EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Sequence-order-independent network profiling for detecting application layer DDoS attacks

Sangjae Lee[1*], Gisung Kim[1] and Sehun Kim[2]

Abstract

Distributed denial of service (DDoS) attacks, which are a major threat on the Internet, have recently become more sophisticated as a result of their ability to exploit application-layer vulnerabilities. Most defense methods are designed for detecting DDoS attacks on IP and TCP layers and consequently have difficulty in detecting this new type of DDoS attack. With the profiling of web browsing behavior, the sequence order of web page requests can be used for detecting the application-layer DDoS (App-DDoS) attacks. However, the sequence order may be more harmful than helpful in the profiling of web browsing behaviors because it varies significantly for different individuals and different browsing behaviors. This article introduces a sequence-order-independent method for the profiling of network traffic and the detection of a new type of App-DDoS attacks. Four attributes are extracted from web page request sequences without consideration of the sequence order of requested pages. A model based on the multiple principal component analysis is proposed for the profiling of normal web browsing behaviors, and its reconstruction error is used as a criterion for detecting DDoS attacks. The proposed method is experimentally confirmed with various types of new App-DDoS attacks.

1. Introduction

Distributed denial of service (DDoS) attacks have become a major threat and one of the hardest problems to overcome on the Internet. Various activities, such as telecommunication, online banking, and online shopping, have recently been integrated through the Internet, yet the Internet is now plagued by more than 10 million infected hosts (or zombies) [1]. DDoS attacks have consequently become a serious threat.

DDoS attacks traditionally exploit the vulnerabilities of a network layer, particularly SYN flooding, UDP flooding, and ICMP flooding. These attacks exhaust the network bandwidth and resources of the victim; so that legitimate access is denied. Although many studies have developed defense methods, sophisticated DDoS attacks can now overcome these methods.

One recent sophisticated DDoS attack is called an application-layer DDoS (App-DDoS) attack [2]. Unlike conventional DDoS attacks, this type of attacks exploits vulnerabilities at the application layer rather than at the network layer. App-DDoS attacks send small packets of legitimate content via normal successful TCP connections; no spoofed IP address with standard services such as HTTP and HTTPS. Thus, the DDoS defensive methods mistakenly regard App-DDoS attacks as normal connections. Furthermore, App-DDoS attacks are similar to a flash crowd event, which happens when massive numbers of normal users simultaneously send requests to one web server [3]. Hence, it is difficult to distinguish App-DDoS attacks from legitimate normal traffic.

There are several DDoS defense methods that utilize application-layer information. Ranjan et al. [4] analyzed time-related characteristics of HTTP sessions, such as session inter-arrival time, request inter-arrival time, and session arrival time. Yatagai et al. [5] presented a method that analyzes the correlation between browsing time and page information size. However, time-related features are insufficient to detect App-DDoS attacks because attackers can easily control packet-sending rates by utilizing a large-scale botnet [6]. On the other hand, Kandula et al. [7] developed a system that protects a web server from DDoS attacks by implementing a probabilistic authentication method using CAPTCHAs, but the task of requiring users to solve graphic puzzles causes additional service delays. As a result, the graphic

* Correspondence: sj_lee@kaist.ac.kr
[1]Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Republic of Korea
Full list of author information is available at the end of the article

puzzles cause annoying legitimate users as well as act as another DDoS attack point.

For the detection of App-DDoS attacks, Xie et al. [8] used a hidden semi-Markov model (HsMM) to describe the normal browsing behavior of web users. The HsMM uses the sequence order of web page requests to profile normal web browsing behavior. To detect App-DDoS attacks, they defined a normality threshold and compared it with the model's output values of incoming users. However, the sequence-order-based method can be complex and may cause many false alarms. The sequence order might vary significantly for different individuals and for different browsing behaviors. For example, web users can directly type URLs to request resources or utilize external web links. Furthermore, they can browse the resources of the web server with multiple browsers, possibly causing changes in the relative sequential positions.

In this article, we propose a sequence-order-independent method that distinguishes App-DDoS attacks from normal traffic. We regard the sequence order as a kind of noise rather than good information. We first extract the sequence-order-independent informative attributes from web page request sequences; these attributes represent a web user's activeness, pages of interest, and the breadth and intensity of their interest. We describe them in a matrix and use multiple principal component analysis (PCA) to model the browsing patterns. We then use the reconstruction error of the multiple PCA as a criterion for distinguishing App-DDoS attacks from normal usage.

