Measuring the Normality of Web Proxies' Behavior Based on Locality Principles

Yi Xie and Shun-zheng Yu

Department of Electrical and Communication Engineering,
Sun Yat-Sen University, Guangzhou 510275, P.R. China
xieyicn@163.com, syu@sysu.edu.cn

Abstract. Web Proxy and cache play important roles in the modern Internet. Although much work has been done on them, few studies were focused on the fact that these trusted intermediaries may be utilized to launch Web-based attacks and to shield the attackers' malicious behavior. This paper fills an void in this area by proposing a new server-side detection scheme based on the behavior characteristics of proxy-to-server Web traffic. Proxy's access behavior is extracted from the temporal locality and the bytes of the requested objects. A stochastic process based on Gaussian mixtures hidden semi-Markov model is applied to describe the dynamic variability of the observed variables. The entropies of those pending Web traffics launched by proxies fitting to the model are used as the criterion for attack detection. Experiments based on the real Web traffic and an emulated attack are implemented to valid the proposal.

1 Introduction

Since 1994, Web proxies and caches have been widely deployed to reduce network traffic and provide better response time for Web accesses. The primary aim of proxy is to allow users to access the Web within a firewall. This type proxy runs on a firewall machine and waits for a request from inside the firewall, then, forwards the request to the remote server outside the firewall, reads the response and then sends it back to the client. Apart from letting clients access resources on the Web, proxies have various uses including sharing of various resources, caching, anonymization, transformation of requests/responses, transfer between different protocol system, and filtering/modifying requests/responses. Although many studies have been done on Web proxy, most of them were only focused on performance improvement (e.g., hit ratio, prefetch algorithm, item update model) instead of the security issues.

In this paper, we study the server-side security issues caused by Web proxies and explore the early detection for a new type Web-based attack which is launched by utilizing the vulnerabilities of Web proxy based on HTTP protocol. Being different from other traditional Web-based attacks whose aim is only to shut down the victim or steal customs' privacy information or illegally use Web applications, such attack usually prevents legitimate users from using the service by consuming server's available CPU slots and memory resources. In order to avoid the server-side detection and Web proxies' bypassing, attackers may keep the victim server alive and locate behind proxies during the attack period. The malicious attack behavior are tunneled to the Web server by utilizing various types of HTTP requests (e.g., dynamic Web pages or HTTP requests with "*No-Cache*" headers).

The motivations of proposing a scheme for such attack exist in many aspects: (i) This type attack utilizes the HTTP and the opening TCP port 80 to pass all low-layer firewalls and anomaly detection systems, thus, most detection methods based on IP header or TCP connection (e.g., those surveyed in [1]) become invalid. (ii) According to the "request-response" mechanism of HTTP, Web server has to return response for each incoming request. This working method creates the chance of attack. Furthermore, attackers may also use the normal but expensive computational complexity HTTP requests to consume the server-side resources, thus, it is difficult for those designed for flooding attacks and SQL injection attacks (e.g., [2] [3]) to discover the anomaly signals of such attack. (iii) Attackers are often unseen to the Web server because of the anonymization function of hierarchical proxy architecture. (iv) Measuring the system resource consumption rate (e.g., CPU or memory utilization) maybe a good way to discover the abnormalities caused by such attack. However, this method is not conducive to the early detection or realtime monitoring because when the monitor finds the system resources are occupied abnormally, attack has been successfully going on for a very long time. To the best of our knowledge, few work has been done on this field. This paper proposes a novel server-side anomaly detection scheme to meet this new challenge and fills an void in this area based on proxies' access behavior. The remainder of this paper is organized as follows. In section 2, we introduce rational of our scheme. We valid the proposal by the experiments in section 3 and conclude this work in section 4.

2 Rationale of the Proposed Scheme

Much previous work has approved that statistics is a good method for anomaly detection. Thus, our scheme is also based on statistical methods. One difference between our method and other existing anomaly detection systems is that we implement the anomaly detection by measuring the normality of proxies' application-layer access behavior instead of the IP headers or TCP connections. In order to achieve this aim, we introduce a new way based on temporal locality to extract the access behavior of Web proxies. Then a stochastic process based on Gaussian mixtures hidden semi-Markov Model is used to describe the variety of the normal access behavior and implement the anomaly detection for the pending proxy-to-sever Web traffic.

