Stochastic Protocol Modeling for Anomaly Based Network Intrusion Detection†

Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo
Department of Electronics and Computer Technology
University of Granada – Spain
E-mail: {tapiador, pgteodor, jedv}@ugr.es

Abstract

A new method for detecting anomalies in the usage of protocols in computer networks is presented in this work. The proposed methodology is applied to TCP and disposed in two steps. First, a quantization of the TCP header space is accomplished, so that a unique symbol is associated with each TCP segment. TCP-based network traffic is thus captured, quantized and represented by a sequence of symbols. The second step in our approach is the modeling of these sequences by means of a Markov chain. The analysis of the model obtained for diverse TCP sources reveals that it captures adequately the essence of the protocol dynamics. Once the model is built it is possible to use it as a representation of the normal usage of the protocol, so that deviations from the behavior provided by the model can be considered as a sign of protocol misusage.

1. Introduction

Research in Intrusion Detection Systems (henceforth referred to as IDS) has been an active field during the last twenty years. Nevertheless, current detection technology still suffers performance limitations referring to its high false alarm probability, low detection accuracy and high load of monitoring and computing overhead. Traditionally there have been two main approaches to the problem of intrusion detection: misuse detection and anomaly detection. In *misuse detection*, each known attack is modeled through the construction of a signature. Incoming activities that match a pattern in the library of attack signatures raise an alarm. The percentage of false alarms depends on whether the matching algorithm allows only exact signature matching or some kind of deviation. In *anomaly detection* the main objective is to model normal profiles of the system, so that substantial deviations from this behavior can be labeled as intrusive or, at least, as suspicious. Statistical techniques are surely the most used tools for the construction of normal activity patterns. Interested readers can find good surveys about IDS in [1] and [2].

Regardless of the method used for detecting attacks, an IDS can be alternatively classified as host based or network based depending on its source of input data. A host based IDS tries to identify intrusions analyzing activities at hosts, mainly users and programs. For example, Denning [3] proposed a scheme in which patterns related to login times and resources consumed by users and programs were constructed. On the contrary, network based IDS do not focus on activities on hosts but on the traffic that is transported over the network [4]. Examples of network based IDS are Snort [5] and Bro [6].

The need to define the normal state of a monitored system is a crucial question for any anomaly based IDS. Several authors agree and point out that probably the most important challenge for these methods is the choosing of features to be modeled [7], [8]. Such a features must characterize with precision the service, system or network usage patterns, in order to obtain an accurate model of the normal behavior of the object. But at the same time, they must have enough discriminant capacity to perform a correct separation of intrusive and non-intrusive activities. Measuring system normality turns thus into one of the most important points concerning the performance improvement in current detection systems.

In the case of host based IDS, several works have shown that the sequences of system calls executed by a program are excellent features for modeling the normal behavior [7], [9]. Once that an application is "sampled" by means of an ordered set of the system calls that it has executed, it is possible to extract some kind of statistical properties with the aim of modeling its behavior. Markov

† This work has been partially supported by Spanish MECD under National Program PNFPU (reference AP2001-3805) and Spanish MCYT under project TIC2002-02798 (FEDER funds 70%).

chains, rule learning systems and other approaches have been used for this purpose (e.g., see [10], [11]).

In the context of network based IDS, it has been argued that several features associated with traffic modeling, like volume of traffic in the network or statistics of the operation of application protocols, are particularly suited for detecting general novel attacks [12], [13]. Another proposed approaches define the normal state of the network by means of a finite automaton, obtaining thus that each sequence of normal actions can be expressed by allowed transitions between states [8], [14]. Some of these proposals are signature based approaches, and state machines are used as a framework for the construction of attack patterns.

In this work we present a special case of anomaly based method for detecting protocol misusages in computer networks. A protocol anomaly detector is designed to monitor a given protocol looking for deviations from its normal usage. Justification for this approach comes from the fact that a large amount of network attacks are founded on diverse protocol usages that fall out of the official protocol description. Building such a detector requires an analysis of the specific protocol implementation existing across the network.

The approach taken in our work is inspired in that used in host based IDS. The basic idea is to define a set of features for a given protocol in such a way that they can be conceived as the equivalent of the system calls executed by the applications (i.e., as a signature of its operation). These features are subsequently used for characterizing network traffic that utilizes the protocol. The "normal" protocol usage is then modeled by means of a Markov chain, using these sequences of observations as inputs. Likewise, in this contribution we propose the use of a specific measure, called *MAP*, for evaluation purposes

The rest of this paper is organized as follows. Section 2 introduces a brief background on Markov chains and their use for sequence recognition. We describe in detail our approach to protocol modeling in Section 3, specifically its application to TCP. Section 4 provides further discussion concerning the proposed scheme and the results obtained. Finally, Section 5 summarises the paper by presenting our main conclusions, the benefits of the work developed and future research objectives.

