause a change in the probability of the request length. In the case of lengths from attack payloads, the variability is stronger. As can be easily observed, there are attacks with payload lengths ranging from a few bytes up to several thousand.

Regardless of the experimental results, which are provided below, the main conclusion is that the payload length should not be used as an isolated feature for distinguishing between normal and anomalous payloads. However, its use in conjunction with other features is shown to be a good choice [8], since it contributes information related to the normality of the payload.

### *3.2. Payload histogram*

As previously stated, in the case of certain application protocols, like DNS or HTTP, the

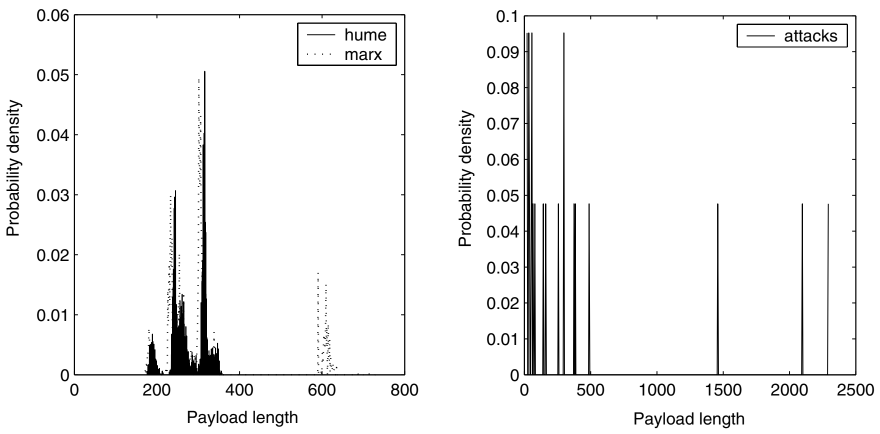

Fig. 1. HTTP payload length (in bytes) probability density for data sets corresponding to normal traffic in hosts marx and hume (left), as well as attack payloads (right).

content carried into the payload is usually composed of several character strings that store protocol-related information, like the name of the resource(s) addressed, options of the user or the client application, etc. These strings are generated from an alphabet $\Sigma$, which in this case is the ASCII code. Since the strings are partially structured according to the protocol syntax rules, it is not unreasonable to suppose that not all the elements (i.e., characters) have equal probabilities of occurrence [8].

We computed the probability density function for each of the payloads present in our data sets, grouping them by the service each one refers to (for instance, in the case of HTTP there exist several services like GET, POST, OPTIONS, etc.) [15]. A simple visual inspection reveals that, for a specific host, they are very similar. Although these histograms pass the Kruskal–Wallis test (that is, they seem to come from the same distribution; see Appendix A), it is possible to verify these results numerically by means of the following experiments.

Given a host and a specific service (for example, GET requests destined to host hume), both the mean payload histogram and the standard deviation are computed. Note that we are not working with the mean and standard deviation of the pdf, but with the mean and standard deviation for each symbol in the ASCII code. Fig. 2 shows these results for both hosts.

Besides the standard deviation, which is small enough to ensure that payloads are very similar, another way of measuring the degree of similarity between each histogram $h_i(c)$ and the mean histogram $H_{\rm m}(c)$ is by calculating the cross correlation between them for all $i$. By these means, a cross correlation vector with this value for each histogram is determined. The mean cross correlation obtained is 0.9358 (a value of 1.0 indicates that functions are the same), with a standard deviation of 0.002. Additionally, we carried out the Kolmogorov–Smirnov test (see Appendix A) in order to determine whether the mean histogram $H_{\rm m}(c)$ is a good discriminative feature. Thus, at a significance level of 0.01 (i.e., a 99% probability of acceptance of the null hypothesis that the sample follows the distribution), more than 90% of the population obtains a positive result in the test. These results are valid for the two data sets of normal traffic used (requests destined to hosts hume and marx).

These results support the hypothesis that payload histograms or certain measures derived from them are a good feature for measuring normality, as pointed out in [8]. Nevertheless, the discrimination between normal and anomalous payloads provided by the histogram is not accurate. For instance, Fig. 3 shows the histograms derived from three different attacks. While attacks 1 and 3 exhibit a pdf that is clearly different from that cor-

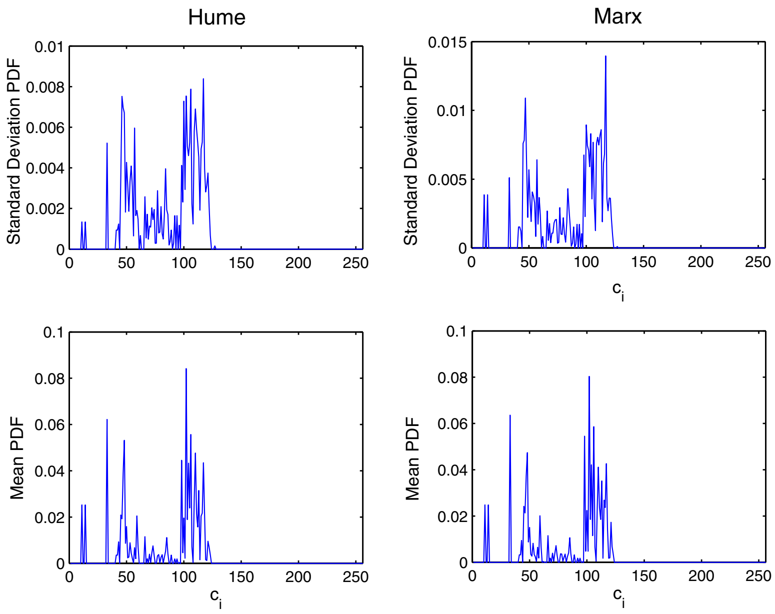

Fig. 2. HTTP/GET payload mean probability density and standard deviation computed for hosts `hume` and `marx`. The results obtained for other services like POST are similar.

