HMM-Web: a framework for the detection of attacks against Web applications

Igino Corona, Davide Ariu and Giorgio Giacinto

Abstract—Nowadays, the web-based architecture is the most frequently used for a wide range of internet services, as it allows to easily access and manage information and software on remote machines. The input of web applications is made up of queries, i.e. sequences of pairs *attribute←value*. A wide range of attacks exploits web application vulnerabilities, typically derived from input validation flaws. In this work we propose a new formulation of query analysis through Hidden Markov Models (HMM) and show that HMM are effective in detecting a wide range of either known or unknown attacks on web applications. In addition, despite previous works, we explicitly address the problem related to the presence of noise (i.e., attacks) in the training set. Finally, we show that performance can be increased when a sequence of symbols is modelled by an *ensemble* of HMM. Experimental results on real world data, show the effectiveness of the proposed system in terms of very high detection rates and low false alarm rates.

Index Terms—Multiple Classifiers System, Intrusion Detection, Web Applications, Hidden Markov Models

1 Introduction

Web-based architectures are frequently used for internet services. The core of web-based services is the "web application", i.e. a piece of software which processes in real time web user inputs (queries) and returns some type of informative content (e.g. an HTML page). In general, a web application query is a sequence of pairs attribute←value. The ubiquity of web-services, makes them extremely exposed to remote attacks [14], in particular due to input validation flaws. These flaws can be exploited by remote attackers to either access confidential information, or acquire high level privileges on targeted servers, or slow down or completely disrupt services [16].

Currently deployed security solutions typically rely on Web Application Firewalls (*WAF*, e.g. `ModSecurity` [2]), which filter application inputs using a predefined set of rules, and signature-based Intrusion Detection Systems (*IDS*, e.g. `Snort` [15]), which detect attacks using a set of (known) attack signatures. Unfortunately, these systems can be evaded modifying known attacks or exploiting new vulnerabilities [6].

In this work we propose an anomaly-based IDS for protecting web applications against attacks that exploit input validation flaws. To this end, we train HMM on normal (legitimate) queries on web applications and during the detection phase discriminate between normal and anomalous (attack) queries. In particular, for each web application, we model two main features of its queries: the sequence of attributes, and, for each attribute, its input strings.

The proposed IDS is trained in an *unsupervised* way, by processing web application queries as logged by the web server (training set). Such an approach relies on the assumption that the great majority of web application queries in the training set is legitimate. This is a reasonable assumption because the typical web user accesses services as expected.

Our original contribution is a system design which is resilient to the noise in the training set, where the *noise* is made up of attacks. Our system not only is able to effectively model the legitimate traffic, but also to detect attacks that are similar to the *noise* in the training set. To this end, we optimise IDS parameters on the basis of the fraction of non-legitimate queries we expect in the training set. Experimental results show that even a raw estimate for this parameter can effectively enhance the detection rate, with a small amount of false positives.

Another contribution of this paper is the codification of the input sequence analysed by HMM. Many works in the security area simplify the intrusion detection problem by training classifiers on raw data, skipping some known semantics of data. On the other hand, we explicitly codify queries to light out the most relevant "semantic" aspects of their structure. This operation substantially leads to better IDS performances and to a complexity reduction of the learning task for HMM.

Finally, the proposed IDS exploits the Multiple Classifier System (MCS) paradigm [10] and applies a novel solution to correlate HMM based on the *a priori* knowledge of this intrusion detection problem.

The paper is organised as follows. A review of the related works is reported in sec. 2. A detailed description of the proposed IDS is presented in sec. 3. Experimental results are reported in section 4. Conclusions and future work are drawn in sec. 5.

2 Related Works

The variety of applications based on HMM is very large, and goes from applications in speech recognition [11]

• *The authors are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi 09123 Cagliari, Italy. E-mail: {igino.corona, davide.ariu, giacinto}@diee.unica.it.*

to DNA-sequences modelling [4]. The power of HMM resides in their solid mathematical foundations, that make them powerful and resistant to noisy data [17, 11].