2.1 Stack distance model for temporal locality

Temporal locality of reference is one of the cornerstones of computer science. Primitively, it was born from efforts to make virtual memory systems work well

. After that, temporal locality has been widely applied in many fields, e.g., Memory behavior [4], CPU cache [5], program behavior [6], Characterizing reference pattern of Web access [7] and Web proxy cache replacement strategy and performance improvement [8].

Intuitively, temporal locality refers to the property/likehood that referencing behavior in the recent past is a good predictor of the referencing behavior to be seen in the near future, whereas resource popularity metric only represents the frequency of the requests without indicating the spacing between the requests, i.e., the correlation between a reference to a document and the time since it was last accessed.

The temporal locality of Web traffic can be defined by the following probability function:

$$\begin{aligned} F(t) \overset{\text{def}}{=} Prob[&document\ i\ is\ referenced\ at\ time \\ x+t \mid &document\ i\ was\ last\ referenced\ at\ time\ x] \end{aligned} \tag{1}$$

Stack distance model is often utilized to capture the temporal locality relationships in most previous work, e.g., [7]. We denote a reference stream $\mathcal{R}_i = \{r_1, r_2, \cdots, r_i\}$, where r_i denotes the i^{th} requested document's name. Index i indicates that i requests have already arrived at a server. We define the least recently used (LRU) stack $\mathcal{L}_i$, which is an ordering of all documents of a server by recency of usage. Thus, at index i, the LRU stack is given by $\mathcal{L}_i = \{u_1, u_2, \cdots, u_N\}$, where $u_1, u_2, \cdots, u_N$ are documents of the server and u_1 is the most recently accessed document, u_2 the next most recently referenced, etc. In other words, u_1 is just accessed at index i, i.e., $r_i = u_1$. Whenever a reference is made to an document, the stack must be update. Considering that $r_{i+1} = u_j$, then the stack becomes $\mathcal{L}_{i+1} = \{u_j, u_1, u_2, \cdots, u_N\}$. Suppose now that $\mathcal{L}_{i-1} = \{u_1, u_2, \cdots, u_N\}$ and $r_i = u_j$, i.e., the request r_i is at distance j in stack $\mathcal{L}_{i-1}$. Let d_i denote the stack depth of the document referenced at index i. Then, a new relation can be obtained by the following equation "$if\ r_i = u_j\ then\ d_i = j$", where j denotes the stack depth of the requested document at index i. An example of this LRU stack model is shown in Figure 1. Based on the initial stack ($\mathcal{L}_0 = \{C, E, A, D, B\}$), the final stack distance sequence correspondent to the input reference string ($\mathcal{R}_9 = \{A, D, C, A, B, D, E, A, B\}$) is $\{3, 4, 3, 3, 5, 4, 5, 4, 4\}$. In other words, the reference symbol stream $\mathcal{R}_i = \{r_1, r_2, \cdots, r_i\}$ is transformed to a numerical stream $\mathcal{D}_i = \{d_1, d_2, \cdots, d_i\}$ based on stack distance model.

2.2 Profiling the access behavior

Since we focus on the anomaly detection of proxy-to-server Web traffic, we need to profile the normal proxy behavior. Much previous work on Web proxy [7] [8] has approved stack of object references is a good model for characterizing the behavior of proxies and caches. The main advantage of stack distance model for describing the Web proxies' access behavior is that a request string can be converted into a distance string that preserves the pattern of activity, but does

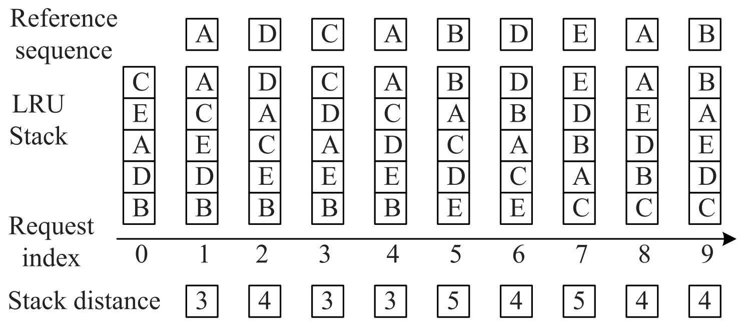


Fig. 1. Least recently used stack model

not depend on document names. For these reasons, we apply the stack distance model to extract the proxy behavior characteristics in this paper.

