A modular architecture for the analysis of HTTP payloads based on Multiple Classifiers

Davide Ariu[1] and Giorgio Giacinto[1]

Department of Electrical and Electronic Engineering, University of Cagliari, Italy
`[davide.ariu,giacinto]@diee.unica.it`,
WWW home page: `http://prag.diee.unica.it`

Abstract. In this paper we propose an Intrusion Detection System (IDS) for the detection of attacks against a web server. The system analyzes the requests received by a web server, and is based on a two-stages classification algorithm that heavily relies on the MCS paradigm. In the first stage the structure of the HTTP requests is modeled using several ensembles of Hidden Markov Models. Then, the outputs of these ensembles are combined using a one-class classification algorithm. We evaluated the system on several datasets of real traffic and real attacks. Experimental results, and comparisons with state-of.the.art detection systems show the effectiveness of the proposed approach.

Keywords: Anomaly Detection, IDS, HMM, Payload Analysis

1 Introduction

The always increasing number of Web-based applications that are deployed worldwide, makes their protection a key topic in computer security. The traditional defense systems (e.g. Intrusion Detection/Prevention Systems) are based on a database of signatures that describe known attacks. Unfortunately, the large number of new attacks that appears everyday, and the wide use of custom applications on web servers, make almost impossible to have signature-based systems always updated to the most recent and effective attacks. A possible solution to this problem is offered by the "anomaly based" approach to intrusion detection.

An anomaly based system builds a model of the "normal" behavior of the resource to be protected. An attack pattern is detected if it appears "anomalous" with respect to the normal behavior, that is if it significantly deviates from the statistical model of the normal activity. The normal behavior is defined as a set of characteristics that are observed during normal operation of the resource to be protected, e.g., the distribution of the characters in a string parameter, the mean and standard deviation of the values of integer parameters [[6], [7]]. One of

This research was sponsored by the RAS (Autonomous Region of Sardinia) through a grant financed with the "Sardinia PO FSE 2007-2013" funds and provided according to the L.R. 7/2007. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the RAS.

Fig. 1. An example of legitimate HTTP payload

```
GET /pra/ita/home.php HTTP/1.1       ⟹       Request-Line

Host: prag.diee.unica.it
Accept: text/*, text/html            ⟹       Request-Headers
User-Agent: Mozilla/4.0
```

the reasons that initially prevented anomaly-based IDS from becoming popular is the fact that they tend to generate too high rates of false alarms. In fact the false alarm rate is a crucial parameter in the evaluation of an IDS since an IDS is generally required to manage large amounts of patterns (hundreds of thousands) every day. A strategy which is usually employed to mitigate this problem is that of realizing IDSs based on multiple classifiers in order to increase the overall classification accuracy [[2],[6],[7],[9],[11]].

The anomaly based IDS recently proposed in the literature for the protection of web servers and web applications basically analyze the requests received by the web server. HTTP requests are carried in the data portion of the network packet that is generally called "HTTP payload". An example of HTTP payload is presented in Figure 1. The HTTP protocol is defined by RFC 2616 [1]. According to this RFC, a HTTP payload contains a Request-Line plus a certain number of Request-Header fields. More in detail:

– The Request-Line begins with a method token (e.g. `POST`, `GET`), followed by the Request-URI and the protocol version, and ending with CRLF. The Request-URI contains the name of the resource requested on the web server. In Figure 1 the resource requested is the page `/pra/ita/home.php`.
– The Request-Header fields are used by the client to provide additional informations to the web server. For example, with the `User-Agent` header, the client host notifies to the web server the type and the version of the web browser. This information can be used by the web server to optimize the response sent back to the client according to the version of the browser. In Figure 1 the value of the `User-Agent` header is `Mozilla/4.0`.

Anomaly based IDS use statistical models to represent and analyze HTTP requests. Basically they create a statistical model of the bytes' distribution within the payload. Some of them, such as *HMM-Web* [5] or *Spectrogram* [11], focus on the Request-Line only and perform an analysis based on Hidden Markov Models (HMM), and on Mixture of Markov-chains respectively. Other approaches, such as *PAYL* [13] and *HMMPayl* [2] analyze the bytes' distribution of the whole payload using $n - gram - analysis$ or HMM. These IDSs are based on the assumption that the bytes' statistics of HTTP payloads containing attacks are different from the bytes' statistics of the legitimate traffic. Nevertheless, at the best of our knowledge, none of the IDSs proposed in the literature, exploits the a-priori knowledge of the structure of the HTTP payload.