In last years, HMM have been applied to computer security mostly in host-based IDS to model system call sequences [5]. Anyway, there are also applications of HMM for network-based detection. In [3] a solution based on MCS paradigm is proposed for detecting attacks against an FTP server by analysing the exchange of commands between a client and a server machine. Recent studies showed experimentally that the MCS paradigm can outperform single classifier systems, being an MCS more difficult to evade and able to reduce the false positive rate [13].

Previous works on anomaly-based detection of attacks against web services, as well as for other network services, relied mostly on the analysis of network traffic (network-based) [7, 12]. Recently, a more specific (host-based) analysis have been proposed [9, 8], by analysing web server logs. From an operational point of view, the context of the proposed system is the same as [9, 8]. In particular, we compare our approach with respect to that in [9], which is the most similar work.

In [9], the authors proposed a multi model framework for the detection of attacks against web applications. They modelled (legitimate) queries using both *spatial* features (related to a single request), and *temporal* features (related to multiple consecutive requests). Different models to represent these features were applied, being the HMM the more powerful of them. In particular, they codified web application queries by extracting the sequence of attributes, and for each attribute, its input, codified as a sequence of characters. HMM have been used to model attribute inputs without taking into account typical *semantic* differences between classes of characters (alphabetic, numeric, non-alphanumeric), which usually determine their meaning. Moreover, the authors definitely did not exploit the powerful of such a model, because they rounded every non-zero probability value to one. Finally, the training set was assumed *without* attacks, by filtering it with a signature based IDS, in order to throw out at least known attacks.

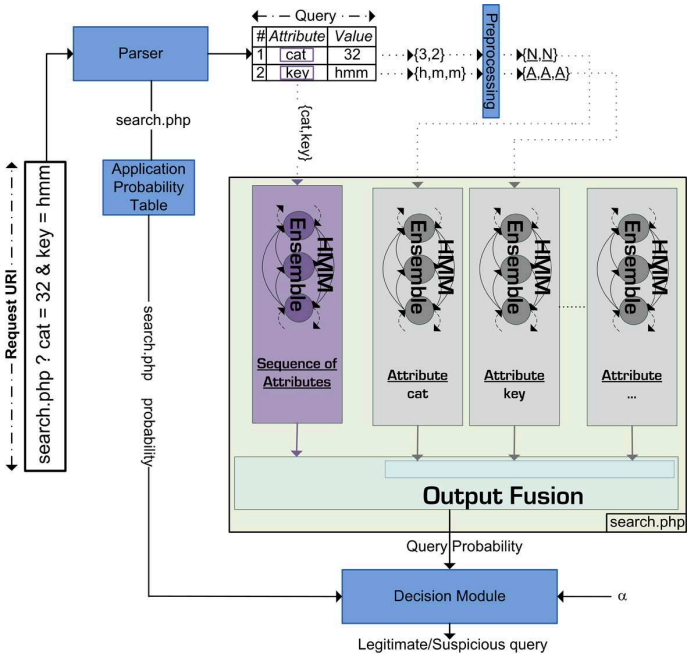

Fig. 1. IDS scheme. The Parser processes the request URI and identifies the web application (i.e. search.php) and its input query. Applying a threshold on the probability value associated to the codified query, it is labeled as legitimate/anomalous. The threshold depends on the web application probability and the α parameter.

3 The Proposed IDS

Our aim is detecting both simple and sophisticated attacks against each web application that resides on a web server. Focusing on this goal, the IDS is composed by a set of (independent) application-specific modules. Each module, composed by multiple HMM ensembles, is trained using queries on a specific web application and, during the operational phase, outputs a probability value for each query on this web application. Furthermore, a decision module classifies the query as suspicious (a possible attack) or legitimate, applying a threshold to this probability value. Thresholds are fixed independently for each application-specific module.