Considering downloading is another main factor affecting the server performance, we also take into account the bytes of requested documents. One problem of obtaining the bytes of requested documents is, if the requests are for dynamic Web pages, front-end detection system may not know the exact values of bytes before the server responds the requests. If we put the detection system on the outgoing path, it will be not conducive to the realtime detection. Thus, we use a compromise solution to resolve this issue here. We build a database to record the bytes of previously visited documents or routines (e.g., JAVA scripts). When a request arrives at the front-end detection system, the system looks up the database and estimate the bytes of its corresponding response. If the requested document or routine can not be found in the database, we will use the mean bytes of the documents/routines recorded in database. In order to avoid confusion between the estimated bytes value and the actual one, we call the response bytes used in this paper as "prospective return bytes (PRBs)". The PRBs of the corresponding document or routine recorded in the database will be updated by exponential forgetting based on the server's response if the incoming request is legitimate:

$$s_i(j) = (1-\rho)s_i(j-1) + \rho s_{ij},\ 1 \geq \rho \geq 0 \tag{2}$$

where ρ is the decay rate, $s_i(j)$ and s_{ij} is the new PRBs and the real response bytes value of the i^{th} object (document/routin) requested at the j^{th} time.

Thus, assuming a HTTP reference stream $\mathcal{R}_i = \{r_1, r_2, \cdots, r_i\}$, we have two corresponding numerical streams: $\mathcal{D}_i = \{d_1, d_2, \cdots, d_i\}$ and $\mathcal{S}_i = \{s_1, s_2, \cdots, s_i\}$, where s_i denote the PRBs of the document referenced at index i.

If we use $\mathbb{M}(t)$ to denote the set of HTTP requests which appear during the t^{th} second, and use $\mathbb{N}(t)$ to denote the size (i.e., the number of requests) of $\mathbb{M}(t)$, then, the average stack distance value and average PRBs per second can be respectively calculated by the following equations:

$$\{\bar{d}_t, \bar{s}_t\} = \frac{1}{\mathbb{N}(t)} \sum_{\forall i: r_i \in \mathbb{M}(t)} \{d_i, s_i\}, \qquad t = 1, 2, ... \tag{3}$$

For brevity of notation, we use $\boldsymbol{o}_t$ to denote a time series which is made up of $\bar{d}_t$ and $\bar{s}_t$, i.e., $\boldsymbol{o}_t = (\bar{d}_t, \bar{s}_t)$, where t is the index of second. Previous work [7] [8] indicated that Web temporal locality streams are statistically self-similar or long-range dependence and that this property can be used to explain aspects of the request string that are the result of spatial locality [7]. This means that correlations between object references can occur at widely varying timescales. Such characteristics can have a significant impact on profiling the proxy-to-server traffic. Therefore, better understanding of the nature of the temporal locality is useful to profile the access behavior of proxy-to-server Web traffic. In order to achieve this aim, we use a stochastic process to describe the dynamic variation of stack distance value instead of the traditional ways designed for proxy performance improvement based on pure statistical method (e.g., mean or variance).

Among existing stochastic models, hidden semi-Markov Models (HsMMs) [9] is one of the useful tools to describe most practical stochastic signals without too many assumptions. A major advantage of using the HsMM is its efficiency in estimating the model parameters to account for an observed sequence. Furthermore, it can capture various statistical properties of time series, including long-rang and short-rang dependence, non-stationary and the non-Markovian [10]. Thus, HsMMs have been widely applied in many areas such as mobility tracking in wireless networks, activity recognition in smart environments, and inference for structured video sequences.

In this paper, we assume the time series $\{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_T\}$ is controlled by an underlying Markov Chain. Each underlying Markov state represents one type joint probability distribution of average stack distance value and average PRBs per second, or say a type of proxy's access behavior pattern. Transition of the hidden Markov states implies the change of proxy's access behavior pattern from one kind to another one. Residential duration of the Markov state can be considered as persistence of the request profile.

Let $x_1, x_2, \cdots, x_M$ be states of a semi-Markov chain, q_t denote the state of the semi-Markov chain at time t and $\lambda = \{\pi_m, a_{mn}, b_m(\boldsymbol{o}_t), p_m(d)\}$ be the parameters of a given HsMM, where $\pi_m \equiv Pr[q_1 = x_m|\lambda]$ is the initial state probability function, $a_{mn} = Pr[q_t = x_n|q_{t-1} = x_m, \lambda]$ the state transition probability function, $b_m(\boldsymbol{o}_t) = Pr[\text{output vector of the model is } \boldsymbol{o}_t|q_t = x_m, \lambda]$ the output probability function and $p_m(d) = Pr[\text{duration of } q_t \text{ is } d|q_t = x_m, \lambda]$ the state duration probability function.