2. A brief background on Markov chains

2.1. Foundations

Let us suppose a system which evolves through numbered states in accordance with probabilistic laws satisfying the Markov hypothesis (i.e., the state at time $t+1$ only depends on the state at time t). Each state of the set of possible states $\mathbf{\Gamma}=\{S_1, S_2, ..., S_N\}$ represents a different and specific situation in which the system can be.

Let the variable that represents the current state at time t be q_t. Then, if $\mathbf{P}[q_t=i] > 0$, define a_{ij} by

$$a_{ij} = P[q_{t+1} = j \mid q_t = i] = \frac{P[q_t = i, q_{t+1} = j]}{P[q_t = i]} \qquad (1)$$

and let $\boldsymbol{A}$ be the matrix $[a_{ij}]$. Then, if $\mathbf{P}[q_t=i] > 0$,

$$a_{ij} \geq 0, \qquad \sum_j a_{ij} = 1 \qquad (2)$$

Thus the matrix of probabilities of transitions $\boldsymbol{A}=[a_{ij}]$ represents the probability of being in the state i at some time t, and reach the state j at time $t+1$. According to the previous definitions any matrix $\boldsymbol{A}=[a_{ij}]$ satisfying (2) can be used, together with initial probabilities $\mathbf{\Pi}=\{\pi_i\}$, so that $\pi_i=\mathbf{P}[q_1=i]$, satisfying

$$\pi_i \geq 0, \qquad \sum_i \pi_i = 1 \qquad (3)$$

to define a Markov chain with stationary transition probabilities. The probability $p_j^{(n)}$ of state j at time n is given recursively by

$$\begin{aligned} p_j^{(1)} &= \pi_j \\ p_j^{(n)} &= \sum_i p_i^{(n-1)} a_{ij}, \qquad n > 1 \end{aligned} \qquad (4)$$

Good introductory texts about Markov chains are [15] and [16], and interested readers can found there more detailed information.

2.2. Parameter estimation in Markov chains

In this discussion we suppose that the knowledge concerning different states reached by the system is acquired through the observation of the system outcomes. These outcomes are elements from a finite set $\mathbf{\Theta}=\{O_i\}$, so that the possible outcomes O_i are referred to as possible states of the system.

Let us suppose that a set of system observations O_1, O_2, ..., O_T, is given. In the theory of Markov chains we consider the simplest generalization which consists in permitting the outcome of any trial to depend on the outcome of the directly preceding trial (and only on it) [15]. Thus the matrix of probabilities of transitions can be estimated by:

$$a_{ij} = \frac{P[q_t = O_j, q_{t-1} = O_i]}{P[q_{t-1} = O_i]} \quad (5)$$

Both terms of the previous expression can be calculated by means of a simple process of counting occurrences into the sequence of observations. On the other hand, initial probabilities vector $\boldsymbol{\Pi}$ can be estimated in a similar way if a set of outcome sequences is available. Thus initial probability of each symbol can be computed by simply counting the number of times the corresponding symbol appears at the beginning of the sequences.

2.3. Sequence recognition with Markov chains

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

$$MAP(O, \lambda) = \pi_{O_1} \cdot \prod_{t=1}^{T-1} a_{O_t O_{t+1}} \quad (6)$$

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

$$LogMAP(O, \lambda) = \log(\pi_{O_1}) + \sum_{i=1}^{T-1} \log(a_{O_i O_{i+1}}) \quad (7)$$

The use of accumulated probabilities presents the inconvenient that no one probability can be zero. This is usually solved by means of a previous *smoothing* of the model. Although several methods exist for this purpose, probably the simplest smoothing technique consists in setting those probabilities lower than a given threshold to a fixed value "ε".

3. TCP Modeling with Markov chains

3.1. Parameterization and quantization

Information concerning signaling and dynamics in network protocols is located at PDU (*Protocol Data Unit*) headers. Thus, it might be expected that useful variables for modeling the "normal" protocol behavior will be the values of header fields or some combination of them. Our basic approach consists in obtaining a representation of the network traffic at a given layer (i.e., the modeling of the corresponding protocol) as a sequence of scalar observations.

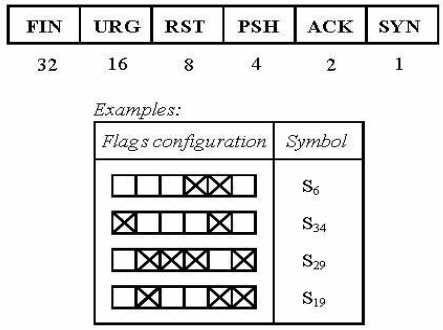

Figure 1. Illustration of the TCP quantization process. Flags are considered as a binary number n of 6 bits, so that S_n is the symbol associated with the TCP segment.