The following sections provide details about (a) the feature extraction process, (b) the application-specific modules, (c) decision module and (d) the building of HMM ensembles. Throughout these sections we will refer to fig. 1, where it is showed the IDS processing for a `search.php` application, which could be used to list publications of a certain category (`cat` attribute), containing a key-word (`key` attribute). It may be finally useful to remark that in some works the term web application is used to identify the a set of executables/scripts that offers a certain number of services (e.g. a search engine: main page, database interrogation, images generation). For the sake of clarity, in the following we will refer to each program or script which generate dynamic web contents, as a different web application.

3.1 Feature extraction

Web application queries are extracted from web server logs. In particular, we select all requests where the method is `GET` and that receive a successful response (status code 2xx, as described in RFC2616). Then, the web application and its input query are obtained from the Uniform Resource Identifier (URI), by considering the web server configuration and its URI parsing routine.

For a generic query, the sequence of attributes and, for each attribute, its input string, are extracted. Regarding the sequence of attributes, we consider each attribute as a symbol of the sequence, as it is enough to detect anomalies either in the order or in the presence of suspicious attributes. On the other hand, it is useful to provide for a more complex codification of attribute inputs w.r.t. the

simple extraction of the character sequence. By scrutinising attacks against web applications (`cve.mitre.org`), it is evident that, typically, non-alphanumeric characters have higher relevance than alphanumeric characters, when interpreting the *meaning* of attribute inputs. Non-alphanumeric characters could be used as *meta-characters*, with a special "meaning", during the processing made by web applications. Thus, a distinction between them (e.g. between "/" and "-") is definitely necessary. On the contrary distinguishing between digits or between alphabetic letters is not useful to detect input validation attacks. Consequently, we substitute every digit with the symbol N and every letter with the symbol A, while all the other characters remain the same. For example, the attribute value "`/dir/sub/1,2`" becomes "/AAA/AAA/N,N". The obtained sequence of symbols is then processed by HMM.

3.2 Application-specific modules

As shown in fig. 1, each application-specific module consists of: (1) an HMM ensemble which analyse the sequence of attributes (i.e. {`cat, key`}); (2) for each attribute found in training queries, an HMM ensemble which analyses the input sequence for that attribute (i.e. for `cat`, {N, N}, for `key`, {A, A, A})[1]. Thus, each HMM ensemble models a different feature of the web application query. Our goal is to detect an anomaly in *any* of these features of the query, because they can be related to a different attack typology. To this end, we apply the *minimum* rule between the HMM ensemble outputs.

3.3 Decision module

For the aim of the following discussion, let us define for a specific training set:

- M the number of web applications.
- $q(w_i)$ the set of queries for the i-th web application w_i, $\forall i \in [1, M]$. $|q(w_i)|$ is the number of queries for w_i.
- $N = \sum_{j=1}^{M} |q(w_j)|$ is the total number of queries collected.

The decision module classifies a query as suspicious if its probability is under a threshold, otherwise, the query is classified as legitimate.

Now, it is important to note that, for a certain web application, the basic assumption that a considerable amount of its training queries is legitimate *may not be valid*. The attacker may exploit web applications which are rarely interrogated by users, to perform some unauthorised action. For example, this is the case of applications for testing/configuration purposes ([1], exploits: 6287, 6314, 6269, 5955). If such a kind of attacks are inside the training set, in the worst case, we model only instances of *attack queries* (instead that legitimate queries) for the web applications involved.

To cope with this problem we consider the relative frequencies of queries toward each web application $freq(w_i) = |q(w_i)|/N$. This frequency reflects in some way how strong is the assumption that its queries are actually legitimate. The higher this frequency, the stronger the assumption that its queries are actually legitimate. In the detection phase, the frequency $freq(w_i)$ represents an estimate of the probability of (a query on) the web application w_i and it is stored in a look-up table (fig. 1).