In this paper, we propose an IDS based on HMM that effectively exploit the analysis of the different portions of the HTTP payload structure. In particular, for each header of the payload, we use a different ensemble of HMM to analyze the related values. Another ensemble of HMM is used to analyze the Request-Line. Attack detection is performed by stacking the outputs of the HMM ensembles, and using this vector as input for a one-class classifier. The experimental results achieved on several datasets of legitimate traffic and attacks confirm the effectiveness of the proposed approach, and its superiority with respect to similar IDS proposed in the literature.

The rest of the paper is organized as follows. In section 2 a review of the State of Art is provided. In Section 3 a detailed description of the IDS is provided. In sections 4 and 5 we describe respectively the experimental setup and results. We then draw the conclusions in section 6.

2 State of the Art

In the recent years, several anomaly based IDS have been proposed for the protection of web servers and web applications. They usually rely on multiple classifiers or models for two main reasons. First, multiple classifiers generally lead to better classification accuracy. In the case of IDSs, this means an higher percentage of detected attacks and a smaller percentage of false alarms. Examples of applications of Multiple Classifiers are [[2],[6],[7],[9],[11]]. The second reason for using multiple classifiers is that they usually increase the robustness of the system against attempts of evasion. This topic is gaining an increasing attention in the last years not only in the Intrusion Detection area but also in related fields such as spam detection and biometric authentication [4, 8].

Intrusion detection techniques such as those proposed in [[5],[7],[11]] limit their analysis to the structure of the Request-Line, and in particular they focus on the value of the input parameters received by the web applications. These approaches are tailored for the detection of the most frequent attacks against web applications, e.g., SQL-Injection, Cross-Site Scripting, that basically exploit the flaws of the web applications in the validation of the received input. Nevertheless, these IDS are completely ineffective against attacks that exploit other vulnerabilities of web applications.

IDS such as [[2],[9],[13]] cover a broader range of attacks since they model the bytes' distribution of the whole payload. As a consequence, they are theoretically able to detect any kind of attack that makes the payload statistics deviating from those of the legitimate traffic. The bytes' distribution of the payload can be modeled in several ways. *PAYL* [13] performs an $n - gram$ analysis using a very small value for n, since the size of the features space exponentially increases with n. This represent a severe limitation for *PAYL* since the IDS can be easily evaded if the attacker is able to mimic the statistics of the legitimate traffic. *HMMPayl* performs an analysis of the payload based on HMM. This analysis is equivalent to the $n - gram$ analysis, but it is able to circumvent the limitation on the value of n from which *PAYL* suffered. This lead to an increased classification accuracy of *HMMPayl* with respect to *PAYL*. Nevertheless, the analysis performed by *HMMPayl* is quite complex. This might be an issue since an IDS such as *HMMPayl* must be able to keep up with the network speed. For this reason, in this paper we propose to exploit the a-priory knowledge of the payload structure in order to significantly reduce the complexity of the classification algorithm without affecting the classification accuracy.

The largest part of the IDSs proposed in the literature are based on outliers detection techniques, and one-class classifiers. This means that the classifiers, or, more in general, the statistical models on which they are based, are built using samples of legitimate patterns only. There are various reasons for this choice. First of all, the main aim of anomaly based IDS is to recognize those patterns that are anomalous with respect to those assumed to be legitimate. In addition, a two-class model (normal vs. attack) would not be probably the one that best fits the problem. In fact the attack class would contain patterns that are completely different each other, as they exploit vulnerabilities of different type and, as a consequence, they exhibit statistical properties that are completely different [2]. Another (and more "practical") reason for using one-class classifiers is that collecting representative samples of the attack class is usually quite difficult. In fact, the attacks a web server might be susceptible to, depend on several elements such as the platform (e.g. the operating system), the hosted applications, the network topology, and so on. One could certainly scan the web server and analyze the web applications looking for possible vulnerabilities, and then create samples of attacks that exploit them. Nevertheless, the assessment of all the possible attacks a web server might be subject to, remains a task quite difficult and time consuming. If such a knowledge should be available, then there would be better ways to protect against known attacks than training a classifier, that is patching the vulnerabilities. Another possibility is to extract the signatures for that attacks, and deploy them in a signature-based IDS.