The rational and parameter estimation of discrete HsMM can be found in [9]. In this paper, we simplify the HsMM by applying the Gaussian mixtures into the output probability function $b_m(\boldsymbol{o}_t)$, i.e.:

$$b_m(\boldsymbol{o}_t) = \sum_{k=1}^{\kappa} c_{mk} b_{mk}(\boldsymbol{o}_t) = \sum_{k=1}^{\kappa} c_{mk} \mathcal{N}(\boldsymbol{o}_t, \boldsymbol{\mu}_{mk}, \Sigma_{mk}), \tag{4}$$

where κ is known; $c_{mk} \geq 0$ for $1 \leq m \leq M$, $1 \leq k \leq \kappa$; $\sum_{k=1}^{\kappa} c_{mk} = 1$ for $1 \leq m \leq M$; and $\mathcal{N}(\boldsymbol{o}, \boldsymbol{\mu}, \Sigma)$ denotes the multi-dimensional normal density function of mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. We also assume the transition of hidden states obeys the Birth-death process, i.e., $a_{mn} = 0$ when $|n - m| > 1$.

We directly use the forward and backward variables and three joint probability functions defined in [9] , i.e. :

$$\alpha_t(m,d) \overset{\text{def}}{=} P[\boldsymbol{o}_1^t, q_t = x_m, \tau_t = d|\lambda] \tag{5}$$

$$\beta_t(m,d) \overset{\text{def}}{=} P[\boldsymbol{o}_{t+1}^T|q_t = x_m, \tau_t = d, \lambda] \tag{6}$$

$$\zeta_t(m,n) \overset{\text{def}}{=} P[\boldsymbol{o}_1^T, q_{t-1} = x_m, q_t = x_n|\lambda] \tag{7}$$

$$\eta_t(m,d) \overset{\text{def}}{=} P[\boldsymbol{o}_1^T, q_{t-1} \neq x_m, q_t = x_m, \tau_t = d|\lambda] \tag{8}$$

$$\gamma_t(m) \overset{\text{def}}{=} P[\boldsymbol{o}_1^T, q_t = x_m|\lambda] \tag{9}$$

where τ_t denotes the state duration of q_t. Then, We define the probability that the k^{th} component of the m^{th} mixture generated observation $\boldsymbol{o}_t$ as

$$\begin{aligned} \gamma_t(m,k) &\overset{\text{def}}{=} \Pr(q_t = m, Y_{mt} = k|\boldsymbol{O}, \lambda) \\ &= \gamma_t(m)\frac{c_{mk}b_{mk}(\boldsymbol{o}_t)}{b_m(\boldsymbol{o}_t)} \end{aligned} \tag{10}$$

where Y_{mt} is a random variable indicating the mixture component at time t for state m. When there are E observation sequences the e^{th} being the length of T_e, the parameters of this parametric HsMM can be iteratively calculated by:

$$\hat{\pi}_m = \sum_{e=1}^{E} \gamma_1^e(m)/E \tag{11}$$

$$\hat{a}_{mn} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \zeta_t^e(m,n)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{n=1}^{M} \zeta_t^e(m,n)} \tag{12}$$

$$\hat{c}_{mk} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m,k)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m)} \tag{13}$$

$$\hat{\boldsymbol{\mu}}_{mk} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m,k)\boldsymbol{o}_t^e}{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m,k)} \tag{14}$$

$$\hat{\Sigma}_{mk} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m,k)(\boldsymbol{o}_t^e - \boldsymbol{\mu_{mk}})(\boldsymbol{o}_t^e - \boldsymbol{\mu_{mk}})^{\mathrm{T}}}{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(m,k)} \tag{15}$$

$$\hat{p}_m(d) = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \eta_t^e(m,d)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{d=1}^{D} \eta_t^e(m,d)} \tag{16}$$

Most previous work have approved Viterbi algorithm is a good method for decoding. Thus, we modify the Viterbi algorithm designed for hidden Markov