If we expect an overall fraction of attack queries α in the training set, it will be equal to:

$$\alpha = \frac{1}{N} \sum_{i=1}^{M} \alpha_i \cdot |q(w_i)| \tag{1}$$

where α_i is the fraction of suspicious training queries toward w_i. The simplest solution may be $\alpha_i = \alpha$, that is, an equal fraction α of training queries is classified as suspicious by each application-specific module. However, this setting does not take into account how strong is the assumption that training queries are really legitimate.

Aiming at including this information, for each application-specific module $S(w_i)$, we fix a threshold t_i so that the value of α_i is in inverse proportion with respect to $freq(w_i)$. In this case, in agreement with eq. 1, α_i are given by

$$\alpha_i = \frac{\alpha}{M \cdot freq(w_i)} \quad \forall i \in [1, M] \tag{2}$$

It is easy to see that, with such a setting, the smaller the frequency of a web application, the larger the fraction of training queries classified as suspicious. In other terms, the weaker the assumption that web application queries are legitimate, the bigger the fraction of training queries classified as suspicious.

Thus, the α parameter is used to estimate the threshold t_i to be used in the *detection phase*, as α can be considered as an overall "confidence factor" for the legitimacy of training queries.

3.4 HMM building

In this work we address only two out of the three basic problems for HMM: the *Learning Problem* (during the training phase) and the *Evaluation Problem* during the detection phase. We use the well-known Baum-Welch algorithm to train HMM [11]. As the algorithm may find only a local minima of the likelihood function (that is, the HMM models well only a subset of training sequences) we use an ensemble of HMM in order to better model the whole training set. Moreover, considering that HMM performance depend on parameters as *number of states*, *initial state*, *symbol emission* matrix, *state transition* matrix. As the estimation of the best suited values of HMM parameters is more art than science, the use of an ensemble of HMM can counterbalance this lack of knowledge.

1. Braces and commas are not part of the sequence, we use them just to represent it.

TABLE 1
References for attacks inside A, see [1].

Attack Type	Exploit N.	Paper N.
SQL Injection	6512, 6510, 6502, 6490, 6469, 6467, 6465, 6449, 6336, 3490, 5507	16, 174, 202, 215
XSS	2776, 2881, 2987, 3405, 3490, 4681, 4989, 6332	162, 173, 192

We set an equal number of states for each HMM inside an ensemble. This number is equal to the average length of training sequences, rounded to the nearest greater integer. Also, an *effective* length definition is used; the length of a sequence is given by the number of different symbols in this sequence. For example, in the sequence $\{a, b, c, b, c\}$ there are 3 different symbols, a, b and c. Consequently 3 is the effective length for this sequence. In consequence of the heuristics we used to fix the number of states, each state can be associated to an element of the analyzed sequence, rather than a particular state of the web application.

Both the state transition and the symbol emission matrices are *randomly* initialised. Considering IDS performance, this choice seems to be reasonable. In fact, our IDS consists of a large number of HMM, and using the a priori knowledge of the problem to model the structure of matrices could be a time and effort expensive task. Finally, we build the dictionary of symbols by extracting them from training sequences. HMM in the same ensemble use the same dictionary.

3.5 Fusion of HMM outputs

In principle, the best fusion rule for HMM inside an ensemble is unknown. However, it may be useful to refer to a theoretical analysis of HMM outputs. Given an input sequence s, the output of the i-th HMM m_i (out of K HMM) in the ensemble can be written as $p(s|m_i) = p(m_i|s) \cdot p(s)/p(m_i)$.

We could set the same *a priori* probability for all models, that is, $p(m_i) = c, \forall i \in [1, K]$. This can be a reasonable assumption as in principle all models are equally valid. It is easy to see that, when using the *maximum* fusion rule (output=$\max\{p(s|m_i), i \in [1, K]\}$), the output is proportional to $\max\{p(m_i|s), i \in [1, K]\}$ (the term $p(s)$ is a constant). So, using the maximum rule, we may select the model in the ensemble that best "describes" the analyzed sequence to compute the probability of the sequence. This reasoning is in agreement with the original goal of an HMM ensemble, that is, to exploit the *diversity* of multiple HMM to better modelling the whole set of training sequences.