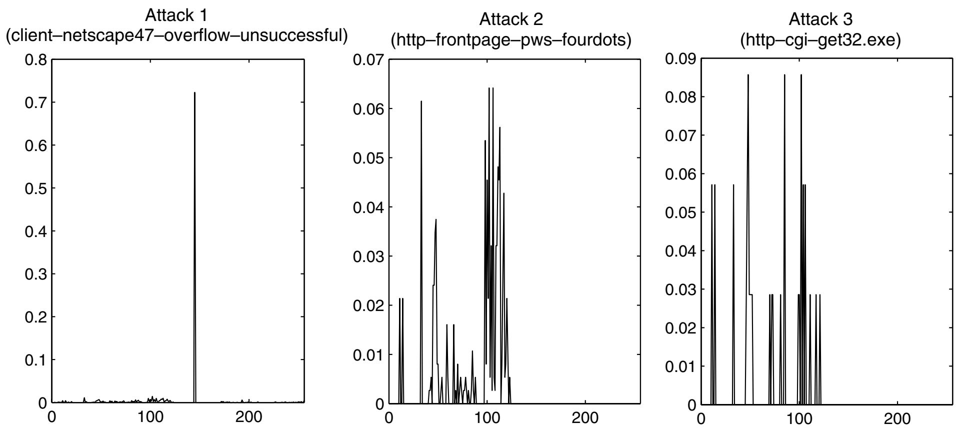

Fig. 3. Probability density functions for three different attacks.

responding to the normal behaviour, attack 2 is practically identical to the correct histograms, and this is confirmed by statistical tests.

### 3.3. Evaluation and results

After the preliminary evaluation of the two above-mentioned features, a series of experiments were carried out. For evaluation purposes, the data sets described in Section 2.1 were randomly divided into two subsets for training and testing. The training set was composed of 70% of the payloads from the normal connections, with the remaining 30% together with the attack payloads constituting the test examples. Note that, since our aim was to obtain models for anomaly detection, only traffic instances corresponding to normal behaviour were included in the training data.

The first experiment is devoted to payload length. In this case, deciding whether a given payload should be considered anomalous (detection process) is carried out by evaluating the expression proposed in [8]. Given a request with length $\ell$, an anomaly score $\mathrm{AS}_\ell$ which rises exponentially with the length, is computed as

$$\mathrm{AS}_\ell = 1.5^{(\ell-\mu)/(2.5\sigma)}, \qquad (1)$$

where $\mu$ and $\sigma$ are the parameters of the model, namely the mean and the standard deviation of the payload length among the training set. The detector performs a simple threshold comparison: if the anomaly score for a given payload is greater than this threshold, the current payload is classified as anomalous. By varying this threshold, it is possible to obtain different classification results in terms of false and true positives. The results obtained for different threshold values are shown in Fig. 4.

The second experiment employs the payload histogram as a single parameter for detection purposes. The classification of a given payload into normal or anomalous payload is carried out by means of the Kolmogorov–Smirnov test. Given the histogram $h_i(c)$ of the $i$th payload, the test is performed to determine whether it belongs to the samples generated by the model $H_\mathrm{m}(c)$ (i.e., the mean histogram obtained from the training data). Thus, the detector implements the Kolmogorov–Smirnov test to decide, with a given significance level, $\alpha$, whether the payload is anomalous or not. By changing $\alpha$ it is possible to evaluate the system under distinct false positive rates. The results obtained are also shown in Fig. 4.

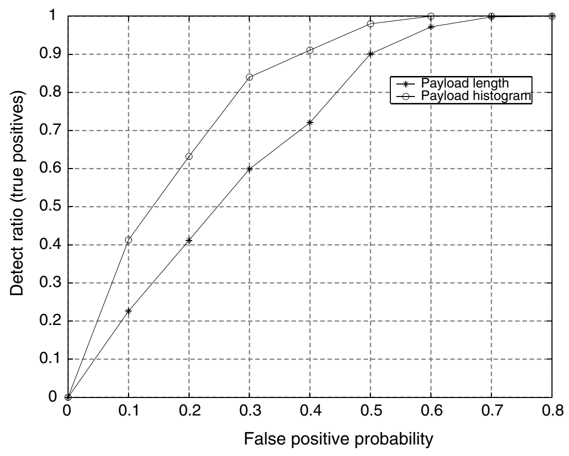

Fig. 4. ROC curves obtained for the two models corresponding to payload length and payload histogram.

The ROC curves obtained for payload length and histogram show that the best results are achieved by using the second feature. Nevertheless, the results obtained are not very accurate, since detection rates higher than 90% are achieved with false positive rates of nearly 40% and 50% for payload histogram and length, respectively. As stated previously, these features seem to contain important information for discriminating between normal and anomalous payloads, although their use in isolation does not yield very satisfactory results.

## 4. A new stochastic approach for anomaly-based intrusion detection at the application layer

In this section, we present a new stochastic approach intended to improve on the general anomaly-based intrusion detection results provided by currently used techniques. The proposal is introduced as an evolution of using single static features to characterize application traffic payloads. Information concerning the system dynamics is considered and used for the purpose of modelling its normal behaviour.

On a Markov chain basis, two variants are developed, differing by the features considered to represent the corresponding application traffic. First, an extension of the payload histograms studied in Section 3.2 is presented, after which a more sophisticated technique is discussed, in which protocol-dependent information is considered.

### *4.1. On the use of conditional probabilities for payload modelling*

The occurrence probability of a character $c_i$ in a payload, $P(c_i)$, is considered in [8] to model application-based traffic. Following this idea, we

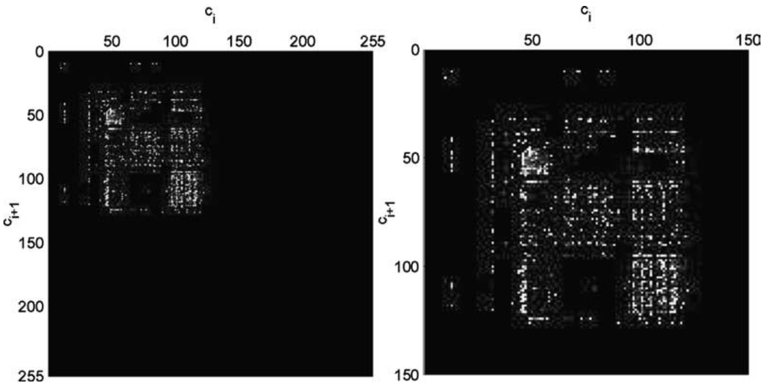

Fig. 5. Matrix of conditional probabilities for HTTP/GET requests. Horizontal and vertical axes represent the ASCII code for each character, while each pixel measures the probability $P(c_{i+1}|c_i)$ on a grey scale, from 0 (darker) to 1 (whiter). The right picture shows an enlargement of the dense region in the left one.

propose using $P(c_i|c_{i-1})$, that is, the probability of a character given the previous one (conditional probability), as a natural extension to the simple probability model. Similarly to the fact that not all characters have a similar probability of occurrence within a payload, it is hoped that, given a fixed character, the probability of the next one will be different among the possible cases.