Model [11] for our HsMM based on the following recursion for $\delta_t(m)$, the posterior probability of the best state sequence ending in state m at time t:

$$\delta_t(m) = \max_d \delta^*_{t-d}(m) p_m(d) b_m(\boldsymbol{o}_{t-d+1}, ..., \boldsymbol{o}_t) \tag{17}$$

$$\delta^*_t(m) = \max_m \delta^*_t(m) a_{mn} \tag{18}$$

2.3 Detection Scheme

We use an average logarithmic likelihood $\varepsilon(t)$ (i.e., entropy) of observations $(\boldsymbol{o}_1^t)$ fitting to the given HsMM (λ) as the detection criterion at the t^{th} second. The $\varepsilon(t)$ is defined as Equation (19):

$$\varepsilon(t) \overset{\text{def}}{=} \frac{1}{t} log\{P[\boldsymbol{o}_1^t|\lambda]\} = \frac{1}{t} log\{\sum_{m,d} \alpha_t(m,d)\} \tag{19}$$

The whole anomaly detection scheme is outlined in Figure 2. The details are shown as the following. When a proxy's reference string reaches the ingress of the victim Web server, the detection scheme begins to work. First, Information Extraction (*IE*) module is performed on the incoming observed data for calculating the average stack distance and requested file size (i.e., $\boldsymbol{O}_t = \boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_t$). If the data are used for training model, they will be sent to the Iteration Calculation (*IC*) module which will output the parameters (λ) of the model to the Forward Process (*FP*). Otherwise, the observed data may be directly sent to the *FP* which will form the decision of pending reference string's normality based on its entropy. If the decision is positive, the switch between the Data Pool (*DP*) and the Service Queue (*SQ*) will be kept on opening to prevent those suspected HTTP requests from entering the *SQ* module and affecting the Web server performance. In order to avoid the overflow of *DP*, we can start a timer while a new reference stream enters the *DP*. When the value of timer is zero, the corresponding reference stream will be deleted. Once, the decision of the pending request stream is negative, switch will be put on for transmitting the correspondent reference string to the *SQ* module where the proxy's requests are waiting for Web Server's response according to the First in First out (FIFO) policy or other Quality of Service (QoS) strategies.

3 Experiments and numerical results

In this section, we use a 72 hours real Web traffic which includes 258 proxies to valid our detection scheme. The simulation is implemented in the NS2 simulator [12] by two application-layer modules (i.e., "Web cache application" and "PackMime-HTTP: Web Traffic Generation"). There are three phases in our simulation: the first 24 hours traffics are used for constructing a stable temporal locality stack for proxies; the second 24 hours traffics are used for model training and the remaining are used for testing. During the attack period, we let

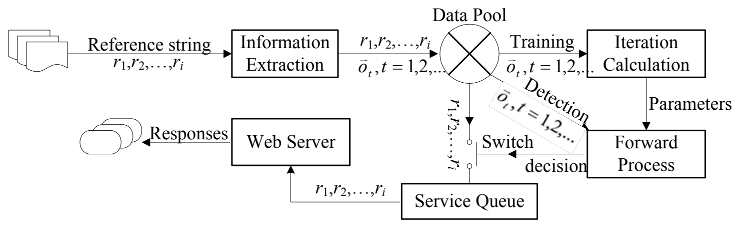
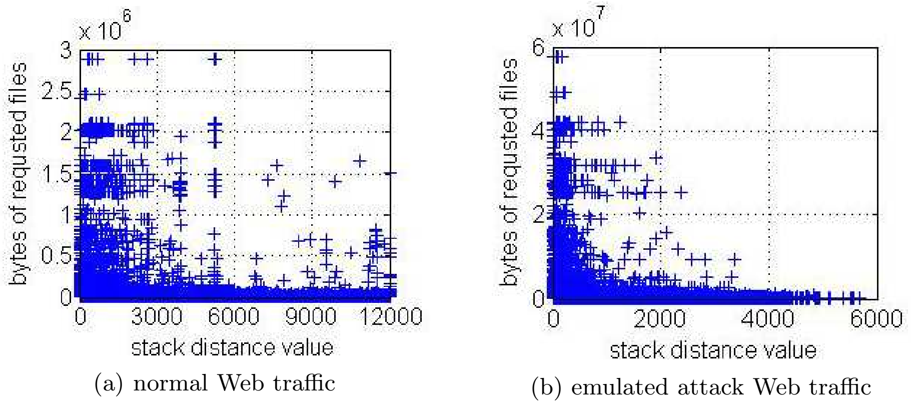