4 Experiments

4.1 Dataset and performance evaluation

In order to test our IDS in a realistic scenario, we collected a data set of queries from a production web server of our Academic Institution. Let us call D this dataset. It consists of more than 150,000 queries collected in a period of six months. Queries are distributed over a total of 52 web applications. In particular, 24 of these applications provide services for registered users, the remaining 28 provide public services. As D consists of a set of real requests from a web server log files, it may contain both normal and attack queries. Our first goal is to assess IDS performance in terms of false alarm rate and detection of attacks similar to those which may be inside the training queries. To this end, each query inside D has been labelled as attack or legitimate, through a semiautomatic process, further described in sec. 4.1.1. Furthermore, D has been split randomly into 5 parts (all containing the same number of queries) in order to apply a 5-fold cross-validation. As we are dealing with real traffic each split of D will contain unknown attacks and, in consequence of the random sampling, we assume the percentage of attacks being the same in all of five splits. Exploiting the labelling process, we evaluated both the false positive (FPR) and the detection rate (DR) on D.

In order to evaluate the detection rate, we selected a set of attacks published on [1] and used these as a basis to build a dataset (called A) of attacks *exploiting* specific vulnerabilities on our set of applications. This dataset consists of 19 SQL Injection and 19 Cross-site Scripting (XSS) attacks, on a subset of 18 web applications (see table 1).

It is worth to note that, due to privacy issues and the problem formulation, public data sets are not available. Anyway, there are attacks as those referred in [8] that are related to known vulnerabilities of widely deployed and open-source web applications. On the other hand, in practice, web applications which manage critical information (i.e. public administration, home banking) are typically highly customised and their source code is not public. This reflect also our case, and it is the reason why attacks inside A are just derived from well know attacks, representing a version of them customised against applications in our set.

4.1.1 Attack queries inside dataset D

In order to distinguish between attacks and legitimate queries in the training set we exploit the IDS itself. As attack queries are typically in low number w.r.t. legitimate queries, and their structure is typically different (this is definitely evident from the working exploits in table 1), we expect that they will receive lower probability than legitimate queries. In fact our experiments fully confirmed this behaviour. However, it may not be possible to fully automate the labelling process without fall into error, i.e. simply because data set D does not contain enough information to do that.

Thus, for each web application w_i having a relatively high number of queries, we identified (and labelled) attack queries inside the training set by manually inspecting queries which receive the lowest probability. For web applications having a relatively low amount of queries, we inspected all training queries, because an

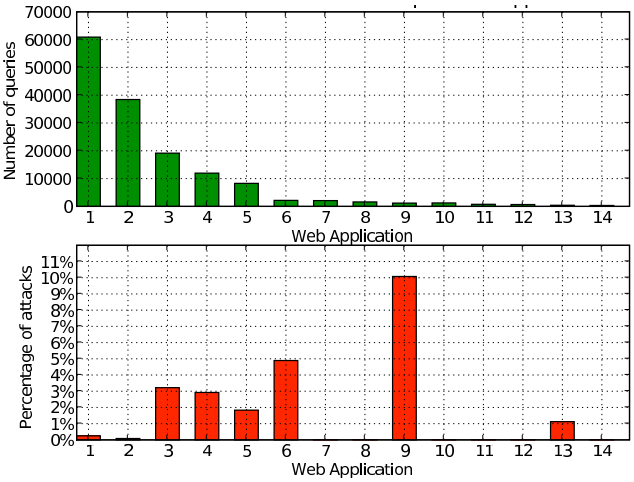

Fig. 2. Distribution of queries and percentage of attacks for the 14 most frequent web applications.

attack query could not receive lower probability than a legitimate query (i.e. the majority of queries are attacks), as discussed in sec. 3. Spotting attack queries inside D, we exploited additional information.

