HMMPayl: an application of HMM to the analysis of the HTTP Payload

Davide Ariu DAVIDE.ARIU@DIEE.UNICA.IT
Giorgio Giacinto GIACINTO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Italy

Editor: Nello Cristianini, John Shawe-Taylor and Tom Diethe

Abstract

Zero-days attacks are one of the most dangerous threats against computer networks. These, by definition, are attacks never seen before. Thus, defense tools based on a database of rules (usually referred as "signatures") that describe known attacks cannot do anything against them. Recently, defense tools based on machine learning algorithms have gained an increasing popularity as they offer the possibility to fight off also zero-days attacks. In this paper we propose *HMMPayl*, an anomaly based Intrusion Detection System for the protection of a web server and of the applications the server hosts. *HMMPayl* analyzes the network traffic toward the web server and it is based on Hidden Markov Models. With this paper we provide for several contributions. First, the algorithm implemented by *HMMPayl* allows to carefully model the payload increasing the classification accuracy with respect to previously proposed solutions. Second, we show that an approach based on multiple classifiers leads to an increased classification accuracy with respect to the case where a single classifier is used. Third, exploiting the redundancy within the information extracted from the payload we propose a solution to reduce the computational cost of the algorithm.
Keywords: Intrusion Detection, Anomaly Detection, HMM, Multiple Classifiers Systems

1. Introduction

The always increasing number of Web-based applications that are deployed worldwide, makes their protection a key topic in computer security. Unfortunately, the large number of new attacks that appear everyday makes almost impossible to have signature-based systems always updated to the most recent attacks. A possible solution to the problem is offered by the "anomaly based" systems. An anomaly based system creates 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.

. 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.

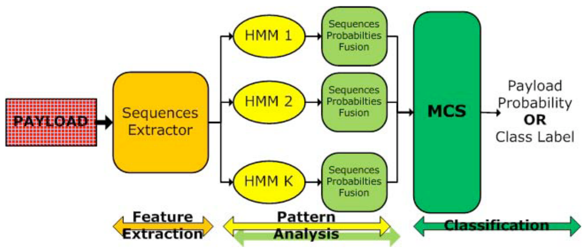

PAYLOAD
Sequences Extractor
HMM 1
HMM 2
HMM K
Sequences Probabilties Fusion
Sequences Probabilties Fusion
Sequences Probabilties Fusion
MCS
Payload Probability OR Class Label
Feature Extraction
Pattern Analysis
Classification

Figure 1: A simplified scheme of *HMMPayl*.

If the resource to be protected is a web server (including the related web applications) its normal behavior might be represented using a statistical model of the distribution of bytes within the HTTP payload, which is the portion of the network packet that contains the request sent by the web browser to the web server. The key idea is that the statistics of a payload containing an attack are different from those of a legitimate one. The seminal work on Intrusion Detection Systems based on payload statistics has been proposed by Wang and Stolfo (2004). They proposed *PAYL*, which is an IDS that models the distribution of bytes within the payload using the n-gram-analysis. The main limitation of *PAYL* is represented by the size of the features space that exponentially increases with n (256^n) which makes impossible using a value of n larger than two. Perdisci et al. (2009) proposed an IDS called *McPAD*, which implements a feature extraction algorithm that can be considered as an approximation to the n-gram analysis. In particular the n-gram analysis is approximated combining an ensemble of classifiers each one of which works in a features space of size 256^2.

In this paper we propose an IDS that has been inspired by *PAYL* and *McPAD*. It uses Hidden Markov Models to produce an accurate model of the payload. One of the reasons which initially motivated its development was the consideration that HMM have the same expressive power of the n-grams. The advantage of using HMM is the possibility of modeling sequences of bytes without the limitations of computational cost of which the n-gram analysis suffers. The resulting model is able to better take into account the structure of the payload. From a practical perspective this leads to an improved classification accuracy and to a higher difficulty of evading the IDS. The remainder of the paper is organized as follows. In Section 2 we provide for a description of HMMPayl. Section 3 provides a description of the experimental setup. In Section 4 we describe the experimental results. Then, we draw conclusions in Section 5.

2. IDS Architecture

This section provides an outline of the architecture of *HMMPayl* which is also represented in Figure 1. A more comprehensive description can be found in [Ariu (2010)]. Here we will describe only the *Detection* phase of *HMMPayl*, as during the *Training* phase emission and transition matrices of HMM are estimated using the well known Baum-Welch algorithm [Rabiner (1989)]. We exploited the fact that this algorithm only finds a local maximum of the likelihood function to obtain ensembles of HMM that are diverse even if they are trained on the same training set. In the following three paragraphs we will describe the three different steps we devised for processing the payload.