Fig. 2. Detection scheme for proxy-to-server Web traffic



(a) normal Web traffic (b) emulated attack Web traffic

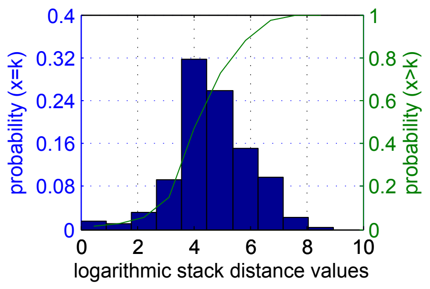
Fig. 3. joint distribution of stack distance and bytes in different periods

the emulated "bad" clients replay the normal users' requests, most of which are for dynamic contents(e.g., database searching) and large byte documents (e.g., audio and video). Once the "bad" requests are produced, they are sent to the proxies. Thus, both the "bad" requests and the "good" requests of proxies are sent to the victim Web server.

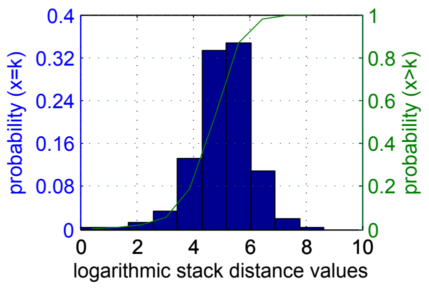
The time unit of the final observation $\boldsymbol{o}_t$ is second. This time series is then blocked into frames. Each frame spans 10 seconds or 10 observed vectors. Consecutive frames overlap by 5 seconds. On the other words, each frame is multiplied by a Hamming Window with width of 10 seconds and applied every 5 seconds.

3.1 Statistical Analysis for observed data

In Figure 3 , we compare the joint distributions of stack distance and PRBs between period of the normal Web traffic and the emulated attack period. We can find that most points of both the normal and attack periods fall into the low-value area. Furthermore, during the emulated attack period, the largest stack distance value is much smaller than that of the normal period. This result shows that it is not easy for us to distinguish the attack requests from the normal requests sent by proxies only by the statistical properties.

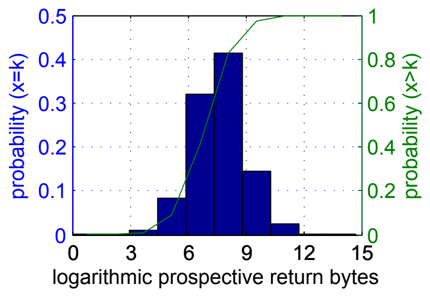


(a) normal Web traffic

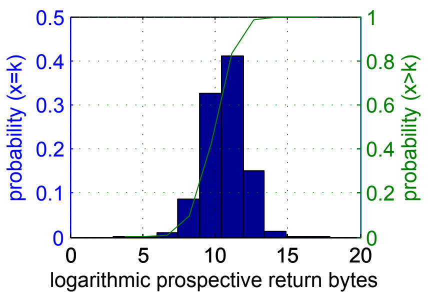


(b) emulated attack Web traffic

Fig. 4. marginal distribution of stack distance in different periods



(a) normal Web traffic



(b) emulated attack Web traffic

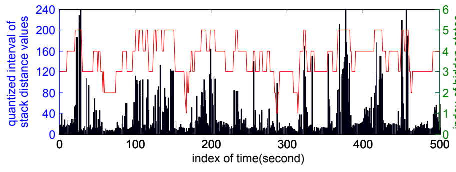
Fig. 5. marginal distribution of prospective return bytes in different periods

We plot the distribution of logarithmic stack distance values and PRBs of different testing data in Figure 4 and Figure 5, respectively. All these figures are Gaussian-like distributions with similar means and variances. These results also show, it is ineffective for the pure statistical methods to distinguish the abnormal requests from the normal ones.

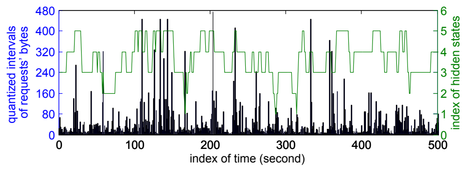
3.2 Detection based on HsMM

A five-state HsMM is used in this experiment. The model parameters are obtained based on the previous HsMM algorithm. In Figure 6 and Figure 7, we use a 500 seconds Web traffic fragment of one of the sample proxies to show the stochastic processes of stack distance values and PRBs based on HsMM. Comparing both the hidden state processes of normal and emulated attack testing periods, we can find the obvious differences: dynamic range of state transition of normal Web traffic is wider and more homogeneous than that of the evaluated attack Web traffic. This shows the hidden semi-Markov states can be used to extract the proxy behavior and to recognize the abnormality.