Fig. 5 shows the visual aspect of the matrix of conditional probabilities $[P(c_j|c_i)]$ obtained for payloads destined to host `hume` (results for `marx` are similar). It is evident that this matrix is not random at all. Just as the histogram reveals that certain regions present a greater probability of occurrence than others, the conditional distribution shows that this fact can be seen with a more accurate perspective when conditional probabilities are considered. The use of these probabilities could thus enhance the process of measuring normality for the purposes of anomaly detection, and therefore provide an additional feature.

The use of conditional probabilities as a feature for the evaluation of the degree of normality of a given payload can be achieved by using a Markov chain as follows (Appendix A). As in Section 3.2, an alphabet $\Sigma$ corresponding to the ASCII code is considered. Assuming this, during a training stage a set of normal payloads is analysed in order to determine two probability terms: (a) the probability that a given character from the alphabet will follow any other in the payload, $P(c_j|c_i)$, $\forall c_i, c_j \in \Sigma$; and (b) the probability of a given character being the first in the payload. These probability terms correspond to the transition matrix, **A**, and the initial probability vector, $\boldsymbol{\Pi}$, of a Markov chain, respectively.

Once the Markov model is constructed, it is possible to estimate the probability of a given sequence of symbols (payload) as generated by the model according to expression (A.10). The detection procedure is motivated as follows: while incoming symbols correspond to those expected by the model, the respective probabilities of transition between states are adequate and, therefore, the accumulated sum given by (A.10) has no abrupt changes of slope. On the contrary, the occurrence of any pattern of non-expected symbols produces a burst of consecutive low probabilities (an anomaly is, by definition, a low-probability event). This phenomenon is captured by an abrupt change in the slope of the function.

A useful method for detecting these changes and, hence, the presence of anomalies in the input, is to control when the derivative of the function 'LogMAP' exceeds a fixed threshold $\tau$. To achieve this purpose, we used the family of functions:

$$D_{W_m(t)} = \left| \mathrm{LogMAP}(t) - \frac{1}{W_m} \sum_{i=1}^{W_m} \mathrm{LogMAP}(t-i) \right| \tag{2}$$

for values of the parameter $W_m = 1, 2, 3, \ldots$ Note that the second term in (2) is the mean of the last

$W_m$ outputs. This function, which is an approximation of a discrete derivative, provides an output after each symbol analysed in the sequence. The values obtained can be compared with a fixed detection threshold $\tau$, so that the payload is labelled as normal if the resultant value does not exceed this threshold. Otherwise, an alarm is triggered. Thus, it is possible to obtain performance measures related to the discriminative power of the model between normal and anomalous HTTP payloads.

This approach contains two major tuning parameters: $\epsilon$ and $\tau$. As stated in Appendix A, the value $\epsilon$ is assigned to transition probabilities which would otherwise be zero or less than $\epsilon$ in the trained model. As in other contexts where probabilistic models are used, the global effect derived from the use of such a value is a smoothing of the model. The value of $\epsilon$ represents a permissiveness factor, such that lower values produce a system that is more sensitive to deviations with respect to the profile (i.e., anomalies), and vice versa. The threshold $\tau$ used to decide whether a sequence is normal plays an analogous role. While lower values of $\tau$ lead to a detector that is more liable to be affected by false alarms, an overestimated threshold may be unable to detect attacks with a low level of abnormality.

This trade-off can be observed in Fig. 6, which shows the ROC curve obtained after evaluating the proposed scheme with normal traffic from the host `marx` and the attacks used in this study (results for `hume` are very similar to these). The curve corresponds to a value of the parameter $\epsilon = 10^{-15}$, and is obtained by varying the detection threshold $\tau$. Note that lower values of $\tau$ imply a higher number of correct detections, but also an increased number of false alarms. The ROC curve graphically summarizes this behaviour.

The best performance is achieved with the above-mentioned value for $\epsilon$; greater values invariably produce worse results. The results obtained are not very accurate, since detection rates higher than 90% are achieved with a false alarm rate of close to 30% for conditional probabilities. Nevertheless, the most important conclusion is the improved performance obtained by this approach in comparison with those discussed in Section 3.

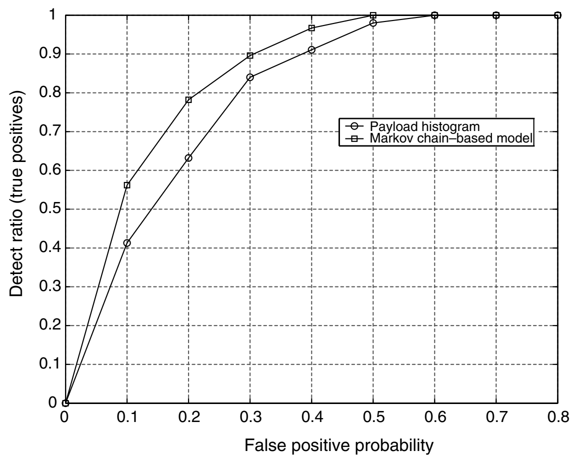

Fig. 6. ROC curve corresponding to the Markov chain-based modelling for $\epsilon = 10^{-15}$, trained with data sets from both hosts. Results obtained with payload histograms are also shown for comparison.

## 4.2. *Inclusion of protocol-dependent information*

Given the role that network traffic plays as a common avenue for misuse and security incidents, the analysis of network packets is a major concern in the development of alarm systems. *Protocol analysis* is a term that groups a broad spectrum of techniques to detect misuse and anomalies in network traffic, through the supervision of the behaviour of protocol instances. Some of the approaches proposed make use of the formal syntax and/or semantics of the protocol in order to perform the detection, while other techniques are solely based on a model of normal behaviour derived from the application of diverse machine-learning algorithms to captured traffic (see, for example, [8,16–18]).