Step 1: Feature Extraction At a low level, an HTTP payload is an array of L bytes, where the value of L is variable depending on the web content that is transferred. We want to extract from the payload a set of sequences of equal length and we propose to do it using a sliding window of width "n". The window slides over the payload byte by byte, and each group of n bytes that falls inside the window is considered as a sequence that we will call s_j. Consequently, a payload P of length L is represented by a set S of N sequences of length n, where $N = L - n + 1$.

Step 2: Pattern Analysis The input for this step is the set S of N sequences extracted from a payload P. Let us assume that an HMM is trained on sets of sequences extracted from normal payloads according to the technique previously described. We assume also that p_j is the probability (for an HMM λ) of emitting each one of the sequences inside S (at detection time).

Thus, for each payload P, the HMM produces a set of N probabilities associated to the N sequences in S. A simple solution to obtain an overall probability for P is to calculate the *arithmetic mean* of the output probabilities:

$$P(S|\lambda) = \frac{1}{N}\sum_{j=1}^{N} p_j = \frac{1}{N}\sum_{j=1}^{N} P(s_j|\lambda) \quad \textit{with} \quad p_j = P(s_j|\lambda),\ j = 1, ..., N \tag{1}$$

as the arithmetic mean provides an efficient and unbiased estimate.

The scheme of *HMMPayl* reported in Figure 1 shows that K different HMM are used in parallel during the Pattern Analysis step. Each HMM within the ensemble receives as input the whole set of sequences S, and produces as output a set of N probabilities. The set of output probabilities from each HMM is the input for the "Sequences Probabilities Fusion" block which calculates the arithmetic mean of the N probabilities produced by each HMM. At the end of this step we obtain a vector of K probabilities assigned to the payload by the K HMM. All the HMM in the ensemble have the same number of states, and are trained on the same training set, while their transition and emission matrices are different due to a different random initialization.

Step 3: Classification At this level, the problem is that of combining the outputs of several classifiers. To keep the system simple, we decided to use static combination rules, i.e., the minimum, the maximum, the mean, and the geometric mean rules [Kuncheva (2004)]. The resulting probability is then thresholded according to desired trade-off between detection and false positive rates.

3. Experimental Setup

In this Section we describe the characteristics of the datasets we used in our experiments.

HMMPayl has been deeply tested on three different datasets of normal traffic, and on four datasets containing different types of attacks. For what concerns datasets of normal traffic, one of them contains simulated traffic, while the other two contain real traffic collected at academic institutions. The dataset containing simulated traffic consists of the HTTP requests extracted from the first week of the DARPA'99 dataset [Lippmann et al. (2000)].

(*a*) Number of packets and size (MB) of traces of normal traffic

	DARPA		**DIEE**		**GT**	
Day	**Size (MB)**	**Packets**	**Size (MB)**	**Packets**	**Size (MB)**	**Packets**
1	19	161,602	7.2	10,200	131	307,929
2	23	196,605	7.4	10,200	72	171,750
3	23	189,362	6,6	10,200	124	289,649
4	30	268,250	6	10,200	110	263,498
5	18	150,847	6.4	10,200	79	195,192
6	–	–	6.7	10,763	78	184,572
7	–	–	–	–	127	296,425

(*b*) 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
CLET Attacks	144	Shell-code attacks modified with the polymorphic engine CLET
XSS-SQL Attacks	38	Cross-site Scripting and SQL-Injection attacks

Table 1: Datasets used for the evaluation of HMMPayl

Altogether it is composed of five entire days of simulated normal traffic to and from an air force base. For what concerns the real traffic, the first dataset is made up of HTTP requests towards the website of the College of Computing at the Georgia Tech (GT), USA. The other 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.