Once this transformation is achieved, the next step will be the modeling of such a sequence. For this purpose it is necessary to carry out a quantization stage of the protocol headers. In the case of TCP, most of the information related to the signaling is located in the fields known as flags [17]. A simplistic but effective approach is to consider the flags configuration of each TCP segment as its signature. Thus, it is possible to associate a unique symbol S_p with each segment:

$$S_p = \sum_{i=1}^{6} w_i \cdot b_i = syn + 2 \cdot ack + 4 \cdot psh + 8 \cdot rst + 16 \cdot urg + 32 \cdot fin \quad (8)$$

The idea behind this simple quantization scheme is illustrated in Figure 1. Flags are retrieved from each segment and disposed in the order shown in the cited figure. The symbol associated with the segment is obtained according to expression (8), i.e., considering the flags configuration as a binary number. We obtain thus a 64-valued quantization dictionary, in which each element represents a different configuration of flags.

According to the protocol specification [17] not all of these configurations are valid. For example, a TCP segment with SYN and RST flags simultaneously set to 1 is not coherent with the correct protocol usage and, hence, can be considered as a protocol misuse. Most of these protocol misuses are basic tools for information gathering processes like port scanning. Current techniques used in NIDS to detect this kind of attacks are signature-based, so

Table 1. Data sets of normal traffic used for the construction of a TCP model. The size of each trace indicates the number of recorded TCP headers.

Service FTP			Service HTTP			Service SSH		
Trace	No. of sessions	Total Size	Trace	No. of sessions	Total Size	Trace	No. of sessions	Total Size
ftp.1	14	5207	http.1	29	8975	ssh.1	11	3349
ftp.2	9	3762	http.2	41	13862	ssh.2	9	3294
ftp.3	18	6862	http.3	102	28107	ssh.3	12	3766
ftp.4	32	18101	http.4	57	19343	ssh.4	24	7069
ftp.5	69	27753	http.5	98	50462	ssh.5	143	63252
ftp.6	78	51345	http.6	62	21310	ssh.6	218	122355
ftp.7	156	133615	http.7	117	41329	ssh.7	241	151142

that a pattern representing the attack is constructed. Subsequently, some kind of pattern matching algorithm is used to find evidences of any known attack in the incoming network traffic. Surely the most limiting characteristic of this approach is the impossibility of recognizing those attacks that have not previously been typified by means of a signature.

3.2. Data sets

As a first approach we have used incoming TCP traffic filtered by destination port (i.e., by application or service) as training sequences. Applications monitored for our experiments have been SSH, HTTP, and FTP, so that several connections have been recorded for each one of them.

Table 1 shows some characteristics of the traffic files used. Such a traffic has been obtained monitoring normal, incoming connections to a single host running an FTP server, an SSH server, and an HTTP server in our laboratory. The capture, filtering and extraction of the TCP headers can be easily made with *tcpdump* [18] or any similar tool. Each file contains several non-interleaved sessions. To be precise, each session is a sequence of ordered TCP headers which will be transformed into a sequence of symbols according to the quantization process. For example, Figure 2 shows a portion of a SSH file with two complete sessions (each session always has the symbol S_1 as starting value).

3.3. Model estimation

Figure 3 graphically illustrates the estimation process for the model. TCP headers collected in the data sets are quantized so that each session is represented as an ordered sequence of symbols like that shown in Figure 2. These traces are then used as inputs for the estimation algorithm briefly described in section 2.2.

Results provided after this process concern the matrix of transition probabilities and the vector of initial probabilities. This task is achieved separately with the traces corresponding to each application. The obtained models are shown in the Figure 4.

For example, the model obtained with sequences from FTP traffic presents four states with non-null probability of transition: S_1, S_2, S_6, and S_{34} (see Figure 4). State S_1 corresponds to a TCP segment with SYN flag set to 1, and represents the request for the establishment of a connection. States S_2 and S_6 are conceptually identical and represent the acknowledgment of a received packet. Nevertheless, while S_2 only has ACK flag set to 1, state S_6 corresponds to a segment with ACK and PSH flags set to 1. This difference could be originated by different states of network load, so that certain packets are labeled with PSH flag for their immediate delivery. Finally, state S_{34} corresponds to a packet with FIN and ACK flags set to 1. It represents an acknowledgment of a previous packet and simultaneously the closing of the connection.