In this context, we propose an anomaly-based scheme to detect attacks against the HTTP service that combines the two paradigms, learning and specification. In essence, our approach follows the above-discussed basic idea of modelling the payload as a Markovian process. The results obtained in Section 4.1 show that the occurrence of characters within the payload is not random, and so it can be modelled to obtain statistical measures of normality. Nevertheless, it is possible to use the protocol syntax to improve the modelling technique proposed previously. The HTTP specification defines a common structure for every payload, which is composed of several sections each containing different information units. Since each section has its own set of allowed values according to its purpose and semantics, it is natural to suppose that the probability of occurrence of certain strings within each section of the payload is not uniform throughout the request.

With this rationale in mind, an architectural view of the system is depicted in Fig. 7. During the training stage, each HTTP payload is initially segmented into blocks of characters according to a number of delimitation elements provided by the corresponding protocol specification. The sequence of blocks obtained is then used as a learning instance for a training algorithm, which estimates the various parameters that define the model (a Markov chain in this case). After the training period, a model is available for the evaluation of incoming HTTP traffic. The payloads are analysed separately, with an initial segmentation stage identical to that used for the training. The use of a special-purpose algorithm allows us to measure the degree of similarity between the incoming payload and the assumed model of normality. Details concerning this approach, together with experimental results, are provided below.

### 4.3. Segmentation and training

As in the case of lower-layer protocols, one of the main features of application traffic is the fact that the information is highly structured according to the syntax rules provided by the corresponding protocol. In the case of HTTP, the most important information related to resource access is contained within the so-called *request line*. To be precise, the name of the resource addressed and the CGI-like queries on scripts and other processes in the local server are both located within the URL (see the appropriate RFCs in [15,19] for details).

The structured nature of HTTP payloads makes it feasible to perform a precise segmentation of the content into blocks delimited by well-known reserved symbols (characters), like carriage return (CR) and line feed (LF) for distinct parts of the payload. In addition to CR and LF, a URL can be segmented according to the following seven characters: '/', '.', '?', '&', '=', '#', and blank space. For instance, a complete URL containing a query destined to a CGI script usually takes the following form:

```
host.name.domain/directory/
script.name?var1=my+value&
var2=value2
```

The sequences obtained after the segmentation correspond to server and resource names,

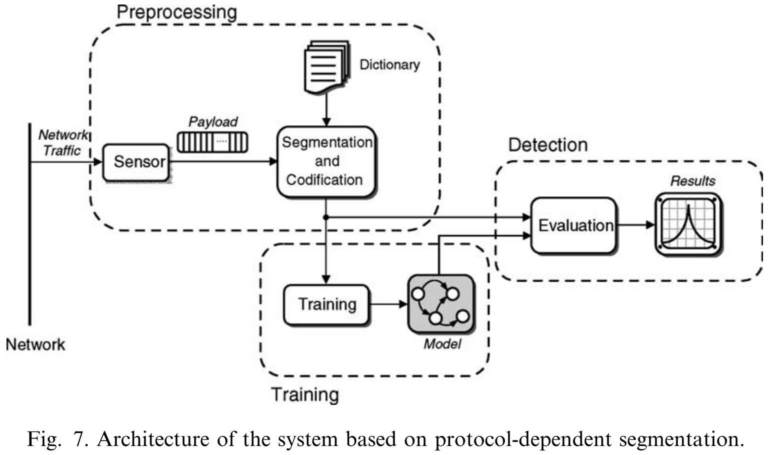

Fig. 7. Architecture of the system based on protocol-dependent segmentation.

directories, variables and values destined to CGI scripts, etc.

The first stage of the training is the construction of a dictionary of blocks. This structure is a list with an entry for each distinct sequence of characters delimited between the previously cited reserved symbols. Distinct sequences are obtained from the training data set by segmenting each payload and retrieving the segments obtained. After this stage, a unique symbol is associated with each entry in the dictionary.

Once the dictionary has been constructed, each segmented payload is transformed into a sequence of symbols. To do this, each segment is replaced by its corresponding scalar entry in the dictionary. The array of symbols obtained is used to train a Markov chain, each payload constituting a training instance. Thus, the model captures both the occurrence probability of each sequence and its spatial appearance within the payload. The results obtained from this process concern the matrix of transition probabilities and the vector of initial probabilities, as well as the dictionary itself.

In our experiments, the dictionaries obtained have a size of 356 distinct blocks for traffic destined to host `hume` and 408 in the case of `marx`.

## 4.4. Evaluating the scheme

After the training phase, a Markov chain-based model is available for the evaluation of incoming HTTP traffic destined to each specific host. The assessment process is as follows: each payload is segmented according to the delimiters described in the previous section, thus obtaining a sequence of blocks. Each block is then transformed into its respective scalar index in the dictionary. If the current observed block is not included in the dictionary (known as the OOV—*out of vocabulary*—problem), a special symbol $S_0$ is associated with it. This symbol is reserved for blocks that have not been previously seen, and is associated with a fixed probability of transition, $P(S_0|S_i) = P(S_j|S_0) = \epsilon$, $\forall i, j$, within the model. The sequence of symbols obtained is then passed through the model and evaluated according to the methodology discussed in Section 4.1.

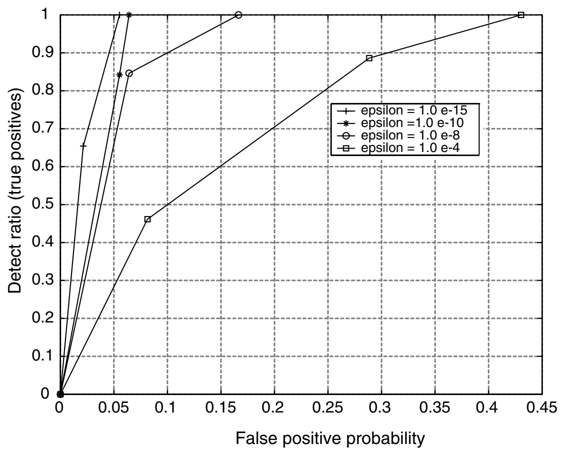

Fig. 8. ROC curves corresponding to the new technique introduced for different values of the parameter $\epsilon$.

