Detection of Web-based Attacks through Markovian Protocol Parsing*

Juan M. Estévez-Tapiador
Dept. of Computer Science
University Carlos III Madrid
jestevez@inf.uc3m.es

Pedro García-Teodoro
Dept. of Signals, Telematics, and Communications
University of Granada
pgteodor@ugr.es

Jesús E. Díaz-Verdejo
Dept. of Signals, Telematics, and Communications
University of Granada
jedv@ugr.es

Abstract

This paper presents a novel approach based on the monitoring of incoming HTTP requests to detect attacks against web servers. The detection is accomplished through a Markovian model whose states and transitions between them are determined from the specification of the HTTP protocol, while the probabilities of the symbols associated to the Markovian source are obtained during a training stage according to a set of attack-free requests for the target server. The experiments carried out show a high detection capability with low false positive rates at reasonable computation requirements.

1. Introduction

The security of the information, in a broad sense, and specifically, the security across Internet-enabled systems is, nowadays, an increasing concern. The undeniable existence of vulnerabilities both in software and hardware makes not only feasible but fairly probable for a system connected to Internet to be attacked by various means. Evidence shows a continuous increase in the number of reported security incidents and in their magnitude [1]. On the other hand, the dependence of human activities on Internet-based systems is also increasing, which aggravates the security risks. In this context, Intrusion Detection Systems (IDS) provide methods and mechanisms to detect the existence of attacks. Ideally, an IDS should report every attack in real time. But current IDSs are far from ideal. They cannot report every attack and, even in this case, they probably can not do it in real time. Furthermore, it is possible to detect normal activity as an attack, which leads to the false positives problem. Although those three aspects (detection accuracy, near real time operation and low false positive rates) are not the only features to evaluate an IDS, they are the most important ones.

Many of the attacks reported nowadays concern application layer traffic, and more specifically web-based services. Thus, a number of proposals have been made in the literature to detect security-related anomalies in the usage of application-layer protocols like HTTP, DNS, or SMTP [2]. For example, Krügel *et al.* show in [3] that the length of a request to a server and its probability density function are good features for evaluating the correctness of a payload. This fact is justified by the very nature of the required service input. For instance, HTTP and DNS payloads are primarily composed of several strings specifying the service requested, coupled with the resource to which the service is to be applied, protocol version, request modifiers, client information, etc. These strings are partially structured according to the protocol syntax rules. In this context, the *protocol analysis* approach tries to derive a model directly from the formal syntax and/or semantics of the protocol that generates the traffic labeled as "normal". The interested reader can find examples of these techniques in [4], [5], [6], and [7].

With this idea in mind, the authors propose a technique aimed at modeling the normal behavior of an application layer protocol, specifically HTTP, and apply it for the purpose of anomaly detection. The introduced model is hybrid in the sense that two sources of information are considered: (a) the HTTP-payloads structure, as defined in the corresponding standards, and (b) the contents of the observed payloads in the monitored environment, i.e. the nature of the resulting IDS is network-based. Both sources of information are merged to construct a Markov model, which is used to evaluate the probability of an observed payload, and subsequently to label it as

* This work has been partially supported by the Spanish Government through MECD (National Program FPU, reference AP2001-3805) and MCYT (Project TIC2002-02798, FEDER funds 70%).

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**

normal or anomalous. As it will be demonstrated, the experimental results show an excellent behavior, in terms of detection accuracy, real time operation and low false positives rate, when compared with other current anomaly-based IDSs, yielding a very low false alarm rate while detecting all the attacks tested.

The rest of the paper is structured as follows. The basis of the modeling technique and its application to HTTP payload modeling are explained in Section 2, thus providing experimental results related to its classification capabilities in Section 3. This same section also explains some concerns related to the application of the proposed system in real environments. Finally, Section 4 summarizes the main conclusions and further extensions of this work.

2. Stochastic protocol parsing for anomaly detection

After the brief introduction presented in the previous section, this point introduces a new approach in order to model normality in HTTP payloads for anomaly detection in web-based traffic. Based on the Markov modeling theory ([8], [9]), the detection scheme is as follows. First, a Markov model is built from two information sources: the specification of the HTTP protocol and the payloads observed during a training phase of the system. The specification is used to determine the states of the model and its topology, while the payloads, once adequately processed, are associated to the observation symbols of the Markov model. Therefore, this approach combines two paradigms: learning and specification. Finally, once the model is constructed, each payload to be evaluated is parsed according to the model, which yields the probability that the payload has been generated by it.