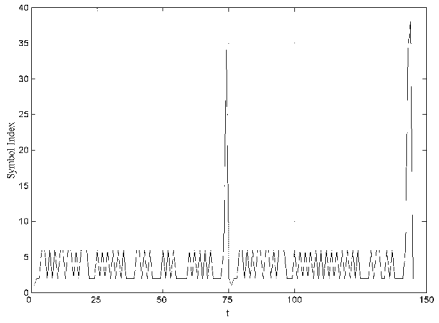

Figure 2. Sequence of symbols corresponding to two short SSH sessions. The first session starts at time t=1 and finishes at time t=75, while the second one starts at time t=76 and finishes at time t=148.

The analysis of the transitions obtained for this model reveals that it has captured the correct dynamics specified for the protocol TCP [17]. More specifically, this model is a subset of the well known TCP state machine.

The previous discussion is likewise applicable to the models obtained for HTTP and SSH services. Although they are essentially equivalent, the observed differences – like the apparition of states with flags RST– are originated by the usage that the particular application makes of the protocol. Anyway, it is possible to identify the same semantics corresponding to the protocol utilization in these models.

3.4. Testing the model

After the training period a Markov chain is available for the incoming TCP traffic from each specific application. These models can be evaluated according to expressions (6) and (7), obtaining thus performance measures related to their discriminative power between correct and wrong TCP usage.

The testing procedure is as follows. Incoming traffic is filtered according to its destination port (i.e., the receiver application). Each packet in the flow is then processed by extracting its TCP header and quantized according to expression (8). The obtained sequence of symbols is then passed through the model and evaluated. Figure 5 shows examples of outputs produced by the corresponding model during two HTTP sessions.

A smoothed model has been used during the evaluation period in order to solve the problem of null probabilities. The implemented method was that briefly described in Section 2.3. Those probabilities which are lower than a given threshold $\varepsilon=10^{-6}$ were setting to the value of ε.

The output shown in the upper graph in Figure 5 corresponds to a "normal" session. The function *LogMAP* for this kind of traffic has always a shape similar to that shown in the figure. While incoming symbols correspond well with those expected by the model, the respective probabilities of transition between them are adequate and, thus, the accumulated sum given by the *logMAP* has no abrupt changes of slope.

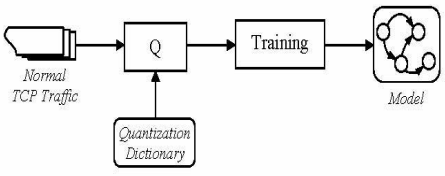

Figure 3. Graphical illustration of the Markov chain estimation process.

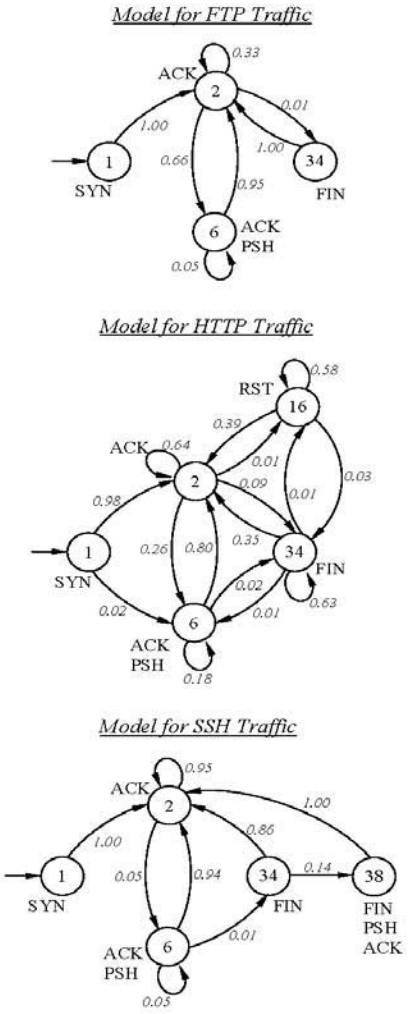

Figure 4. Estimated models for different services over TCP. The values of the transition probabilities between states are also shown. Each transition is defined by the current state S_i and the next state S_{i+1}. Transitions not shown in the table are zero.

On the contrary, the appearance of any pattern of non-expected symbols produces a burst of consecutive low probabilities. This phenomenon can be easily observed by an abrupt change in the slope of the output, like those shown in the lower graph in Figure 5.