3 A modular architecture for the analysis of HTTP payloads

This section provides the details of the IDS proposed in this paper. A simplified scheme of the system is presented in Figure 2. Several solutions proposed in the literature (e.g. *HMM-Web*[5], *Spectrogram* [11]) focus their analysis only on the Request-Line (which is in red in the figure). Otherwise, solutions such as *PAYL* [13], or *HMMPayl* [2] analyze the bytes' distribution of the whole payload but do not take into account its structure. This paper aim to investigate the use of a model of the HTTP payload which reflects its structure as it has been defined by the RFC 2616 [1]. This analysis is performed in two steps. First, the payload is split in several "fields", that are analyzed by several HMM ensembles. Second, the outputs of the HMM are combined using a one-class classifier that finally assigns a class label to the payload. A more detailed description of these steps follows.

HMM Ensembles. We briefly reminded the HTTP payload structure in section 1, recalling that this structure consists of a Request-Line plus (eventually)

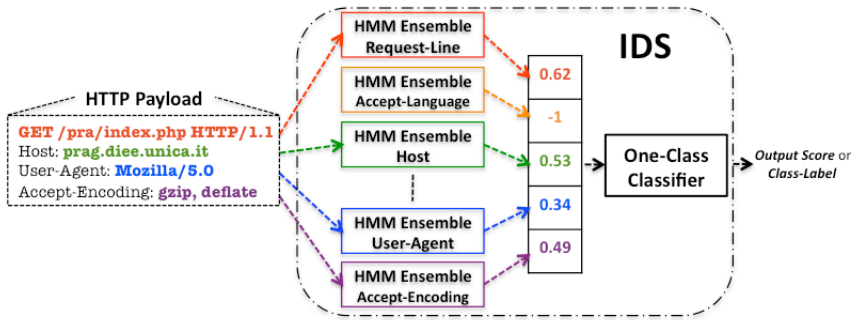
IDS
HTTP Payload
GET /pra/index.php HTTP/1.1
Host: prag.diee.unica.it
User-Agent: Mozilla/5.0
Accept-Encoding: gzip, deflate
HMM Ensemble Request-Line
HMM Ensemble Accept-Language
HMM Ensemble Host
HMM Ensemble User-Agent
HMM Ensemble Accept-Encoding
0.62
-1
0.53
0.34
0.49
One-Class Classifier
Output Score or Class-Label

Fig. 2. A simplified representation of the IDS architecture. An exhaustive list of all the Request-Headers analyzed by the IDS is reported in Table 1. The analysis of each header can be carried out by a single HMM or by an ensemble of HMM.

one or more Request-Headers. We also remind here that the Request-Line specifies the resource requested by the browser to the web server (e.g. the `index.php` page), whereas the Request-Headers are used by the web browser to provide additional information to the web server (thus their presence is optional). The RFC 2616 defines for the HTTP protocol a large number of possible Request-Headers, the most part of which is generally unused. In order to simplify the analysis, a list of the headers that a web server is actually using can be easily produced through a simple inspection of the incoming network traffic. Table 1 reports the list of the 18 headers that we observed within our datasets during the experimental evaluation of the proposed technique.

Once the set of the Request-Headers to be analyzed has been defined, the payload is processed as follows. First, a probability is assigned to each Request-Header and to the Request-Line by a different HMM ensemble through the *Forward-Backward* procedure [10]. With respect to the example shown in figure 2, the strings analyzed by the different HMM ensembles are the following: the string from "`GET`" to "`HTTP/1.1`" is analyzed by the the Request-Line ensemble (red); the string "`prag.diee.unica.it`" is analyzed by the `Host` ensemble (green); the string "`Mozilla/5.0`" is analyzed by the `User-Agent` ensemble (blue) and so on. Details about the setting of the HMM parameters will be provided in the following section. It is just worth noting that the use of HMM ensembles instead of single HMM allows mitigating the risk of having a single HMM that performs poorly, due to the random initialization of the parameters. The output of the HMM ensemble is thus computed by averaging the outputs of the individual HMMs, as they differ for the parameter initialization only.

One Class classifier. The analysis performed by the HMM ensembles produces as output a set of probabilities assigned to the Request-Line and Headers by the ensembles (see figure 2). Obviously a fusion stage is required in order to combine the outputs of the the different ensembles. We considered static rules (e.g. the mean or the product rule) to perform the combination. Unfortunately,

Table 1. List of the Request-Headers analyzed by the IDS. A detailed description of the role played by each header within the HTTP protocol can be found in [1].