Fig. 8 shows the ROC curves obtained after evaluating the proposed scheme with normal traffic from the host `marx` and the attacks used in this work (the results for `hume` are very similar to these). Each curve in the figure corresponds to a different value of the parameter $\epsilon$. The best results are obtained for lower values of the smoothing parameter. For example, with $\epsilon = 10^{-15}$ the full detection rate is achieved with only 5.76% of false positives. Nevertheless, values of $\epsilon$ lower than $10^{-15}$ do not produce better performance. A comparison between these results and those provided by the previous methods reveals the substantial improvement achieved by this model. As summarized in Table 3, in the case of the model based on conditional probabilities, detection rates higher than 95% are achieved with a false positive rate close to 40%. By using the proposed tech-

Table 3
Summary of experimental results obtained throughout the evaluation

| Modelling technique | %FA (detect ratio = 95%) |
|---|---|
| Payload length | 57.36 |
| Payload histogram | 44.17 |
| Markov chain (conditional probabilities, $\epsilon = 10^{-15}$) | 39.81 |
| Markov chain (syntactical segmentation, $\epsilon = 10^{-10}$) | 5.61 |
| Markov chain (syntactical segmentation, $\epsilon = 10^{-15}$) | 4.98 |

nique, this rate can be reduced by up to 5%, which implies a reduction of one order of magnitude.

## 5. Application in real environments

The scheme and results discussed in the previous section provide the basis for the construction of anomaly-based security monitors designed to inspect the incoming traffic destined to a web server in search of attacks carried over HTTP. Although the experimental results have demonstrated the benefits of the proposed method, the deployment of an operative detection system in a real environment still poses some challenges. These and other related questions are discussed below; we find that the proposed system presents the same general limitations as do other currently used IDS.

### *5.1. System operation*

Mechanisms based on anomaly detection for the purpose of intrusion detection should not be viewed as a general solution to the problem of identifying attacks in a network infrastructure. On the contrary, their use in conjunction with misuse-based techniques is strongly recommended.

In a real environment, operation in conjunction with a misuse-based system can follow a logical architecture like that depicted in Fig. 9. Note that in the case of application-layer intrusion detection, both modules (misuse and anomaly-based devices) share a common infrastructure for the capture of network traffic and the subsequent packet reassembling. In most existing network-based IDSs (e.g., SNORT), this stage is already included in the system, since it is necessary for the search for attack signatures in the payload. In summary, anomaly-detection can be conceived as an additional analysis performed over the application-layer payload, together with detection algorithms based on pattern matching, to obtain evidence about the likelihood of an ongoing attack.

Whereas a signature-based detection device needs constant updating of its database of signatures in order to detect new attacks, an anomaly-based system requires periodic retraining. It is usually assumed that all such systems to be modelled evolve over time, and therefore that the behaviour model obtained should be adapted periodically.

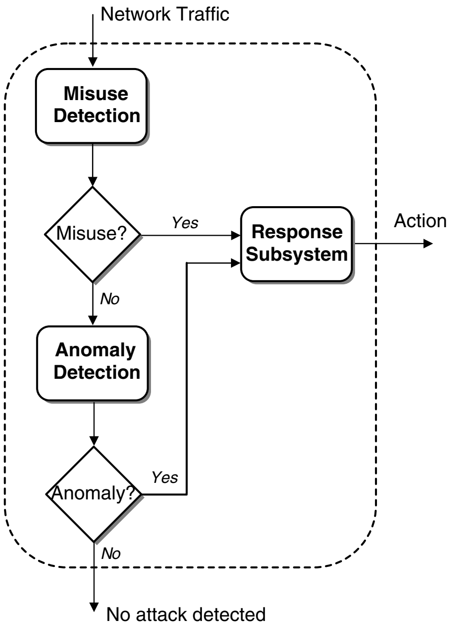

Fig. 9. Joint operation of misuse and anomaly-based detectors.

It is important to recall that every model intended for anomaly-detection has to be constructed using "normal" payloads obtained during a representative period of time. The reliance of our model on the data sets is not very different from that presented by other anomaly-based techniques. In the present case, it is necessary to periodically adapt the model by means of new normal payloads. This retraining step is particularly crucial when the HTTP server contents are changed, since they are closely related to the model.

Finally, the problem of using ciphering techniques in network traffic must be mentioned. It is obvious that the ciphering of payloads enters into conflict with the capture and evaluation of network traffic. This problem underlies all intrusion–detection mechanisms which require plain access to certain packet fields that are carried throughout the network and protected by cryptographic codes. All signature-based IDSs are also subject to this constraint. The only current solution to this severe drawback is that of moving the analysis location into the protocol stack of the destination system, specifically above the layer responsible for deciphering the content. Note that this approach does not mean a change in the network-based nature of the detection system; even though the detector is indeed located in the destination host, its main source of data is network traffic.

### 5.2. *Efficiency and scalability*

As stated in Appendix A, the construction of a Markov chain requires two basic elements: the transition matrix $\mathbf{A}$ and the initial distribution vector $\boldsymbol{\Pi}$. In addition to this, the scheme presented in this study involves a preprocessing stage in which each payload is segmented and codified as a sequence of symbols. A brief analysis of the efficiency of this approach, measured in time complexity, is provided below.

The construction of the dictionary of symbols is a linear process in which each payload is segmented and each segment is included in the list. This stage has to be carried out only once, since subsequent updates of the dictionary can be performed by adding new elements to the existing data. Although the dictionary is a simple list of segments, its elements can be rearranged into a more suitable structure with the aim of facilitating the search process. For example, a hash table can be used for this purpose, which allows us to obtain the corresponding symbol of a given segment in $O(1)$ operations. Thus, the segmentation and codification stage can be performed in $O(n)$ operations.

Concerning the model construction, the estimation of the transition matrix $\mathbf{A}$ according to expression (A.8) requires $O(n^2)$ operations. Estimation of the initial distribution vector ($\boldsymbol{\Pi}$) is not relevant from the point of view of its complexity since it can be carried out at the same time as that of the transition matrix.

According to the decision criterion used in this study, the evaluation of an input payload is an algorithmic process that requires $O(n)$ operations. Provided a Markov chain $\lambda$ and an input sequence of symbols $S$, obtaining $P(S|\lambda)$ (i.e., the probability that the sequence has been generated by the model) is achieved by accumulating the individual probabilities of transition between states. After this, another pass is required over the output sequence of probabilities in order to detect anomalous events.

Table 4
Efficiency of the various stages involved in the system operation

| Operation mode | Task | Complexity (time) |
|---|---|---|
| Training | Construction of the dictionary | $O(n)$ |
| Training, test | Segmentation and codification of a payload | $O(n)$ |
| Training | Construction of the model $\tau = (\mathbf{A}, \boldsymbol{\Pi})$ | $O(n^2)$ |
| Test | Evaluation of a codified sequence | $O(n)$ |

Table 4 summarizes the analysis. Current misuse-based IDSs, which are mainly based on pattern-matching algorithms, show complexities similar to those described here. This comparison supports the hypothesis that the proposed approach can be successfully deployed in environments that are currently running an IDS.