This article is organized as follows. In Section 2, we describe App-DDoS attacks. In Section 3, we propose a new App-DDoS detection method that includes new sequence-order-independent attributes and a multiple PCA model. In Section 4, we validate our detection model with real traffic and discuss an early warning system. Finally, in Section 5, we present our conclusions.

2. App-DDoS Attacks

App-DDoS attacks exploit victim servers by exhausting the resources such as sockets, CPU, memory, and disk bandwidth. According to [9], server resources may become the bottleneck of the Internet applications. App-DDoS attacks are also launched on mobile devices, such as smart-phones and ubiquitous sensors, because they require few resources in the client side. Thus, App-DDoS attacks cause more serious problems than in the past.

Figure 1 shows an example of an App-DDoS attack. The attacker first exploits a popular web server, the worm distribution server, to insert malicious codes. It causes web users to download malicious codes and makes their hosts become infected. When the attacker begins an attack via the command server, an excessive number of infected hosts make requests for web pages from the victim; as a result, the victim's resources are eventually exhausted.

One of the differences between conventional DDoS and App-DDoS attacks is that App-DDoS attacks utilize only legitimate methods instead of vulnerabilities of protocols. App-DDoS attacks usually send small packets via successful TCP connections, and real IP addresses thus are used to launch attacks. In particular, App-DDoS attacks send packets through standard services such as HTTP and HTTPS. Moreover, these packets are generated with various sending rates to mimic legitimate users. Thus, these application-layer requests are indistinguishable from those generated by legitimate users.

3. Proposed method

In this section, we propose a defensive method that can be used for detecting an App-DDoS attack. We show how to represent a set of web browsing behaviors using sequence-order-independent attributes instead of web page request sequences. We then present a multiple

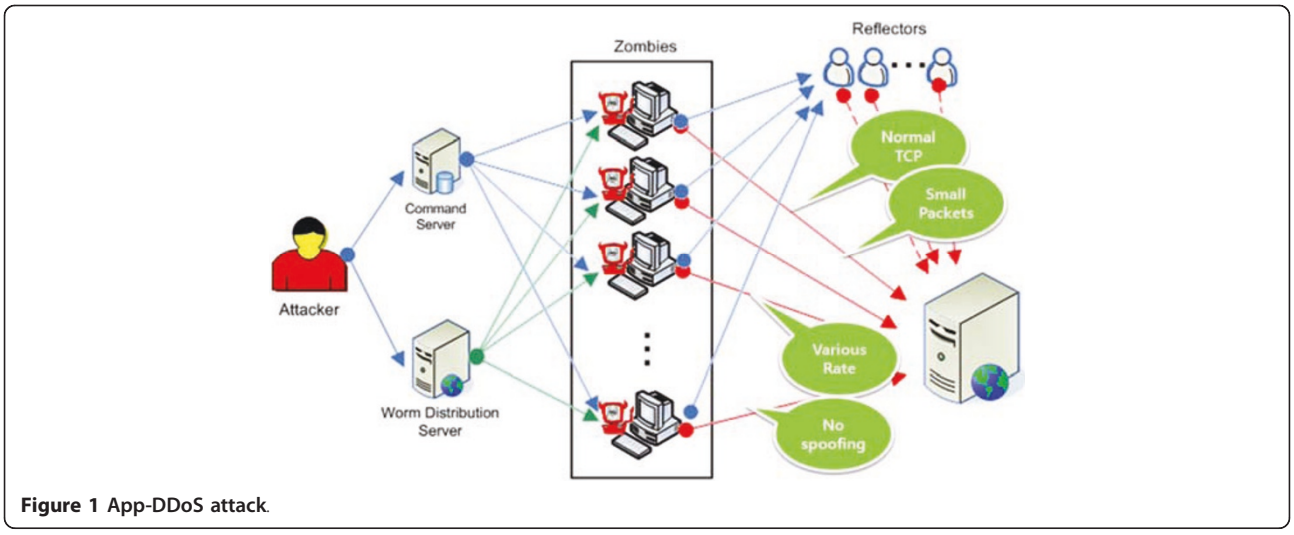

Figure 1 App-DDoS attack.

PCA model to profile normal web browsing patterns and distinguish App-DDoS attacks. Since DDoS attack detection systems are required to handle an extremely large volume of traffic, we base our description of the web browsing patterns on PCA instead of nonlinear methods such as kernel methods and manifold learning [10,11].