A useful method for detecting these changes and, hence, the presence of anomalous traffic is to control when the derivative of *logMAP* is higher than a fixed threshold. We have used for that purpose the family of functions:

$$D_{W_m}(t) = \left| LogMAP(t) - \frac{1}{W_m} \sum_{i=1}^{W_m} LogMAP(t-i) \right| \qquad (9)$$

for values of the parameter W_m = 1, 2, 3, ... Note that the second term in (9) is the mean of the last W_m outputs. Figure 6 shows the effect of this parameter in the response produced by the detector. An increment of its value induces an amplification in the output. Note that the smoothing parameter ε plays an equivalent although inverse role: small values of ε will produce more abrupt changes of slope.

Data sets of anomalous traffic used during the test period have been obtained using tools that exploit several TCP weakness and ambiguities for different purposes. For example, *nmap* [19] and other scanning tools utilize certain TCP segments like the followings in order to achieving their objectives:

- *Null scan*, in which no one flag is activated.
- *Xmas scan*, in which all the flags are set to 1.
- *Stealth FIN*, in which a segment with the flag FIN activated, is sent against a port without a previous established connection.

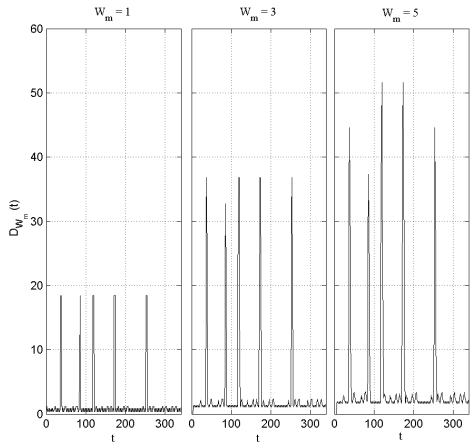

Figure 6. Effect of the parameter W_m in the response produced by the detector. Higher values produce an amplification of the output.

These and other techniques are well known and appropriate filters could be written and installed on a signature based IDS for their detection. However, it is

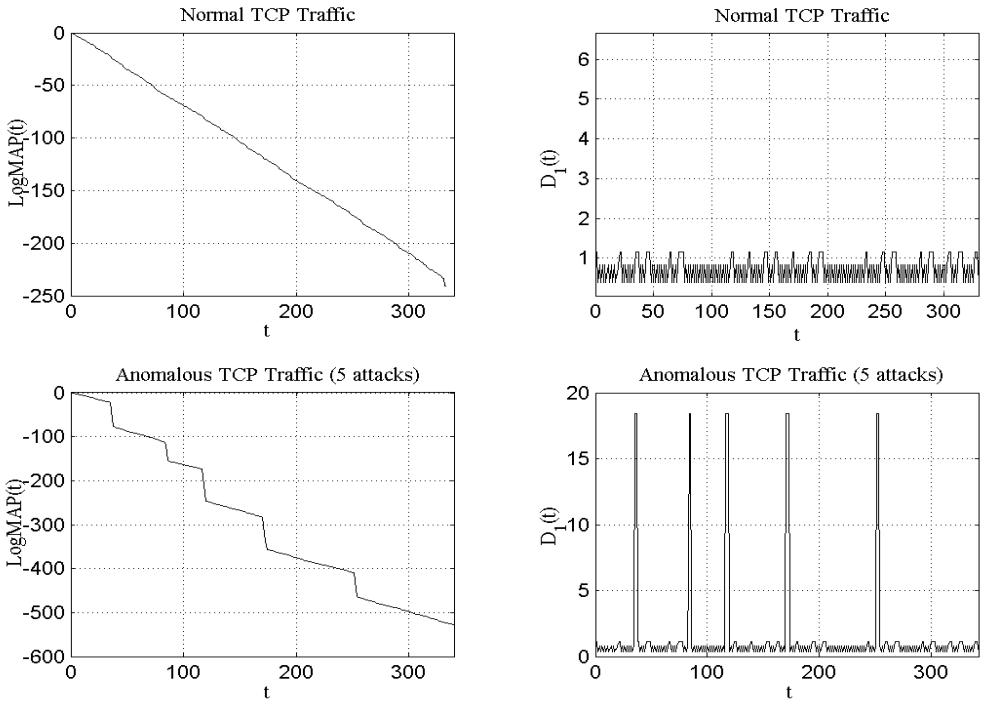

Figure 5. Comparative output graphs produced by the HTTP chain with normal and anomalous TCP traffic corresponding to two sessions. In the lower graph, attacks are located at time t=37, t=85, t=118, t=172, and t=235.