In our context, *scalability* refers to the efficient handling of a high volume of input traffic. In a high-speed environment, like gigabit links, current IDS technology suffers limitations due to the amount of resources required to store and process such a volume of traffic in real time. In addition to the benefits that can be obtained from an improvement in the algorithms involved in the detection process, performing a distributed analysis seems to be a good choice for the application in an environment with these constraints. The simplest way to carry out this task is by means of a replication of the processing units that perform the detection, coupled with a scheduler module that captures payloads and distributes them to idle units in the cluster. This arrangement, which has been used in many applications, allows us to simultaneously process as many payloads as processing units exist in the cluster.

### 5.3. *A note on content inspection and legal issues*

Processing application-level payloads unavoidably involves the capture and inspection of the information carried over them. During a session with a certain server, the contents interchanged can sometimes include user information whose inspection, even for monitoring tasks, could involve certain legal issues. Given the current requirements concerning privacy and related matters, this task should be carried out carefully.

In the modelling technique proposed in this work, information carried over the HTTP-like payload is segmented according to the proposed scheme and then injected into the model. Furthermore, the codification stage involves a substitution of each segment by a symbol from within the dictionary. This operation mode implies that, although compromising or private information may be carried over the payload, its semantics is absolutely broken up during the training stage (note that the dictionary involved in our approach is a plain structure that maps blocks of characters defining a complete request into a finite set of symbols).

At first, information relative to the data source (e.g., IP address or even user identity) is not taken into consideration during the decision-making process. It is true, however, that the dictionary can include blocks corresponding to private information. Although in its current form it is absolutely impossible to map blocks with their original users or systems, the model itself should be carefully protected against potential misuses.

Nonetheless, it is undeniable that this problem should be considered in current and future security devices aimed at supervising activities that are directly or indirectly generated by users. The application of techniques similar to that described in this study in other application-layer protocols, such as mail, requires analogous attention. In this context, the construction of user profiles for the purpose of host-based intrusion detection would arouse similar concerns. In the same way that many institutions and companies are obliged to protect private and user-sensitive data, those monitoring systems that record and process this information should also be subject to similar regulations.

## 6. Conclusions and future work

This paper presents our work developed in the field of application-specific anomaly detection for intrusion–detection purposes. The experiments carried out reveal that there exist some features which could discriminate between normal and anomalous HTTP requests. These results provide the possibility of developing anomaly-based detectors which, in conjunction with signature-based systems, could enhance the detection capabilities of current defences in network environments.

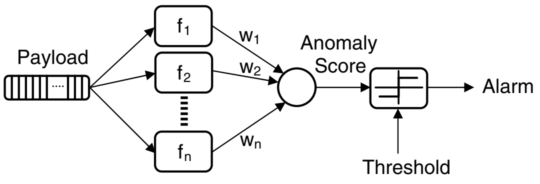

Fig. 10. Multifeature anomaly detection.

In addition to the above, a preliminary analysis was carried out to measure normality, taking into consideration several features simultaneously. Once the related features are obtained, the evaluation methodology is as follows (Fig. 10 depicts the process):

1. Label each payload with $n$ features: $f_p = < f_1, f_2, \dots, f_n >$. The set of normal payloads is thus transformed into a cloud of points, $\mathcal{G}$, in an $n$-dimensional space.
2. Obtain the mean vector $\mu_{\mathcal{G}}$ over the previous cloud of points, as well as the covariance matrix $\Sigma_{\mathcal{G}}$.
3. Compute the distance $\mathcal{D}$ from each payload $f_p$ to the mean $\mu_{\mathcal{G}}$.
4. Obtain the distance distribution (pdf) of $\mathcal{D}(f_p, \mu_{\mathcal{G}})$.

It would also be interesting to explore functions which take into consideration all the above features to obtain an *anomaly score* $A_s$. For example, a weighted sum like the following:

$$A_s = \sum_{i=1}^{n} w_i f_i, \tag{3}$$

where $f_i$ are the mentioned relevant features and $w_i$ are weights to establish the relative importance of each feature in the decision criterion. An important aspect of such an approach concerns how to tune the weights. They are generally fixed experimentally; for instance, a Principal Component Analysis (PCA) would help achieve this goal. In addition to the questions discussed in this paper, our current work is related to the extension of these projects to other application-layer protocols suitable for modelling within this framework. For instance, DNS and mail systems appear to be attractive research objectives and will be considered next. Nevertheless, anomaly detection is still a difficult task in many respects. Further knowledge, derived from continuing research, is required for this technology to achieve a firmly-based maturity and for it to become established in environments that are still evolving and where serious threats remain.

**Acknowledgements**

This study is partially supported by the Spanish Government through "Ministerio de Educación, Cultura y Deporte" (National Program PNFPU, reference AP2001-3805) and "Ministerio de Ciencia y Tecnología" (project TIC2002-02798, FEDER funds 70%). The authors would like to thank the anonymous reviewers for their helpful criticisms of earlier versions of this paper.

## Appendix A. Statistical background

Several statistical approaches have been proposed in the literature to model the behaviour of a system and to compare predicted and real behaviour. This Annex is a brief and formal overview of the statistical tools used in this work. First, the Kruskal–Wallis and the Kolmogorov–Smirnov tests are introduced in the context of the so-called "goodness-of-fit" proofs, used for evaluation in Section 3. Then, Section A.2 presents the fundamental aspects of Markov chains, the mathematical approach underlying the anomaly-based intrusion detection system described in Section 4 and which is the core of this work.

### *A.1. Goodness-of-fit statistical tests*

Two widely used "goodness-of-fit" tests, particularly in the field of intrusion detection, are the Kruskal–Wallis and the Kolmogorov–Smirnov tests.