`Accept`	`Connection`	`Cache-Control`	`Via`	`User-Agent`
`Accept-Charset`	`From`	`If-None-Match`	`UA-CPU`	`Transfer-Encoding`
`Accept-Encoding`	`Host`	`If-Modified-Since`	`UA-OS`	`X-Forwarded-For`
`Accept-Language`	`Referer`	`Keep-Alive`		

as we will show in section 5, they do not result suitable for this purpose. A further possibility is to concatenate the outputs of the ensembles within an array that will be provided as input to a one-class classifier. In this case, the outputs of the ensemble are used as features and a label (attack or legitimate) is assigned to the payload as the result of a classification in this features space.

It can be observed that a legitimate payload typically contains a number of five or six headers. Which headers are included in the payload depends on the settings of the HTTP client. On the other hand, the IDS must be able to analyze all the headers that occur in the network traffic (we observed the presence of 18 different headers in the network traces used in our experiments). As a consequence, the one class classifier will be designed to work in a features space of size equal to the number of observed headers (18 in our case) plus one, since the Request-Line must be also analyzed. From the perspective of the one class classifiers, the absent headers represent "missing features", since a probability will be associated by the ensembles only to the headers within the payload. The problem of managing these missing features is approached differently in the Training and in the Detection phases. During the training of the one class classifier, the missing features are replaced by their average value (computed over the payloads in which they are present). This is a practice well known in the literature [12]. This choice does not affect the results of classifier training since the most important features (e.g. the Request-Line) are present in the largest part of the traffic. In the detection phase, the missing components are set to the value of -1, that is a value outside the output range of the HMM (the output range is in [0,1], as they are probabilities).

Complexity Evaluation Since the training of the IDS is performed off-line, the complexity is estimated for the detection phase only. Let K the number of Request-Headers analyzed by the IDS (K typically assumes a value in the range between 15 and 20). If all of these headers appears in the payload, the IDS has to analyze $K+1$ sequences (the headers plus the request line). In addition a further classification step has to be performed in a features space of size $K+1$. Just to provide a brief comparison with a similar approach let us consider the *HMMPayl* algorithm [2]. In the case of *HMMPayl* the number of sequences analyzed by the IDS is approximately as high as the length (in terms of number of bytes) of the payload. A typical legitimate payload has a length of several hundreds of bytes. Thus, the solution proposed here offers two main advantages: the first is that the number of sequences analyzed is significantly smaller; the second is that this number is known and depends only on the setup of the IDS.

4 Experimental Setup

In this Section we describe the experimental setup on which we performed the experiments.

HMM Parameters The parameter that influences the most the performance of a discrete HMM is the number of (hidden) states. A rule does not exist to estimate the optimum value for the number of states for a bunch of data. Here, we used the "effective-length" of the training sequences, which is an heuristic that has been successfully used also in [5]. The effective length basically counts the number of different characters in a string. For instance the effective length of the string "abc" is 3, that of "abcd" is 4, and that of "aabcdd" is still 4. Thus, for each ensemble of HMM, we set the number of states of every single HMM equal to the average effective-length calculated on the corresponding training set. In addition, the transition and emission matrices are randomly initialized for each HMM. Then, the estimate of the model parameters that maximize the probability assigned by the model to the sequences within the training set is calculated by resorting to the *Baum-Welch* algorithm [3].

Datasets The intrusion detection algorithm proposed has been deeply tested on two different datasets of normal traffic, and on three datasets containing different kinds of attacks. For what concerns datasets of normal traffic both of them consists of real traffic traces collected at academic institutions. One dataset is made up of HTTP requests towards the website of the College of Computing at the Georgia Tech (GT), USA. The other one consists of HTTP request toward the website of our department (DIEE) at the University of Cagliari, Italy.

They consist respectively of seven and six days of traffic. It is worth to remark that both the GT and the DIEE datasets are completely unlabeled. We considered the GT and DIEE datasets as "clean" from known attacks for the purpose of measuring the false positive rate since any evidence of occurring attacks has not been reported in the period in which we collected the traffic.

The experiments have been carried out in the same way on both datasets, for what concerns both training and testing. A k-fold cross validation has been realized, using in rotation one day of traffic for training and all the remaining days for testing purposes. Details about the number of packets and the size (in MB) of each trace are provided in table 2.

We evaluated the detection rate of the IDS on several datasets consisting of attacks frequently observed against web applications. Attack datasets are briefly

Table 2. Details of the legitimate traffic datasets used for the training and to evaluate the false alarms rate.