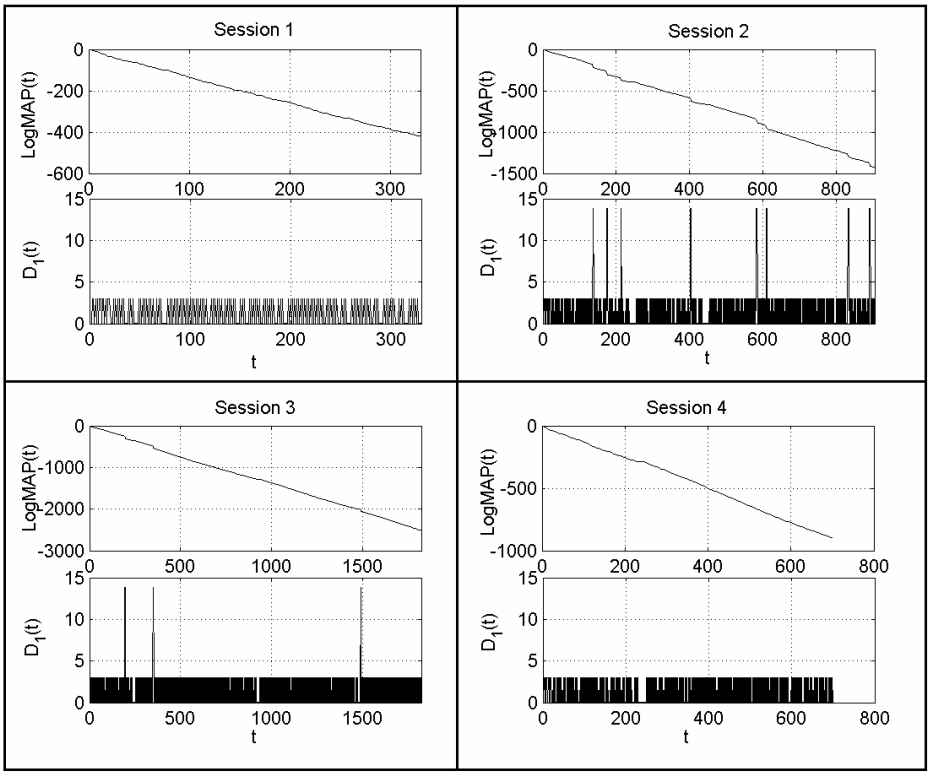

Figure 7. Output produced by the detector during the monitoring of four consecutive SSH sessions. Sessions 2 and 3 contains several attacks, while sessions 1 and 4 are correct. It is clearly shown how the detector has adequately captured the protocol misuses.

obvious that detection capabilities will be given by the library of attack signatures available and, hence, new attacks require new signatures. On the contrary, the use of anomaly detectors implies that not only well known misusages will be detected but too those not exploited yet.

Figure 7 shows the results of monitoring four consecutive SSH sessions. While sessions 1 and 4 do not contain any malicious traffic, sessions 2 and 3 includes several forms of misusages. The graphs illustrate how the detectors correctly capture these anomalies.

4. Discussion

According to the methodology that has been exposed in the previous section, results obtained after the training procedure are a set of individual models: one for each service. To be precise each one of these models contains the "correct" (but specific) usage that a given service makes of the protocol. The deployment of detectors based in this scheme would be as it was previously described: each isolated model monitors incoming traffic whose target is the corresponding application.

Although this approach presents several benefits, its main disadvantage is exactly this specialization property, regardless of performance considerations. It is thus possible that a given service makes use only of a certain subset of the correct protocol usage. The presence of activities that fall into the correct, formal protocol specification but that have not been previously seen by the model raise the alarm. This limitation is inherent to the definition of anomaly based detector: every anomalous event is suspicious.

However, it is reasonable to conceive a unique model for the usage of the protocol (TCP in this case), regardless of the application that utilizes it. In other words, an interesting objective to be tackled is obtaining a model for the usage that the *entire network system* makes of the protocol. Such a model can be easily built within the same previous procedure, but using all the training data without consideration about the destination port.

It is obviously expected that the obtained model with this new approach will be a unification of those individual

chains shown in Figure 4. Although the set of reachable states for such a model is the effective union of states contained in the isolated models, transitions between them can be substantially different. Hence, it is needed to compute them again within the new framework. Likewise, it is accepted an eventual loss of detection accuracy due to the smaller specialization of the complete protocol model.

Figure 8 shows the global TCP chain obtained after the training process using all the data sets described in Table 1. As it was expected, the model for the entire TCP usage is composed by all the states present in the individual chains. On the other hand, new transition probabilities between them can be seen as a "weighted mixture" of the previous ones. It is possible to illustrate this fact with a simple example. Let us consider transition from state S_2 to state S_6. The probability of this transition is 0.66 in the case of the FTP chain, 0.26 in the case of the HTTP chain, and 0.05 for the SSH case (see Figure 4). The corresponding probability value for this transition in the global model is 0.11. Similar comparatives ca be established for the rest of transition probabilities.