#### *A.1.1. The Kruskal–Wallis test*

The Kruskal–Wallis test (KW) is a non-parametric (distribution-free) version of the one-way analysis of variance (ANOVA), and it is used to verify that multiple sample sets are identical against the alternative hypothesis that they come from different distributions [20]. Suppose that we have $K$ independent sample sets of sizes $n_1, n_2, \ldots, n_K$, respectively. Let us combine all the samples into one larger $N$-sized sample, then sort it in ascending order and assign ranks. If $\overline{R}_i$ is the average of the ranks of the observations in the $i$th sample, the test statistic $S_{\mathrm{KW}}$ is given by

$$S_{\mathrm{KW}} = \frac{12}{N(N+1)} \sum_{i=1}^{K} n_i \left( \overline{R}_i - \frac{N+1}{2} \right)^2. \tag{A.1}$$

The null hypothesis that all $K$ distributions are the same is rejected if $S_{\mathrm{KW}} > \chi^2_{K-1}$, where $\chi^2$ is the well-known chi-square distribution. Hence, the Kruskal-Wallis statistic approximates a chi-square distribution with $K-1$ degrees of freedom if the null hypothesis of equal populations is true.

#### *A.1.2. The Kolmogorov–Smirnov test*

The Kolmogorov–Smirnov (KS) is of the best known and most widely used goodness-of-fit tests because of its simplicity [20]. KS tests the null hypothesis that a given data sample $X$ conforms to a certain hypothesized distribution function and, like the Kruskal–Wallis test, KS is a non-parametric test, that is, no assumption about the distribution of data is made.

The KS test is based on the estimation of the maximum difference between the empirical distribution function and the hypothesized cumulative distribution. Let us suppose that $\mathscr{F}'$ is the empirical distribution function of the samples, and $\mathscr{F}$ the hypothesized one. The statistic $D_{\mathrm{KS}}$ is the largest absolute deviation between $\mathscr{F}(x)$ and $\mathscr{F}'(x)$ over the range of the variable:

$$D_{\mathrm{KS}} = \max_{x\in X} \{|\mathscr{F}(x) - \mathscr{F}'(x)|\}. \tag{A.2}$$

The null hypothesis that the sample $X$ conforms to the distribution function $\mathscr{F}$ is rejected or accepted for a given level of significance $\alpha$ by looking it up in a table of values and by comparison with the statistic $D_{\mathrm{KS}}$ obtained. There are several variations of these tables in the literature that use somewhat different scalings for the KS test statistic and critical regions. The KS tables are not specified here since the software programs that perform a KS test provide the relevant critical values.

### *A.2. Basic principles of Markov chains*

Some practical problems can be expressed in terms of a Markov process, which is a stochastic process in which the future distribution of a variable depends only on the variable's current value. Such processes can be modelled by a Markov chain, which is a model of event sequences in which the probability of an event occurring depends upon the fact that a preceding event occurred.

A Markov chain is defined by a set of $N$ states $\Gamma = \{S_1, S_2, \ldots, S_N\}$ and by the pair of probability matrices $(\boldsymbol{\Pi}, \mathbf{A})$. Each of the states represents a different and specific situation in which the system can be at a given time. The matrices allow us to express the temporal evolution of the system from a statistical point of view:

- $\boldsymbol{\Pi} = \{\pi_i\}\ \forall i \in [1, N]$: vector that indicates the probability of the $i$th state being the first of the temporal sequence of observations, that is,

  $$\pi_i = P(q_1 = S_i), \tag{A.3}$$

  where the variable $q_t$ represents the current state of the model at time $t$. The vector of initial probabilities $\boldsymbol{\Pi}$ has the following two constraints:

  $$\pi_i \geqslant 0\ \ \forall i \in [1, N]; \quad \sum_{i=1}^{N} \pi_i = 1. \tag{A.4}$$

- $\mathbf{A} = [a_{ij}]\ \forall i, j \in [1, N]$: given that the system is in the state $i$ at some time $t$, probability of, it reaching the state $j$ at time $t+1$, that is,

$$\begin{aligned} a_{ij} &= P(q_{t+1} = S_j | q_t = S_i) \\ &= \frac{P(q_t = S_i, q_{t+1} = S_j)}{P(q_t = S_i)}. \end{aligned} \tag{A.5}$$

Thus, the matrix $\mathbf{A}$ represents the transition probabilities between states, and has the following two constraints:

$$a_{ij} \geqslant 0\ \ \forall i, j \in [1, N]; \quad \sum_j a_{ij} = 1\ \ \forall i \in [1, N]. \tag{A.6}$$

According to the previous definitions, the probability $P_j^{(t)}$ of state $j$ at time $t$ is given recursively by

$$\begin{aligned} P_j^{(1)} &= \pi_j, \\ P_j^{(t)} &= \sum_i P_i^{t-1} a_{ij}, \quad t > 1. \end{aligned} \tag{A.7}$$

The analysis of a system by means of a Markov chain requires us to solve two main problems: (a) *training*, or how to estimate the probability matrices associated with the model, and (b) *evaluation*, or how a given sequence of observations should be recognized by using a previously estimated model.

#### *A.2.1. Parameter estimation*

In this discussion we suppose that the knowledge concerning different states reached by the system is acquired through the observation of the system outcomes. These outcomes are elements from a finite set $\Theta = \{O_i\}$, such that each of them is referred to as a possible state of the system. Let us suppose that a set of system observations $O_1, O_2, \ldots, O_T$ is given. In the theory of Markov chains we consider the simplest generalization, which consists of permitting the outcome of any trial to depend on the outcome of the directly preceding trial, and only on this outcome [21]. Thus the matrix of probabilities of transitions can be estimated by

$$a_{ij} = \frac{P(q_{t+1} = O_j, q_t = O_i)}{P(q_t = O_i)}. \tag{A.8}$$

The two probabilistic terms in the previous expression can be calculated by means of a simple process of counting occurrences in the sequence of observations.

The vector of initial probabilities $\boldsymbol{\Pi}$ can be estimated in a similar way if a set of outcome sequences is available. Thus, the initial probability of each symbol can be computed by simply counting the number of times the corresponding symbol appears at the beginning of the observed sequences.