3.1. Sequence-order-independent attributes

We represent each request sequence as a vector form of extracted attributes. Let us assume that N users browsed a web server where the total number of web pages is D. For user i who browsed this server, let s_i be the web page request sequence and $\eta_{d,\ i}$ be the number of requests of user i for page d of the server. Then, $L_i = \sum_{d=1}^{D} \eta_{d,i}$ is the total number of requests of user i and $\bar{L} = \sum_{i=1}^{N} L_i/N$ is the average number of requests of users.

Now, let us define several sequence-order-independent attributes for detecting App-DDoS attacks. To give a clearer representation of active user i, we introduce the attribute

$$\tau_i = L_i/\bar{L}, \tag{1}$$

which is the ratio of the number of requests of a user and their average value. The next attribute,

$$h_{d,i} = \eta_{d,i}/L_i, \tag{2}$$

is the proportion of page d among pages requested by user i; it shows how much user i was interested in page d of the server.

To help determine whether incoming users are indicative of a DDoS attack, we supplement the two basic attributes with two other attributes that characterize the web browsing patterns of a user. The first supplementary attribute is defined as the proportion of all server pages requested by user i, i.e.,

$$\alpha_i = \sum_{d=1}^{D} b_{d,i}/D, \tag{3}$$

where $b_{d,\ i}$ is an indicator that equals 1 if $\eta_{d,\ i} >$ and 0 otherwise. This attribute represents the breadth of user i's interest.

The next supplementary attribute shows the intensity of interest in the user's page of greatest interest. This attribute for user i can be defined by using $q_i = \text{argmax}_{\text{d}}\ \{\eta_{d,\ i}\}$ to denote the most frequently requested page. Thus, the intensity of the user's interest in the page of greatest interest can be represented as follows by the ratio of the number of requests for the page of greatest interest and the number of page requests:

$$\gamma_i = \eta_{q_i,i}/L_i. \tag{4}$$

From these attributes, we denote a attribute vector as $w_i = [h_i, \alpha_i, \tau_i, \gamma_i]^{\text{T}}$, where $h_i = [h_{1,i}, h_{2,i},...,h_{D,\ i}]^{\text{T}}$. We then form the attribute matrix $W = [w_1...w_N]$.

3.2. PCA for web browsing behaviors

PCA is the simplest statistical method for transforming given data to new coordinates called principal components. By removing less important components, it can reduce the number of dimensions required to explain the given data. The reduced subspace best represents the given data in a least-squares sense.

We use PCA to model web browsing patterns; the modeling is based solely on normal users' attributes. To model the web browsing patterns, we first denote a mean vector, μ_0, and a covariance matrix, C, as $\mu_0 = \left(\sum_{i=1}^{N} w_i\right) \Big/ N,$, and $C = XX^{\text{T}}/N$, where $X = [x_i...\ x_N]$, $x_i = w_i - \mu_0$, and $i = 1,...,N$. We then compute the eigenvectors and eigenvalues by applying singular value decomposition to the covariance matrix.

If we let u_j be the jth most significant eigenvector of covariance matrix C, then the significant principal components are denoted by $\tilde{U} = \left[u_1 \ldots u_p\right]$, where P ($\ll D$) is the number of significant principal components. Since the remaining eigenvectors $[u_{P+1}...u_{D+3}]$ are less significant, we can reduce the dimensions of the data without significant loss when these eigenvectors are discarded.

If the attribute vectors are projected into the subspace spanned by P significant principal components, then we can represent the attribute of web user i in terms of the following P-dimensional coefficient vector:

$$a_i = \tilde{U}^{\text{T}} x_i, \quad i = 1, \ldots, N \tag{5}$$

This coefficient indicates how much each principal component contributes to the representation of the given attribute.

3.3. Multiple PCA model

Describing real traffic via a single PCA model is difficult because the traffic data usually include many patterns, variations, and different types of noise. We therefore propose to use a multiple form of PCA for effective modeling of real traffic. For the multiple PCA model, we use the k-means clustering method to partition the given data into several clusters [12]. The k-means clustering is a well-known algorithm for unsupervised clustering, but is inappropriate for sparse or concave-shaped

data [13,14]. With our attributes, it is frequently the case that a particular data element may remain zero because some web pages may not be requested for a long period. Accordingly, our attribute matrix may have a high degree of sparsity. To overcome this problem, we perform k-means clustering on the values of the PCA coefficient, a_i, instead of the values of the raw data, w_i. Because the a_i values are low-dimensional and not sparse, we can easily partition the given attributes into several clusters with the coefficient.