Figure 9 shows experimental intrusion detection results for this new model. In this case the evaluation has been made with an smoothing value $\varepsilon=10^{-9}$. It is clearly observed how the model detects protocol misuses similarly it was done by the application-dependant models. However, it is important to comment an important

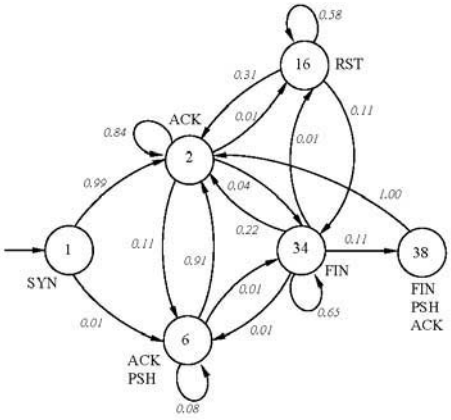

Figure 8. TCP chain obtained with differents sources. Note how the entire model can be seen as an average of the previous, individual chains.

fact. Comparing Figures 5 and 9 it is clearly shown that the output ranges provided by the sequences evaluation have changed. The specific HTTP chain produces values lower than 1.5 for normal traffic and upper than 17 for anomalous traffic. Evaluation of the same traffic with the new model provides an output lower than 6 for normal

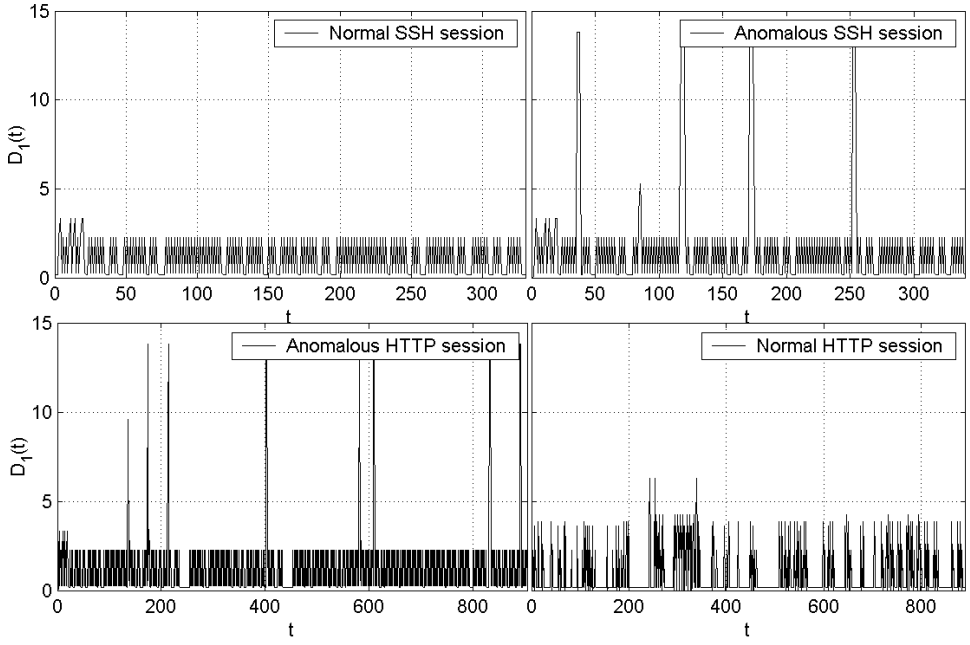

Figure 9. Output produced by the global TCP detector during the monitoring of two SSH sessions and two HTTP sessions. Although the detection accuracy has not decreased, it may be observed how the output ranges have changed.

traffic and upper than 9 for anomalous usages.

This phenomenon is directly related to the loss of specialization of the general model that has been previously discussed. Nevertheless, the detection accuracy can be controlled through the smoothing parameter ε as well as the *Wm*. For example, in the case of application-dependant chains the experiments reveal that a value of $\varepsilon=10^{-6}$ is enough for a good discrimination. However, for the case of the global TCP chain, a value of $\varepsilon=10^{-9}$ or lower is necessary for an accurate separation of correct and wrong TCP usages.

5. Conclusions and future work

In this paper we have presented preliminary results of a new approach for the detection of anomalies in the usage of network protocols. The previously described method, applied to TCP, has demonstrated to be effective in all our experiments.

Besides the modeling scheme proposed, another important contribution is the use of the measure *MAP* and its logarithm for testing purposes. This procedure has been widely used in other applications (e.g., speech recognition) where Markov chains are appropriate solutions for sequence recognition. The "continuous" output given by this function can be easily interpreted as a measure of the probability of recognition of the input sequence. Moreover, derivative of the *logMAP* is an excellent candidate for the construction of anomaly detectors. A simple method based on a threshold can be applied to the response provided by *logMAP*. Differences between outputs of normal and anomalous traffic can be controlled by parameters W_m and ε, facilitating thus the adjustment of the detectors.