Dataset	*Day*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
DIEE	**Packets**	10,200	10,200	10,200	10,200	10,200	10,200	—
	Size (MB)	7.2	7.4	6.6	6	6.4	6.7	—
GT	**Packets**	307,929	171,750	289,649	263,498	195,192	184,572	296,425
	Size (MB)	131	72	124	110	79	78	127

Table 3. Details of the attacks datasets used to evaluate the detection rate.

Dataset Name	# of Attacks	Description
Generic **A**ttacks	66	Shell-code, Denial of Service or Information Leakage
Shell-code Attacks	11	Shell-code attacks from the Generic Attack dataset
XSS-SQL Attacks	38	Cross-site Scripting and SQL-Injection attacks

described in Table 3. Generic and Shell-code attacks are the same used in [[2, 9]]. Attacks into the XSS-SQL dataset are the same used in [[2, 5]].

Performance Evaluation In order to validate the classification performance of our detector, we use the ROC curve analysis, and the related Area Under the ROC Curve (AUC). Since we are interested in evaluating the IDS for small values of the false positive rate, we computed the area under the ROC curve in the range $[0, 0.1]$ of the false positive rate. In order to obtain a performance value in the range $[0, 1]$, we normalized the "partial" AUC (AUC_p) computed in $[0, 0.1]$ by dividing it by 0.1.

5 Experimental Results

This section provides a discussion of the experimental results achieved. The performance, evaluated in terms of AUC_p, has been calculated considering several one-class classification algorithms. In addition, we also varied the number of HMM within each ensemble. A number of HMM from 1 to 3 has been considered.

We first considered the static rules as a possible choice for the one-class classifier. We considered two rules, respectively the average and the product rules. The missing features have been excluded from the computation. For the sake of brevity we report just some examples of the results achieved.

The AUC_p was equal to 0.440 on the **Generic** attacks and equal to 0.464 on the **Shell-code** attacks (DIEE legitimate traffic) for the average rule. The results achieved using the product rule (and on the GT dataset) were equivalent. These results clearly show that static rules are not suitable in this scenario, thus confirming our choice of one-class classifiers as trainable fusion rules.

We experimented using several classifiers to combine the outputs produced by the HMM ensembles. In particular, we considered the Gaussian (**Gauss**) distribution, the Mixture of Gaussians (**MoG**), the **Parzen** density estimators, and the **SVM**. For the first three classifiers we used the implementation provided within the `dd_tools` . For the SVM, we used the implementation provided by `LibSVM`. We used a Radial Basis function for the SVM kernel. We left the setting of the other parameters to the default values.

Table 3(a) and 3(b) report the results achieved on the DIEE and GT dataset respectively. The same tables also reports the average values of AUC_p achieved by *HMMPayl* under the same conditions. From a deep comparison between

Dd_tools - `http://prlab.tudelft.nl/david-tax/dd_tools.html`

LibSVM - `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

(a) DIEE Dataset.

Attack Dataset	HMM	Gauss	Parzen	MoG	SVM	HMMPayl
Generic	1	0.656 (0.216)	0.931 (0.022)	0.874 (0.097)	0.843 (0.036)	0.922 (0.058)
	2	0.659 (0.207)	0.931 (0.023)	0.857 (0.125)	0.848 (0.029)	
	3	0.659 (0.226)	0.933 (0.021)	0.865 (0.129)	0.851 (0.031)	
XSS-SQL	1	0.937 (0.030)	0.941 (0.031)	0.923 (0.046)	0.838 (0.203)	0.847 (0.032)
	2	0.936 (0.030)	0.940 (0.033)	0.915 (0.059)	0.863 (0.175)	
	3	0.935 (0.030)	0.939 (0.034)	0.924 (0.046)	0.871 (0.161)	
Shell-code	1	0.935 (0.030)	0.946 (0.022)	0.923 (0.032)	0.889 (0.061)	0.996 (0.002)
	2	0.942 (0.033)	0.946 (0.022)	0.916 (0.028)	0.899 (0.055)	
	3	0.944 (0.035)	0.945 (0.023)	0.924 (0.028)	0.908 (0.056)	

(b) GT Dataset.