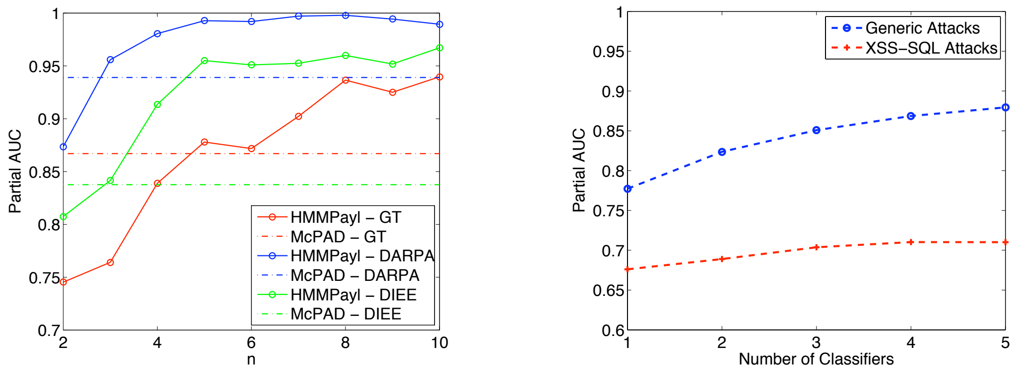
The experiments have been carried out in the same way on all of the three datasets, both for training and testing. A k-fold cross validation has been realized, using in rotation one day of traffic for training and all the remaining days for testing purposes. Details about the number of packets and the size (in MB) of each trace are provided in table 1(a).

We evaluated the detection rate of *HMMPayl* on several datasets consisting of attacks frequently observed against web applications. Attack datasets are briefly described in Table 1(b). Attacks in the Generic, Shell-code and CLET dataset are the same used in [Perdisci et al. (2009)]. Attacks into the XSS-SQL dataset are the same used in [Corona et al. (2009)].

4. Experimental Results

In this section we will briefly describe the attained experimental results. *HMMPayl* has been evaluated in terms of "Partial AUC" (AUC_p) that we defined as the AUC attained by integrating the ROC curve in the range [0 0.1] of the false positive rate, and then dividing the resulting value by 0.1. The other portions of the ROC curve are of no interest in computer security applications, as the related operating points exhibit too high values of false alarm rates. The rest of this section is organized as follows: in 4.1 we show how the

. T. Detristan, T. Ulenspiegel, Y. Malcom, and M. Underduk. Polymorphic shellcode engine using spectrum analysis. Phrack Issue 0x3d, 2003.



(a) AUC_p - Generic Attacks Dataset (b) AUC_p increase with the number of classifiers

Figure 2: Experimental Results on HMMPayl

classification accuracy increases with the length n of sequences. In 4.2 we will describe results obtained applying a sequences sampling strategy which is designed to reduce the computational cost. In 4.3 we will show benefits of the MCS approach.

4.1. The width of the window n

Table 2 shows that *HMMPayl* is particularly effective in detecting attacks within the Shell-code and the CLET dataset. These quite good results can be explained by observing that shell-code attacks contain executable code, i.e. non printable characters. Since the training set is made up of legitimate traffic, it mainly contains only printable characters. As a consequence, the probability emitted for non-printable characters is around zero, and the related payloads can be easily classified as anomalous. On the contrary, payloads containing attacks such as Cross-site Scripting or Denial of Service, produce payload statistic that are similar to those of legitimate traffic. Thus, a value of at least 5 must be set for n to attain a good detection rate of these attacks.

Figure 2(a) confirms this point. It shows the values of AUC_p obtained by *HMMPayl* on the Generic Attack dataset for different values of n. The graph clearly shows that the AUC_p attained by *HMMPayl* increases with the value of n. In fact, the larger the value of n, the better the model of the structure of the payload. We also compared *HMMPayl* with *McPAD*. *PAYL* has not been considered here because it has been already shown that *McPAD* outperforms *PAYL* when these attacks are considered [Perdisci et al. (2009)]. We reported (see Figure 2(a)) the performance attained by *HMMPayl* and *McPAD* when the

Table 2: Average AUC_p for HMMPayl. The average is calculated on different values of n and on different days of traffic. Classifiers are combined using the minimum rule.

	DARPA	**DIEE**	**GT**
Attack	$\overline{AUC_p}$ (σ)	$\overline{AUC_p}$ (σ)	$\overline{AUC_p}$ (σ)
Shell-code	0.999 (0.0019)	0.996 (0.0024)	0.988 (0.0035)
CLET	0.998 (0.0023)	0.997 (0.0051)	0.990 (0.0087)

minimum rule is used. Reported results show that *HMMPayl* can outperform *McPAD* on all of the three datasets of normal traffic. A worthy result is that the increase in the AUC_p is particularly noticeable on the datasets of real traffic (DIEE and GT) that are fairly harder to be modeled with respect to the DARPA dataset (which is synthetic).