Next, we build a PCA model on each cluster. Let $w_i^{(k)}$ be user i's attribute vector that belongs to cluster k. For each cluster, we first normalize the attribute vector by $x_i^{(k)} = (w_i^{(k)} - \mu^{(k)})/(\sigma^{(k)})^2$, where $\mu^{(k)}$ and $(\sigma^{(k)})^2$ are the mean and variance vectors for cluster k, respectively. We then compute P-significant principal components, $U^{(k)}$, for cluster k as described in the previous section. Once the principal components of cluster k are computed, we can reconstruct the original attribute vector with only P principal components. The reconstructed data of $\hat{x}_i^{(k)}$ and their errors, ε_i, are denoted as follows:

$$\hat{x}_i^{(k)} = \left(U^{(k)}\right)\left(U^{(k)}\right)^T x_i^{(k)}, \tag{6}$$

$$\varepsilon_i = \left\| x_i^{(k)} - \hat{x}_i^{(k)} \right\|^2 \tag{7}$$

Designed exclusively for normal traffic, our PCA model produces a good representation of the attributes of normal traffic but a poor representation of the attributes of unseen traffic. As a result, the reconstruction error is low for normal behavior but high for abnormal.

We regard the high reconstruction errors of the PCA as statistical outliers. Hence, we choose a threshold, $\delta^{(k)}$, of cluster k and use it as follows to determine whether the given web browsing behavior is normal:

$$\delta^{(k)} = E[\varepsilon] + \beta\sqrt{E\left[\varepsilon - E[\varepsilon]\right]^2}, \tag{8}$$

where $E[\varepsilon]$ and $E[\varepsilon - E[\varepsilon]]^2$ are the mean and variance of the reconstruction errors for model k, respectively. The β value is a scale factor for defining the outlier range. According to studies on outlier detection, the outlier range should deviate from the mean by more than two or three standard deviations [15].

3.4. Detection method

If a new web user, t, requests a web page from the server, then we first form attribute vector w_t and determine the best fitting model as follows:

$$\pi = \arg\min_k \left\{ \tilde{U}^T(w_t - \mu_0) - m_k \right\}, \tag{9}$$

where $k = 1,...,K$. The value K is the total number of clusters, and m_k is the mean of the PCA coefficients for cluster k. After selecting the best fitting model, π, we normalize w_t using $\mu^{(\pi)}$ and $\sigma^{(\pi)}$. Finally, the model compares the reconstruction error, ε_t, with the error threshold, $\delta^{(\pi)}$. If $\varepsilon_t > \delta^{(\pi)}$, then the model regards the current user as an App-DDoS attack. Figures 2 and 3 show the pseudocodes of the proposed method, which includes model training and testing.

4. Experimental results and analysis

4.1. Datasets

To validate our App-DDoS attack defense method, we used the web-logs from real websites: an educational website, a community website, and an online shopping website. In the educational website, students frequently request some parts of webpages which include educational information. In the community website, users repeatedly request the same webpage which can show different contents using server scripts, such as PHP and ASP. In the online shopping website, customers widely search for their interesting items and wander aimlessly with multiple browsers. We extracted host IPs, request times, and requested webpages from the web-logs, and then constructed sequence datasets; including 30140, 26727, and 99018 sequences. The characteristics of datasets are shown in Table 1. We randomly selected half of the normal sequences to train the model, and for the testing, we used the remainder of the normal sequences as well as the attack sequences. To reliably validate our detection model, we repeated all our experiments with 20 different random seeds.

01: Collect web browsing data from the web server.
02: Remove sequence-order and form the web browsing attribute matrix, W by Eq.(1)-(4).
03: Perform initial PCA Model and compute coefficient vectors by $a_i = \tilde{U}^T x_i$.
04: Perform k-means clustering on the coefficients.
05: For k=1 to K
06: Compute $\mu^{(k)}$ and $\sigma^{(k)}$.
07: Compute $x_i^{(k)}$ by $x_i^{(k)} = (w_i^{(k)} - \mu^{(k)})/\left(\sigma^{(k)}\right)^2$.
08: Compute principal components $U^{(k)}$.
09: Compute $\hat{x}_i^{(k)}$ by $\hat{x}_i^{(k)} = \left(U^{(k)}\right)\left(U^{(k)}\right)^T x_i^{(k)}$.
10: Compute $\delta^{(k)}$ by $\delta^{(k)} = E[\varepsilon] + \beta\sqrt{E[\varepsilon - E[\varepsilon]]^2}$.
11: End

Figure 2 Pseudocode for the training phase of the proposed model.