Attack Dataset	HMM	Gauss	Parzen	MoG	SVM	HMMPayl
Generic	1	0.686 (0.107)	0.920 (0.082)	0.915 (0.035)	0.801 (0.102)	0.866 (0.071)
	2	0.695 (0.087)	0.922 (0.087)	0.917 (0.037)	0.809 (0.095)	
	3	0.709 (0.024)	0.923 (0.093)	0.919 (0.028)	0.816 (0.093)	
XSS-SQL	1	0.718 (0.107)	0.972 (0.018)	0.870 (0.055)	0.806 (0.043)	0.827 (0.056)
	2	0.725 (0.095)	0.972 (0.018)	0.896 (0.052)	0.813 (0.037)	
	3	0.737 (0.083)	0.973 (0.018)	0.904 (0.037)	0.816 (0.030)	
Shell-code	1	0.848 (0.060)	0.928 (0.079)	0.930 (0.044)	0.909 (0.073)	0.988 (0.003)
	2	0.837 (0.041)	0.926 (0.084)	0.910 (0.043)	0.917 (0.075)	
	3	0.837 (0.036)	0.925 (0.088)	0.909 (0.043)	0.917 (0.072)	

Table 4. Average and Standard Deviation values of AUC_p. The rightmost column reports the performance achieved by *HMMPayl* [2].

HMMPayl and other similar algorithms (e.g [5, 9, 11]) *HMMPayl* resulted as the most effective IDS on the same datasets we used here [2]. Thus, in this paper we consider only *HMMPayl* for the sake of comparison.

It can be easily observed that if we exclude the case on which the **Gauss** classifier is used, the proposed solution performs generally well with respect to *HMMPayl*. A result which is worth to notice is that achieved using the **Parzen** classifier. In fact in this case the IDS works significantly better than *HMMPayl* against the **XSS-SQL** attacks (especially on the GT dataset) and it works better also against the **Generic** attacks. On the contrary, *HMMPayl* performs better when **Shell-code** attacks are considered. This is not surprising since *HMMPayl* basically creates a detailed model of the bytes' distribution of the payload, that in the case of the **Shell-code** attacks significantly deviates from that of the legitimate traffic. Nevertheless, we are quite convinced that the effectiveness of our IDS can be easily improved also against attacks of this type by designing more carefully the HMM ensemble. In fact, we observed that for certain headers the length of the sequences can variate heavily from payload to payload. Since the probability assigned by a HMM to a sequence significantly depends on the sequence length, a model that takes into account also the length of the header would be probably preferable in the case of those headers. It must be also considered that the one-class classifier can be further optimized since we left the setting of the parameters to the default values.

We can also notice that increasing the number of classifiers within the HMM ensembles does not provide remarkable benefits. It can be observed that in some cases the AUC_p increases with the number of HMM (e.g. in the **SVM** column) whereas in other cases the AUC_p slightly reduces (e.g. in the **Parzen** column). Notwithstanding, the observed variations are very low for both the average and the standard deviation of the AUC_p.

6 Conclusions

This paper we proposes an IDS that models the HTTP payload structure for the purpose of detecting the attacks against a web server. The IDS heavily relies on the MCS paradigm, since the outputs provided by a set of HMM ensembles are combined using a one-class classifier. The experimental results achieved confirm the effectiveness of the proposed solution and also show that the IDS works generally better than analogous algorithms. In addition, as a consequence of its small complexity, this IDS would be easily implemented in a real system.

References

1. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1, 1999.
2. D. Ariu, R. Tronci, and G. Giacinto. HMMPayl: An intrusion detection system based on Hidden Markov Models. *Computers & Security*, In Press, 2011.
3. L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.
4. B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for adversarial classification tasks. In *MCS*, volume 5519 of *LNCS*, pages 132–141. Springer, 2009.
5. I. Corona, D. Ariu, and G. Giacinto. HMM-Web: A framework for the detection of attacks against web applications. In *IEEE International Conference on Communications*, Dresden, Germany (2009).
6. C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *ACM conference on Computer and Communications Security*, New York, NY, USA, 2003.
7. C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738, 2005.
8. G. L. Marcialis, F. Roli, and L. Didaci. Personal identity verification by serial fusion of fingerprint and face matchers. *Pattern Recognition*, 42(11):2807 – 2817, 2009.
9. R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864 – 881, 2009.
10. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
11. Y. Song, A. D. Keromytis, and S. J. Stolfo. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. In *NDSS*. The Internet Society, 2009.
12. J. Friedman T. Hastie, R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition)*. Springer, 2009.
13. K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *RAID*, volume 3224 of *LNCS*, pages 203–222. Springer, 2004.