### *A.2.2. Sequence recognition*

Let us suppose a given Markov chain $\lambda = (\mathbf{A}, \boldsymbol{\Pi})$, where $\mathbf{A} = [a_{ij}]$ is the matrix of transition probabilities and $\boldsymbol{\Pi} = \{\pi_i\}$ the vector of initial probabilities, and let $O = \{O_1, O_2, \ldots, O_T\}$ be a sequence of observed symbols. The problem of recognition with Markov chains is the problem of estimating $P[O|\lambda]$, that is, the probability that the observed sequence has been generated by the chain. A useful measure for this purpose is the so-called *Maximum A-posteriori Probability* (MAP), defined as

$$\mathrm{MAP}(O, \lambda) = \pi_{O_1} \prod_{t=1}^{T-1} a_{O_t O_{t+1}}. \tag{A.9}$$

A problem with this measure is that it quickly converges to zero. Therefore, it is sometimes more useful to use a representation on a logarithmic scale, that is

$$\mathrm{LogMAP}(O, \lambda) = \log(\pi_{O_1}) + \sum_{t=1}^{T-1} \log(a_{O_t O_{t+1}}). \tag{A.10}$$

The use of logarithmic probabilities presents the drawback that no probability can be zero; this is usually solved by means of a previous *smoothing* of the model. Although several methods exist to do this, because of its simplicity probably the most widely used smoothing technique consists of setting these probabilities lower than a given threshold, to a fixed value $\epsilon$.

Good introductory texts about Markov chains are [21,22], in which interested readers can find more detailed information.

## References

[1] D. Denning, An intrusion–detection model, IEEE Transactions on Software Engineering SE-13 (2) (1987) 222–232.

[2] M. Roesch, Snort—lightweight intrusion detection for networks, in: Proceedings of USENIX LISA 99, USENIX Association, Berkeley, 1999, pp. 229–238. Also available online at <http://www.snort.org>.

[3] V. Paxson, Bro: a system for detecting network intruders in real-time, in: Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, USENIX Association, Berkeley, 1998, pp. 31–51.

[4] G. Vigna, R.A. Kemmerer, NetSTAT: a network-based intrusion detection system, Journal of Computer Security 7 (1) (1999) 37–71.

[5] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, E. Stoner, State of the practice of intrusion detection technologies, Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, Carnegie Mellon, January 2000.

[6] S. Axelsson, Intrusion detection systems: a taxonomy and survey, Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology, Göteborg, 1999.

[7] J.B.D. Cabrera, B. Ravichandran, R.K. Mehra, Statistical traffic modeling for network intrusion detection, in: Proceedings of the 8th IEEE International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems, 2000, pp. 466–473.

[8] C. Krügel, T. Toth, E. Kirda, Service specific anomaly detection for network intrusion detection, in: Proceedings of the 17th ACM Symposium on Applied Computing (SAC), Madrid, Spain, 2002, pp. 201–208.

[9] N. Athanasiades, R. Abler, J. Levine, H. Owen, G. Riley, Intrusion detection testing and benchmarking methodologies, in: Proceedings of the 1st IEEE International Workshop on Information Assurance, Darmstadt, Germany, March 2003, pp. 63–72.

[10] F. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in: Proceedings of the 15th International Conference on Machine Learning (ICML-98), Morgan Kaufmann, San Mateo, CA, 1998.

[11] R. Lippmann, J.W. Haines, D.J. Fried, J. Corba, K. Das, The 1999 DARPA off-line intrusion detection evaluation, Computer Networks 34 (4) (2000) 579–595.

[12] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory, ACM Transactions on Information and System Security 3 (4) (2000) 262–294.

[13] arachNIDS: advanced reference archive of current heuristics for Network Intrusion Detection Systems, 2003. Available from: <http://www.whitehats.com/ids>.

[14] A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, I. Cohen, Self-aware services: using Bayesian networks for detecting anomalies in Internet-

based services, HP Labs Technical Reports HPL-2001-23R1, 2001.

[15] R. Fielding et al., Hypertext Transfer Protocol—HTTP/1.1, RFC 2068, 1998.

[16] M.V. Mahoney, P.K. Chan, Learning models of network traffic for detecting novel attacks, Florida Institute of Technology Technical Reports CS-2002-08, 2002.

[17] M.V. Mahoney, Network traffic anomaly detection based on packet bytes, in: Proceedings of the 18th ACM Symposium on Applied Computing (SAC), Melbourne, FL, USA, 2003, pp. 346–350.

[18] J.M. Estévez-Tapiador, P. García-Teodoro, J.E. Díaz-Verdejo, Stochastic protocol modeling for anomaly-based network intrusion detection, in: Proceedings of the 1st IEEE International Workshop on Information Assurance (IWIA), Darmstadt, Germany, 2003, pp. 3–12.

[19] T. Berners-Lee et al., Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396, 1998.

[20] R. D'Agostino, M. Stephens, Goodness-of-Fit Techniques, Marcel Dekker, New York, 1986.

[21] J.L. Doob, Stochastic Processes, Wiley, New York, 1953.

[22] W. Feller, An Introduction to Probability Theory and its Applications, vol. I, third ed., Wiley, New York, 1968.

**Juan M. Estévez-Tapiador** is a Ph.D. candidate in the Department of Electronics and Computer Technology of the University of Granada, Spain. He received an MS in Computer Sciences and Engineering from the University of Granada, where he obtained the "Best Student" Academic Award. Currently he is holding a Ph.D. grant from the Spanish Ministry of Education. Since 2000, he is a member of the "Research Group on Signals, Telematics and Communications" of the University of Granada, where he has been involved in several public and private research projects. His research is focused on computer and network security, especially in intrusion detection and response systems, new security paradigms and group communications security.

**Pedro García-Teodoro** received his B.Sc. in Physics (Electronics speciality) from the University of Granada, Spain, in 1989. This same year he was granted by "Fujitsu España", and during 1990 by "IBM España". Since 1989 he is Associate Professor in the Department of Electronics and Computer Technology of the University of Granada, and member of the "Research Group on Signal, Telematics and Communications" of this University. His initial research interest was concerned with speech technologies, especially automatic recognition, field in which he developed his Ph.D. Thesis in 1996. From then, his profile has derived to that of computer networks, and although he has done some works in telematics applications and e-learning systems, his main current research line is centred in computer and network security.

**Jesús E. Díaz-Verdejo** is Associate Professor in the Department of Electronics and Computer Technology of the University of Granada (Spain). He received his B.Sc. in Physics (Electronics speciality) from the University of Granada in 1989 and has held a Ph.D. grant from Spanish Government. Since 1990 is member of the "Research Group on Signal, Telematics and Communications" of the University of Granada. This year he became Assistant Professor at this University. In 1995 he obtained a Ph.D. degree in Physics. His initial research interest was related with speech technologies, especially automatic speech recognition. Currently he is working in computer networks, mainly in computer and network security, although he has developed some work in telematics applications and e-learning systems.