Since the hidden semi-Markov state sequences are different between the normal proxy traffic and the abnormal one, the distribution of hidden state can be

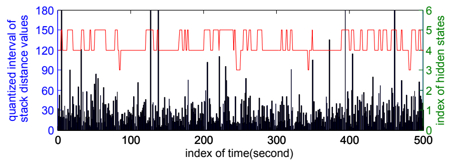


(a) stack distance sequence

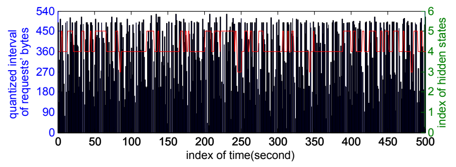


(b) PRB sequence

Fig. 6. profiling the normal proxies behavior by hidden states

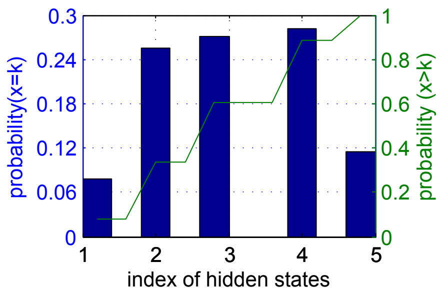


(a) stack distance sequence

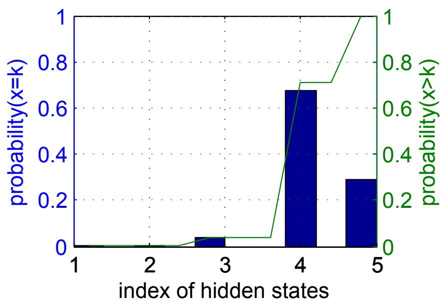


(b) PRB sequence

Fig. 7. profiling the abnormal proxies behavior by hidden states



(a) normal Web traffic



(b) emulated attack Web traffic

Fig. 8. distribution of hidden states in different periods

used intuitively for the anomaly detection. In Figure 8, we show the histograms of hidden states.

It is not credible to draw a conclusion only based on the intuitionists results. In order to build an automatical and numerical detection system, we use the entropy defined in Equation (19) as the measure criterion. We show the entropy distribution and the corresponding cumulative distribution of the training data in Figure 9, from which we find most entropies of normal proxy-to-server Web traffic are belong to [-10,-7]. Since the entropy distribution of normal proxy-to-server is concentrated, it can be used as a criterion to achieve the anomaly detection for the proxy-to-server Web traffic.

Two data sets are used to test the model performance. The first one is used to validate the False Positive Rate (FPR) of the model. It includes the normal proxy-to-server Web traffics which occur after the training data and are pro-