In the case of TCP, we have shown that the results obtained are similar to those that could be derived from a model directly built from the formal specification of the protocol. Nevertheless, this way of actuating is not always feasible for several reasons. First, although an specification of each protocol exists, it uses to be ambiguous and, hence, very reliant on the implementation. For example, it is well known that different operating systems have protocol stacks with different behaviors in some circumstances. In this context, a model of the protocol usage derived directly from its use in the environment is more appropriate. Furthermore, there are protocols that do not have something similar to the TCP state machine. For these protocols it is useful to build a model, not only from its general use, but from the specific utilization that the network applications are making of it. This last fact is a crucial point for any anomaly based network intrusion detection.

The deployment of sensors based on the proposed protocol modeling must not be conceived as a complete solution for detection purposes. On the contrary, it is strongly recommended its use in conjunction with other anomaly detection techniques as well as signature methods. It must be considered that attacks based on protocol misusage are only a piece of the current attack technology.

We firmly believe that a layered approach can be used for the detection of anomalous usages of network protocols. Future work will study the application of this methodology to other protocols. The modeling of application level protocols (e.g., HTTP or DNS) for the detection of abnormal uses and intrusion attempts is especially attractive and will be nextly tackled. A previous theoretical and empirical study of the protocol is required for the completion of this objective in order to obtain those significant features that contain important information concerning its use. Moreover, once that the protocol usage is represented as sequences of observations, other modeling techniques will be studied and evaluated.

Likewise, monitoring of self, outcoming traffic points out as an interesting research topic. Correlation of incoming and outcoming traffic models could provide better results than those obtained by only monitoring incoming activities.

References

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

[2] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy." Available: http://citeseer.nj.nec.com/axelsson00intrusion.html, 2000.

[3] D. Denning, "An intrusion-detection model," in *IEEE Transactions on Software Engineering*, vol.SE-13, No.2, pp. 222-232, February 1987.

[4] B. Mukherjee, L. T.Heberlein and K. N. Levitt, "Network Intrusion Detection", *IEEE Network*, Vol. 8, No. 3, May/June, pp. 26-41, 1994.

[5] M. Roesch, "Snort – lightweight intrusion detection for networks," in *Proceedings of the 1999 USENIX LISA conference*, November 1999.

[6] V. Paxon, "Bro: A System for detecting network intruders in real-time," in *Proceedings of the 7th USENIX Security Symposium*, San Antonio, Texas, 1998.

[7] C. Warrender, S. Forrest and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models", *Proceedings of 1999 IEEE Symposium on Security and Privacy*, pp. 133-145, 1999.

[8] K. Llgun, R. A. Kemmerer, Fellow, IEEE and P. A. Porras, "State Transitions Analysis: A Rule-based Intrusion Detection Approach", 1995.

[9] S. Forrest, S. A. Hofmeyr, A. Somayaji and T. A. Logstaff, "A sense of Self for Unix process", *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, pp. 120-128, 1996.

[10] S. Jha, K. Tan, and R. A. Maxion, "Markov Chains, Classifiers, and Intrusion Detection," in *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pp. 206-219, 2001.

[11] T. Lunt, A. Tamaru, F. Gilham, R.Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. *A real-time intrusion detection expert system (IDES) – final technical report*. Teccnical Report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[12] J.B.D. Cabrera, B. Ravichandran, and R. K. Mehra, "Statistical Traffic Modeling for Network Intrusión Detection," in *Proceedings of the 8th IEEE International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems*, pp. 466-473, 2000.

[13] M. Bykoba, S. Ostermann, and B. Tjaden, "Detecting Network Intrusions via a Statistical Analysis of Network Packet Characteristics," in *Proceedings of the 33rd IEEE Southeastern Symposium on System Theory*, pp. 309-314, 2001.

[14] S. Zheng, C. Peng, X. Ying, and X. Ke, "A Network State Based Intrusion Detection Model," in *Proceedings of the International IEEE Conference on Computer Networks and Mobile Computing*, pp. 481-486, 2001.

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

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

[17] J. Postel, "Transmission Control Protocol", RFC793, September 1981.

[18] V. Jacobson, C. Leres, and S. McCanne, *tcpdump*, http://www.tcpdump.org, June 1994.

[19] Fyodor, "Nmap – Free Stealth Port Scanner for Network Exploration & Security Audits". Available: http://www.insecure.org/nmap/index.html