We assessed legitimate inputs by links contained inside web pages, when browsing as a typical user. Moreover, evidencing attack queries, we exploited our expertise regarding typical attacks and perhaps the corresponding output generated by web applications. In such a way, for each application w_i we computed the corresponding fraction of attacks α_i^* inside D.

Using such a method, we found an overall fraction of $\alpha^* = \sum_{i=0}^{M} \alpha_i^* = 0.995\%$ of attacks inside D. These attacks were very similar and were mainly related to the injection of HTML code.

In addition, the graphics on figure 2 confirm our intuition that the the lower the number of queries, the more weak is the assumption that they are legitimate. In fact, we observe that as the number of queries toward a certain application decreases, the percentage of attacks between them tend to increase (about 10% of attacks for the 9th application).

4.2 Results

In this section we summarise experimental results when (1) a single HMM and (2) multiple HMM are used to model a generic sequence. Our IDS has been always able to detect *all* attacks inside dataset A, so in the following we will focus our analysis on the evaluation of FPR/DR on dataset D (average values over all splits).

Fig. 3 shows the performance of the proposed IDS for the option (1), either with the query codification used in [9] or with the proposed codification. As we can see, the proposed query codification is really effective to heavily reduce the false alarm rate. In particular, we computed IDS performance for different values of α, as in real scenarios we cannot rely on a reliable estimation, but only on raw estimates. On the other hand, with a little

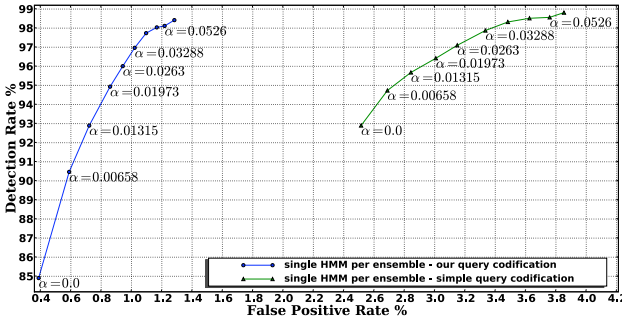

Fig. 3. Average DR and FPR for different values of α and a single HMM per ensemble.

expense in terms of false alarm rate, a positive value of α definitely enhance the detection rate of attacks. It is evident that a precise value is not necessary, because even with a relatively large value, i.e. $\alpha \sim 3\%$ (we know that attacks are about the $\alpha_i^* \simeq 1\%$), we are able to both detect 96% of attacks, and raise a fraction of false alarms lower than 1%. The point $\alpha = 0$ identifies IDS performances when we are fully confident on the legitimacy of the training queries, as in [9]. It is evident that for the proposed query codification, the lowest amount of false alarms is obtained (about 0.4%), but a significant part (about 15%) of attacks inside D cannot be detected.

The proposed codification is advantageous also in terms of learning time: just 168 minutes against 214 (on a computer with CPU CoreDuo2 8100@2.1Ghz, 2GB RAM, O.S. Linux). Anyway, the learning time doesn't affect IDS performances, as training is made off-line.

Results obtained with a single HMM, are compared with those obtained while using 3, 5 and 7 HMM per ensemble in fig. 4. For small values of α, if the number of HMM per ensemble increases, there is an improvement of performance. This is a reasonable result, because using more models the IDS is able to better modelling the information inside the training set. However, the improvement of performance is not as large as we expected. This because the proposed query codification (sec. 3.1), which simplifies the learning task for HMM, reduces the advantages of modelling a generic sequence using multiple HMM. In fact, in some preliminary experiments we performed using the codification used in [9], the improvement of performance of multiple HMM was larger. Moreover, as the value of α increases, the 4 curves become more close each other. This may be explained if we note that the increasing of α can lead to a heavy modification thresholds (in a complex relationship), which may overwhelm advantages of a more thorough computation of the query probability.