2.1. Structure of HTTP payloads

As in the case of lower layer protocols, one of the main features of the application traffic is that information is highly structured according to the syntax rules provided by the corresponding protocol. In the case of the HTTP protocol, the sensible information related to resource access is contained into the so-called *request line*. To be precise, the name of the addressed resource and the CGI-like queries against scripts in the local server are both into the URL (for details see the appropriate RFCs in [10] and [11]).

The structured nature of HTTP payloads enables a fine 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, an 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 has most of the times the typical form:

```
host.name.domain/directory/my_script?
var1=a+value&var2=value2
```

so that the sequences obtained after the segmentation correspond to server and resource names, directories, variables and values destined to CGI scripts, etc.

This way, it is possible to segment each HTTP payload into a set of strings which are combined according to the syntax rules specified by the protocol. Obviously, the set of possible strings depends on the addressed server and its contents, while the syntax rules are fixed according to the protocol.

On the other hand, each segment can be labeled as belonging to a *type of segment* (e.g. server name, directory, variable, etc.), and the syntax rules specified in terms of the types of the segments and their relative positions in a *correct* payload.

This observation is the key for the proposed approach, which is to be described in the next subsections.

2.2. On the use of Markov models for payload description

Let us consider a first-order Markov model $\lambda = (\Gamma, \Theta, \mathbf{A}, \mathbf{B}, \Pi)$, given by

- a set Γ of N states,
- a set Θ of M symbols,
- the matrix **A** of transition probabilities among the states,
- the matrix **B** of observation probabilities for each of the symbols in each state, and
- the vector Π of initial probabilities.

The Markov model can be viewed as a probabilistic finite state automaton (PFSA) which generates sequences of symbols. The occurrence of the different symbols within a sequence of outputs is governed through a probability distribution which, in turn, depends on the system current state. Therefore, a Markov model can be conceived as two interleaved stochastic processes: transition among states (matrix **A**) and generation of a symbol in each of them (matrix **B**).

Now, let us consider each HTTP payload just as a set of segments (symbols) merged according to a set of rules. It is possible to use a Markov model to produce the payload just by:

1. Describing the set of syntax rules in terms of states and transitions among them.
2. Associating the symbols to be generated in each state with the values of each of the segments in the payload.

For example, consider the following URI:

```
host.name.domain/dir1/dir2/script
```

It can be segmented into the following four tokens:

Θ = {"host.name.domain", "dir1", "dir2", "script"}

We can build a 3 states model, with Γ ={"server", "path", "resource"}, and a matrix of possible transitions among states, as well as symbols that can be observed in states (let us ignore the probabilistic interpretation by now). The resulting model can be expressed through the following tables ("A" stands for allowed, "NA" stands for not allowed):

Transitions (matrix **A**)

from \ to	server	path	target
server	NA	A	A
path	NA	A	A
target	NA	NA	NA

Observations (matrix **B**)

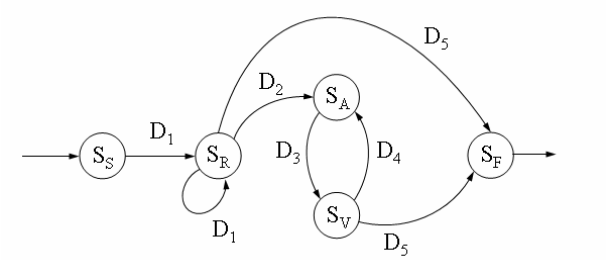
symbol \ state	server	path	target
host.name.domain	A	NA	NA
dir1	NA	A	NA
dir2	NA	A	NA
script	NA	NA	A

As a result, the previous tables encode a basic set of syntax rules that govern the process of production of simple URIs as that shown above. Furthermore, it is possible to provide a probability of production if we consider probabilities of transition and probabilities of observations instead of just "allowed / not allowed" restrictions.

2.3. System architecture

The proposed anomaly detector relies on a Markov model whose topology is shown in Fig. 1(a). The model is composed of a set of states with some transitions allowed among them. The states are:

- S_F: Final state,



(a)

from\to	S_S	S_R	S_A	S_V	S_F	S_{OOS}
S_S	–	D_1	–	–	D_5	D_2, D_3, D_4
S_R	–	D_1	D_2	–	D_5	D_3, D_4
S_A	–	–	D_3	–	–	D_1, D_2, D_4, D_5
S_V	–	D_4	–	–	–	D_1, D_2, D_3
S_F	–	–	–	–	–	–

(b)

Figure 1. Markov model for HTTP: (a) topology, and (b) matrix of transitions.