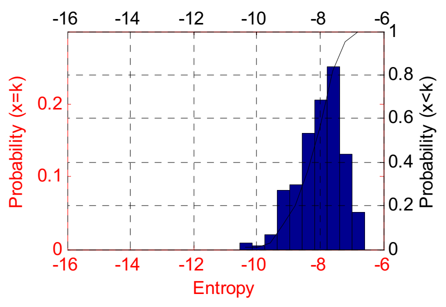


Fig. 9. Entropy of training data

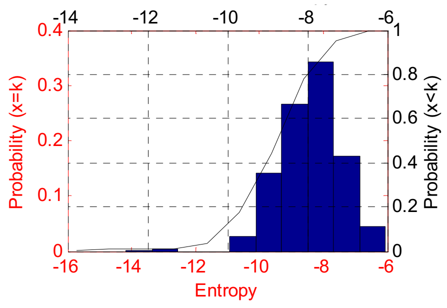


Fig. 10. Entropy of normal testing data

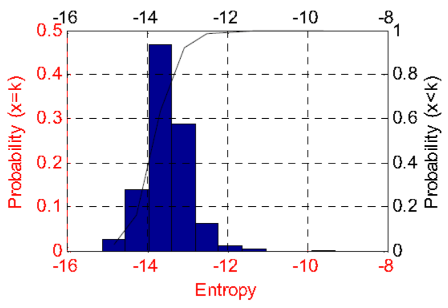


Fig. 11. Entropy of emulated attack data

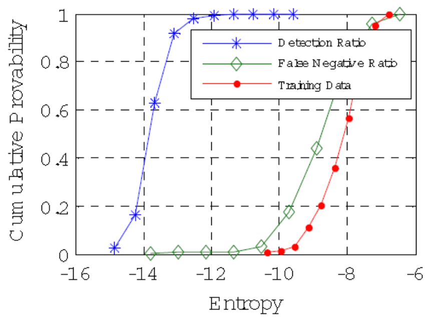


Fig. 12. Detection performance

duced by those proxies that never appear in the training process. The second one including the emulated attacks is used to verify the scheme's Detection Rate (DR). Two attack forms are considered in this experiments, which include the dynamic page attack and the downloading attack. We plot the entropy distributions and the corresponding cumulative distributions of these two data sets in Figure 10 and Figure 11, respectively.

Comparing Figure 9 with Figure 10 and Figure 11, we see, although the sources and the time are quite different between the training data and the normal test data, most entropies of them fall into the same range . The result means the statistical properties of our observations does not depend on or bind with the name of requested documents which are varying with time. Thus, the model is fairly stable for the normal proxy-to-server Web traffic. However, as shown in Figure 11, the entropy distribution of the traffic mixed with emulated attack requests are quite different from those of previous ones, which shows the model is sensitive to the unusual request pattern. This characteristic is very useful for detecting the potential abnormity of proxy-to-server Web traffic. In Figure 12, we plot the receive operating characteristic (ROC) curve which shows the $FPR = 0.1\%$ and $DR = 98\%$ when the decision threshold of entropy takes the value of -12.

4 Conclusion

An early detection scheme focusing on a Web-based attack which utilizes the proxy-to-server Web traffic to shield the attack behavior, is proposed in this paper. Based on the stack distance of temporal locality, Gaussian mixtures HsMM is applied to profile the access behavior characteristics of proxy-to-server traffic and carry out the anomaly detection. The numerical results of experiment demonstrate that the proposed method is expected to be practical in monitoring the attacks hidden in the proxy-to-server traffic.

Acknowledgment

This work was supported by the key Program of NSFC-Guangdong Joint Funds (Grant No.U0735002) and the National High Technology Research and Development Program of China (Grant No.2007AA01Z449). It was performed while the author was with George Mason University as a visiting PHD student.

References

1. Patcha, A., Park, J.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks **51**(12) (2007) 3448–3470
2. Ranjan, S., Swaminathan, R., Uysal, M., Knightly, E.: DDoS-Resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection. Proceedings of IEEE INFOCOM, Barcelona, Spain, April (2006) 1–13
3. Zhang, L., White, G.: Anomaly detection for application level network attacks using payload keywords. Computational Intelligence in Security and Defense Applications, 2007. CISDA 2007. IEEE Symposium on (1-5 April 2007) 178–185
4. Smith, A.: Cache Memories. ACM Computing Surveys (CSUR) **14**(3) (1982) 473–530
5. Hill, M., Smith, A.: Evaluating Associativity in CPU Caches. IEEE Transactions on Computers **38**(12) (1989) 1612–1630
6. Spirn, J.: Distance String Models for Program Behavior. Computer **9**(11) (1976) 14–20
7. Almeida, V., Bestavros, A., Crovella, M., de Oliveira, A.: Characterizing reference locality in the WWW. Parallel and Distributed Information Systems, 1996., Fourth International Conference on (1996) 92–103
8. Mahanti, A., Eager, D., Williamson, C.: Temporal locality and its impact on Web proxy cache performance. Performance Evaluation **42**(2-3) (2000) 187–203
9. Yu, S.Z., Kobayashi, H.: An efficient forward-backward algorithm for an explicit-duration hidden Markov model. Signal Processing Letters, IEEE **10**(1) (2003) 11–14
10. Yu, S.Z., Liu, Z., Squillante, M., Xia, C., Zhang, L.: A hidden semi-Markov model for web workload self-similarity. Performance, Computing, and Communications Conference, 2002. 21st IEEE International (2002) 65–72
11. Rabiner, L.: A tutorial on hidden Markov models and selected applications inspeech recognition. Proceedings of the IEEE **77**(2) (1989) 257–286
12. NS2: (Ns2,http://www.isi.edu/nsnam/ns/)