5 Conclusions and Future Work

In this work we propose an anomaly based IDS based on HMM. This IDS is designed to detect attacks against web applications by analysing the URI of GET requests

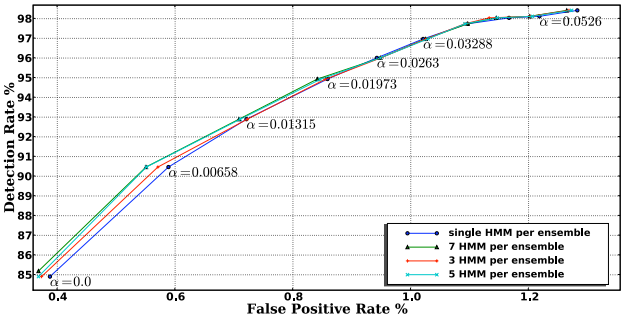

Fig. 4. Average DR and FPR for different values of α either with single or multiple HMM per ensemble.

toward a web server. With a suitable query codification and a well-suited MCS design, very low FPR and very high DR are achieved. The proposed solution is also very efficient both during the training and detection phase.

Despite previous works, we explicitly address the problem related to the presence of attacks *inside* the training set. Experimental results show the effectiveness and the simplicity of the solution we propose to deal with this problem.

We evidence that IDS performance can be improved when an *ensemble* of HMM, w.r.t. a single HMM, even if results show that the higher is α the lower are benefits of using an HMM ensemble.

Our system provides several advantages respect to *WAF*: it can be trained in an automatic way, while *WAF* rules must be written (and updated) by hand, which can be a hard task due to the complexity of the structure of web application inputs. In addition, even if well documented secure-coding practices have been devised, it is still difficult to prevent input validation attacks, like SQL-Injection and XSS. Of course, our IDS can be also effective against attacks other than XSS and SQL-Injection, but the related experiments would require a different dataset (other web applications, with different vulnerabilities).

A possible improvement of this system may be related to some automatic cleaning of the training set. To this end, we are studying how to automatically retrain the IDS discarding (a little amount of) training queries which may be related to attacks (i.e. with lowest probability).

References

[1] milw0rm, www.milw0rm.com.

[2] Modsecurity, www.modsecurity.org.

[3] Ariu D., Giacinto G., and Perdisci R. Sensing attacks in computers network with hidden markov models. *MLDM 2007*, LNAI 4571:795–809.

[4] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge Univ. Press.

[5] H.H. Feng, O.M. Kolesnikov, P. Fogla, and W. Lee. Anomaly detection using call stack information. *Security and Privacy, 2003. Proceedings*, pages 62–75.

[6] Vigna G., Robertson W., and Balzarotti D. Testing network-based intrusion detection signatures using mutant exploits. In *ACM CCS*, pages 21–30, 2005.

[7] Wang K. and Stolfo S.J. Anomalous payload-based network intrusion detection. *RAID 2004*, LNCS 3224:203–222.

[8] Ingham K.L., Somayaji A., Burge J., and Forrest S. Learning dfa representations of http for protecting web applications. *Computer Networks*, 51:1239–1255, 2007.

[9] 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.

[10] Kuncheva L. *Combining Pattern Classifiers*. Wiley, 2004.

[11] Rabiner L.R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[12] Lippmann R., Haines J., Fried D., J. Korba, and Das K. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34:579–595, 2000.

[13] Perdisci R., Gu G., and Lee W. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. *ICDM*, 2006:488–498.

[14] www.sans.org Roger Meyer, SANS institute. Detecting attacks on web applications from log files, 2008.

[15] Snort. open-source ids, www.snort.org.

[16] OWASP www.owasp.org. The ten most critical web application vulnerabilities, 2007.

[17] Yoshua Bengio Y. Markovian models for sequential data, 1996.