- S_S: Server state, associated to the name of the addressed server,
- S_R: Resource state, associated to the requested resource,
- S_A: Attribute state, for the names of attributes or variables in the URI,
- S_V: Value state, for the values of the attributes, and
- S_{OOS}: Out of specification, for purposes related to a violation of the protocol specification.

These states and the possible transitions among them are deduced directly from the specifications of the HTTP protocol. Transitions among states are triggered by a set **D** of delimiters found in the URI which, on the other hand, are used to segment the payload into symbols associated to each state. As previously stated, and according to the HTTP protocol specification, the delimiters are:

- D_1="/", resource delimiter,
- D_2="?", parameters delimiter,
- D_3="=", attribute assignment delimiter,
- D_4="&", inter-parameter delimiter, and
- D_5=blank (ASCII 32), end of resource (EOR) delimiter.

By merging states and delimiters, it is possible to specify a matrix of transitions, in terms of the next state according to the previous one and the observed delimiter, as illustrated in the matrix representation provided in Fig. 1(b).

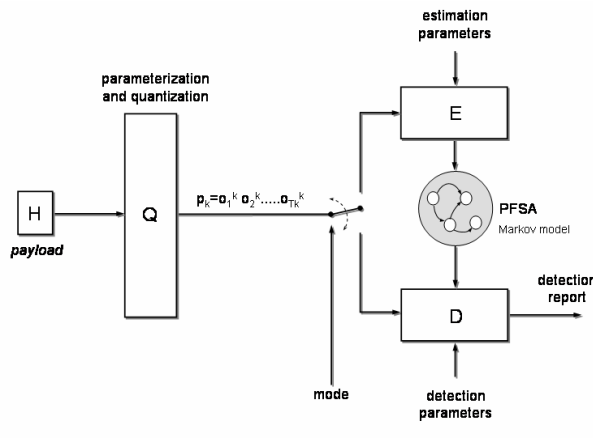


Figure 2. General detection architecture for Markovian protocol parsing.

The set of symbols, Θ, that can be observed in each state and their respective probabilities are obtained during a training stage in which the traffic destined to the server/s to be monitored is captured, segmented and analyzed. During this phase, both the probabilities of transitions, **A**, and the probabilities of observation of each symbol in each state, **B**, are estimated by using adequate methods.

The use of the Markov model to detect anomalies is based on assigning each payload, *p*, a *normality score*, $N_s(p)$, according to the probability of generation of the payload by the model λ:

$$N_s(p) = P(p \mid \lambda) \tag{1}$$

The payload, *p*, will be classified as anomalous if the normality score is lower than a threshold, θ, or the state S_{OOS} is reached, which means that a transition not allowed by the specification has appeared. In order to simplify the reasoning, we can assume that the normality score vanishes when the system enters the state S_{OOS}. Therefore:

$$class(p) = \begin{cases} normal & if \quad N_s(p) > \theta \\ anomalous & if \quad N_s(p) \le \theta \end{cases} \tag{2}$$

The application of the proposed modeling technique suffers from the well known problem of insufficient training, also called *out of vocabulary* problem. Put simply, this problem is related to the appearance of a symbol that has never been observed during training while evaluating a sequence. The usual way of handling this situation is to use a smoothing factor, ε, as the probability of any unobserved symbol.

Fig. 2 summarizes graphically the operation of the proposed technique. The payload is initially introduced in a parameterization and quantization stage wherein the URI is extracted and segmented according to the set of delimiters. In estimation mode two tasks are accomplished. First, a table of symbols, or *dictionary*, is constructed. This data structure contains an entry for each different string (token) observed between two delimiters, in such a way that a unique symbol is associated with it. This quantization dictionary is used to transform the URI into a sequence of symbols, which are manipulated by the training algorithms in order to estimate the main parameters of the Markov model (basically, matrices **A** and **B**). Finally, in detection mode each HTTP request is passed through the Markov model and a normality score is obtained. If the payload receives a low score, an alarm is triggered and intrusion response mechanisms can be launched.