4.2. Sequences Sampling

Here we propose to use a sample of the sequences in the set S to reduce the computational cost. We considered that at each step of the "Features Extraction" process, the sliding window extracts from the payload a sequence that contains the $n - 1$ last bytes of the sequence extracted at the previous step. Thus, the same byte results analyzed n-times by the HMM. Then we thought that was possible to move the sliding window of more than a single-step forward without loosing too much of the information extracted from the payload. In our experiments we set $n = 6$ and we varied the step of the sliding window from 1 (that is the original proposal with no sampling) to 6 (no overlap among sequences).

Table 3 shows the experimental results. In the worst case the loss in the AUC_p due to sampling is not higher than the 15% of the maximum AUC_p. It is easy to see that the maximum value is achieved when the whole payload is considered. The advantage attained by sampling is the reduced number of sequenced processed. In the case the step value is set to 6, the number of sequences to be processed approximately reduces to 17%, with a considerable saving of computational cost. Unfortunately, Table 3 also shows that the loss in AUC_p values is not easily predictable once the sampling step has been determined. A possible explanation for this can be provided if we consider that the information within the payload is basically expressed through an English-written text. Thus, some sampling steps allow to select a subset of sequences that better represents the information within the payload, without relationships with the number of sequences that have been discarded, (e.g. we could discard sequences that are aligned with the beginning of reserved words in the payload).

Table 3: Average AUC_p for different values of the sampling step on the GT dataset.

Attacks	**Advancing Step**	**1**	**2**	**3**	**4**	**5**	**6**
XSS-SQL Injection	AUC_p	0.855	0.732	0.733	0.723	0.736	0.727
Generic		0.870	0.820	0.780	0.801	0.745	0.785

4.3. Benefits of the MCS approach

In this section we show that using multiple classifiers we can realize more accurate attack detection. Figure 2(b) clearly shows that the AUC_p increases with the number of combined

Table 4: Value of the AUC_p for single classifiers. The MCS includes all of the five HMM combined with the minimum rule. Results are averaged on the GT dataset.

	HMM1	**HMM2**	**HMM3**	**HMM4**	**HMM5**	**MCS**
Generic Attacks	0.8036	0.8163	0.8564	0.8973	0.7944	0.918
XSS-SQL Attacks	0.8559	0.8584	0.6155	0.8221	0.6717	0.8546

classifiers. Figure 2(b), and Table 4 both refer to the first day of the GT dataset and to a value $n = 6$ for the width of the window. Results obtained on different days of traffic and for different values of n are similar.

We limited to five the number of classifiers in the ensemble as from some preliminary experiments we observed that a number of five HMM guaranteed a good trade-off between accuracy and computational cost. Especially the red curve in Figure 2(b) shows that further increasing the number of classifiers might not allow attaining significantly better values of AUC_p, whereas the computational cost becomes definitely higher.

In Table 4 we compare the AUC_p achieved by the single HMM with that obtained by the ensemble made up of all the generated HMM. The ensemble outperforms all the single classifiers on the Generic Attack dataset, while on the XSS-SQL dataset, it performs as good as the the best classifier on this dataset. These results are particularly remarkable if we consider that it is really hard to select the best classifier within an ensemble outside of a controlled lab environment. For instance, we can observe that whereas the HMM3 performs poorly against the XSS-SQL attacks, it works very well against attacks in the Generic dataset so that it is really hard to establish what is the best HMM in the ensemble.

5. Conclusions

In this paper we presented *HMMPayl*, an Intrusion Detection System that analyzes the HTTP payload using HMM. We achieved several important results. First, we proposed an IDS that improves the classification accuracy with respect to previously proposed solutions. Second, we proposed a sampling strategy that reduces the computational cost avoiding an unreasonable reduction of the classification accuracy. Third, we demonstrated that an approach based on multiple classifiers is particularly suitable for this application and leads to results significantly better in terms of AUC_p.

References

D. Ariu. *Host and Network based Anomaly Detectors for HTTP Attacks.* PhD thesis, PhD Program in Electronic and Computer Eng. (DRIEI), University of Cagliari, Italy, 2010.

I. Corona, D. Ariu, and G. Giacinto. Hmm-web: A framework for the detection of attacks against web applications. In *Proc. of the IEEE Int. Conf. on Communications*, 2009.

L. Kuncheva. *Combining Pattern Classifiers.* Wiley, 2004.

R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34(4):579 – 595, 2000.

R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Comp. Networks*, 53(6):864 – 881, 2009.

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

K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *Proc. of the 7th Int. Symp. on Recent Advances in Intrusion Detection - RAID*, 2004.