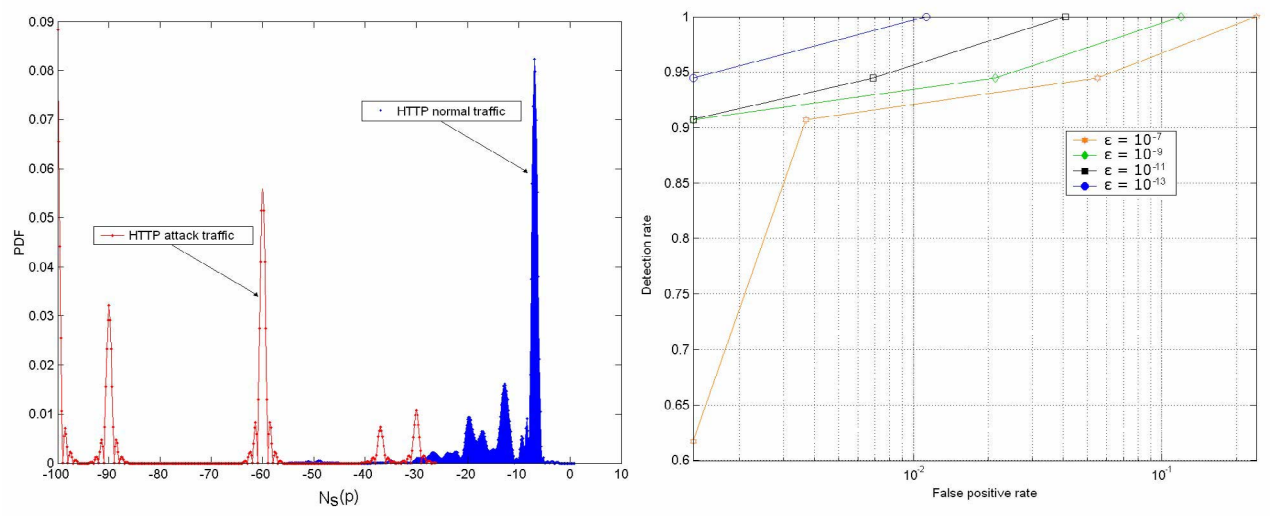
3. Evaluation and results

Some experiments have been carried out in order to evaluate the detection capabilities of the proposed system. The evaluation framework and the results obtained are discussed in this section.

3.1. Test-bed of HTTP traffic

A key aspect of any evaluation process is the dataset to be used [12]. In this work, the DARPA 1999 IDS Evaluation Program [13] has been considered as the main source of experimental data. Despite it presents some unquestioned drawbacks -see [14] for a critique about this dataset-, it is undeniable that this is the only publicly available facility for the evaluation of IDSs. The basic framework is composed of several off-line test sets, each consisting of traffic captured during 5 weeks in a specially designed network environment. The data intended for training purposes consist of the first 3 weeks, while the remaining 2 weeks are devoted to test.

The evaluation of our system requires two complete data sets of both normal traffic and anomalous or attack traffic. The normal traffic has been collected from the above mentioned datasets by considering weeks 1 and 3, which are attack-free. For this purpose, we have extracted packets destined to two different servers: *hume* (IP address 172.16.112.100) and *marx* (IP address 172.16.114.50). The total amount of HTTP requests extracted is 12154 for *hume* and 16539 for *marx*, although there is some redundancy in terms, for example, of various instances of the same payload



(a) (b)

Figure 3. Experimental results: (a) distribution of the normality scores for normal and attack HTTP traffic for $\varepsilon = 10^{-13}$; (b) ROC curves associated to the detector for different configurations.

from different sources. This dataset has been randomly split into two: 70% of the payloads for training the model, and the remaining 30% for testing purposes.

The attack traffic has been collected in a different way, since the number and nature of attack traffic in DARPA datasets are not adequate. For this purpose, we have collected some well-known vulnerabilities in HTTP service listed in arachNIDS database [15], obtaining a total of 86 different HTTP exploits. By using some programs that implement those exploits, a total amount of up to 1500 attack payloads have been generated and particularized to the DARPA framework. Afterwards, they have been captured and recorded in the same way that was done for normal traffic.

3.2. Experimental results

The detection capabilities of the system can be adjusted by tuning two parameters: the smoothing factor, ε, and the detection threshold, θ. The smoothing factor is assigned during the training of the system, since the observation probabilities will be dependent on it. On the other hand, the detection threshold can be tuned once the training is completed.

The system has been trained using the aforementioned traffic records for different values of the parameter ε. The evaluation of the payloads in the test and attack sets provides values of the normality score in the ranges observed in Fig 3(a). It is important to note that if there is no overlapping among ranges, the system would be able to detect all the attacks with no false positives. There exists some overlapping among both kinds of traffic, which depends on the smoothing factor used. For $\varepsilon = 10^{-13}$ the number of payloads in the conflicting zone is low, which, in turn, implies that it should be possible to correctly classify all the attacks with a low false positive rate.

The detection capabilities can be measured by computing the ROC curves (detection probability vs. false positives probability) associated with the system. Those curves can be obtained by varying the detection threshold, θ, which, in terms of Fig. 3(a), determines the point selected to split the two distributions. The results are depicted in Fig. 3(b). As expected, the best results correspond to using a smoothing factor $\varepsilon = 10^{-13}$. According to the obtained results, it is possible to reach a 100% of detection with a false positive rate of around 1%.

3.3. System implementation and performance

In a real time configuration, service requests should be captured and processed online before they reach the server. This restriction can make the detector a bottleneck unless it will be able to process payloads efficiently.

As exposed before, detection is carried out by passing the quantized payload through a Markov model, which theoretically requires O(n) operations. A software prototype of the proposed system has been implemented in C and tested in an Intel Pentium 4

platform at 2.4 GHz with 1 GB RAM. Once constructed the model, the time required to carry out the detection, including the initial quantization stage and the application of the threshold, is 0.0119 ms per payload in average. Taking its inverse, this value leads to a system able to process around 84000 requests/s.

4. Conclusions and future work

A novel paradigm for anomaly detection in network traffic at the application layer has been presented in this paper. The approach is based on the specification of a Markov model to represent the normality of the payloads destined to a given server. For that, the protocol description of the corresponding service is considered to define the states as well as the matrix of transition probabilities among them that compose the model. On the other hand, the output symbols and their observation probabilities in each state are derived from the analysis of captured network traffic during a training stage. The experimental results obtained when applied to the HTTP service, clearly show the high detection performance achieved by the proposed scheme.

Due to space restrictions, a number of relevant questions concerning system operation have been deliberately omitted in this paper. These are mainly related with practical concerns, such as: (1) how often must the system be re-trained? and (2) how can be appropriately tuned the system parameters (θ and ε)?

Other significant user-related service protocols are SMTP and DNS. Since both of them present a well defined syntax and semantic structure, we firmly believe that the application of the approach introduced in this work for the purpose of anomaly detection in SMTP and DNS traffic will result in significant advances in the IDS field. On the other hand, as discussed in Section 4, the authors consider a unavoidable challenge to implement and prove our IDS scheme in a real network environment. Both issues will be tackled in future work.

References

[1] CERT/CC Statistics 1988-2003. Available online at: http://www.cert.org/stats. June 2004.

[2] J.M. Estévez-Tapiador, P. García-Teodoro, J.E. Díaz-Verdejo, "Anomaly Detection Methods in Wired Networks: A Survey and Taxonomy", in *Computer Communications*, 27(16):1569-1584, 2004.

[3] C. Krügel, T. Toth, and E. Kirda:, "Service Specific Anomaly Detection for Network Intrusion Detection", in *Proc. 17th ACM Symp. on Applied Computing*. Madrid, Spain, 2002, pp. 201-208.

[4] J.M. Estévez-Tapiador, P. García-Teodoro, J.E. Díaz-Verdejo, "Measuring Normality in HTTP Traffic for Anomaly-Based Intrusion Detection", in. *Computer Networks*, 45(2):145-193, 2004.

[5] C. Krügel and G. Vigna, "Anomaly Detection of Web-Based Attacks", in *Proc. 10th ACM Conference on Computer and Communications Security*. Washington, D.C. (USA), 2003, pp. 251-261.

[6] M.V. Mahoney and P.K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection", in Lecture Notes in Computer Science, Vol. 2820 (*RAID'03*), pp. 220-237. Springer-Verlag, 2003.

[7] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, "Specification-Based Anomaly Detection: A New Approach for Detecting Network Intrusions", in *Proc. 9th ACM Conference on Computer and Communications Security*, Washington, D.C. (USA), 2002, pp. 265-274.

[8] J.L. Doob. *Stochastic Processes*. John Wiley & Sons, 1953.

[9] W. Feller. *An Introduction to Probability Theory and its Applications. Vol. I, 3rd Edition*. John Wiley & Sons, 1968.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2068. January, 1997.

[11] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers". RFC 2396. August, 1998.

[12] N. Athanasiades, R. Abler, J. Levine, H. Owen, and G. Riley, "Intrusion Detection Testing and Benchmarking Methodologies", in *Proc. 1st IEEE International Workshop on Information Assurance*, Darmstadt, Germany, March 2003, pp. 63-72.

[13] R.P. Lippmann, J.W. Haines, D.J. Fried, J. Corba, and K. Das, "The 1999 DARPA Off-line Intrusion Detection Evaluation", in *Computer Networks*, 34(4):579-595, 2000.

[14] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory", in *ACM Transactions on Information and System Security*, 3(4):262-294, November 2000.

[15] arachNIDS: Advanced Reference Archive of Current Heuristics for Network Intrusion Detection Systems. Available online at: http://www.whitehats.com/ids. March 2004.