01: For new session t, form the web browsing attribute vector w_t by Eq.(1)-(4).
02: Find the best model π for an incoming session t using Eq. (9) .
03: Normalize the vector w_t using $\mu^{(\pi)}$ and $\sigma^{(\pi)}$.
04: Compute $\hat{x}_t$ by $\hat{x}_t = \left(U^{(\pi)}\right)\left(U^{(\pi)}\right)^T x_t$.
05: Compute reconstructed error ε_t by Eq. (7).
06: Compare ε_t with the error threshold $\delta^{(\pi)}$
07: IF $\varepsilon_t > \delta^{(\pi)}$
08: The current session is App-DDoS attack.
09: ELSE
10: The current session is *normal* and periodically monitored until the length of requests become enough.
11: END

Figure 3 Pseudocode for the testing phase of the proposed model.

4.2. App-DDoS attacks

To validate our detection model, we performed experiments with three types of attacks. Most studies on App-DDoS attacks restrict their experiments to random page attacks, which involve requests for randomly selected pages from the web server. Random page attacks can be easily detected by defense methods because their patterns are quite different from the behavior of a normal user. However, as mentioned, DDoS attackers have become more sophisticated and now tend to mimic the normal user behavior. Our experiments therefore include additional types of attacks that mimic the patterns of normal users, particularly main page attacks and dominant page attacks. A main page attack repeatedly requests the main page, which is the page most frequently requested by users; a dominant page attack randomly requests any of the pages frequently requested by most users.

For the App-DDoS attack datasets, we initiated App-DDoS attacks on a web server featured in other studies [4,5,8]. The attack traffic was generated in a network of the Korea Advanced Institute of Science and Technology (KAIST) via a modification of *black energy*, which is a well-known DDoS attack tool. We constructed copies of websites in the KAIST network as target systems of the attacks because of security problems. During the course of the App-DDoS attacks, we collected web logs from the web server and extracted web page request sequences. The collected sequences include 9,492 main page attacks, 14,038 random page attacks, and 9,489 dominant page attacks.

Table 1 Characteristics of sequence datasets

Dataset	Type of site	#Seq.	Avg. seq.	Characteristics
Dataset1	Education	30140	13.03	Requests for mostly dominant pages
Dataset2	Community	26727	10.6	A lot of requests for the same page
Dataset3	Online shopping	99018	22.7	Links to most of pages Requests for a variety of pages

4.3. Results and analysis

To construct our App-DDoS defense model, we first projected the proposed attribute matrix into a subspace spanned by the initial principal components, $\tilde{U}$, because it was highly sparse matrix. Then, we partitioned the data into several groups by k-means clustering based on the initial PCA coefficients. The parameter k was initially set to a sufficiently large number (k = 100) and clusters were merged if similar. Then, we constructed the multiple PCA models. We used seven principal components with the consideration of complexity and overfitting problems [15]; however, it could be adjusted as needed.

In this section, the performance of App-DDoS defense methods is illustrated with several sets of results. We first confirm whether the proposed attributes give good information to discriminate App-DDoS attacks or not. Figure 4 shows the results of the measurement for α_i, τ_i, and γ_i. In this figure, the measured values for normal traffic are shown in first row, and those of attacks are shown in second row. In Figure 5, the values for the h_i are also plotted. It is noted that the dimension reduction is applied for visualizing a multi-dimensional vector, h_i. We can see the differences between normal traffic and attacks in term of these values. Hence, we regard these values as important attributes even though they discard sequence-ordered information.

The proposed method discriminates App-DDoS attacks based on the reconstruction errors of the PCA. Thus, we show the reconstruction errors of a model, k, as shown in Figure 6. Here, the dots and the circles indicate the reconstruction errors of normal traffic for training data and validation data, and the axes indicate those of attack traffic for test data. Since our principal components are obtained to represent only normal traffic, the reconstruction errors for normal traffic are naturally low. On the contrary, the reconstruction errors for attack traffic are much higher than those of normal traffic. It is noted that some high values of this figure are set to 70 so as to visualize them.

We compared our method with a sequence-order-based defense method on the standard metrics such as the detection rate (DR), false positive rate (FPR), and

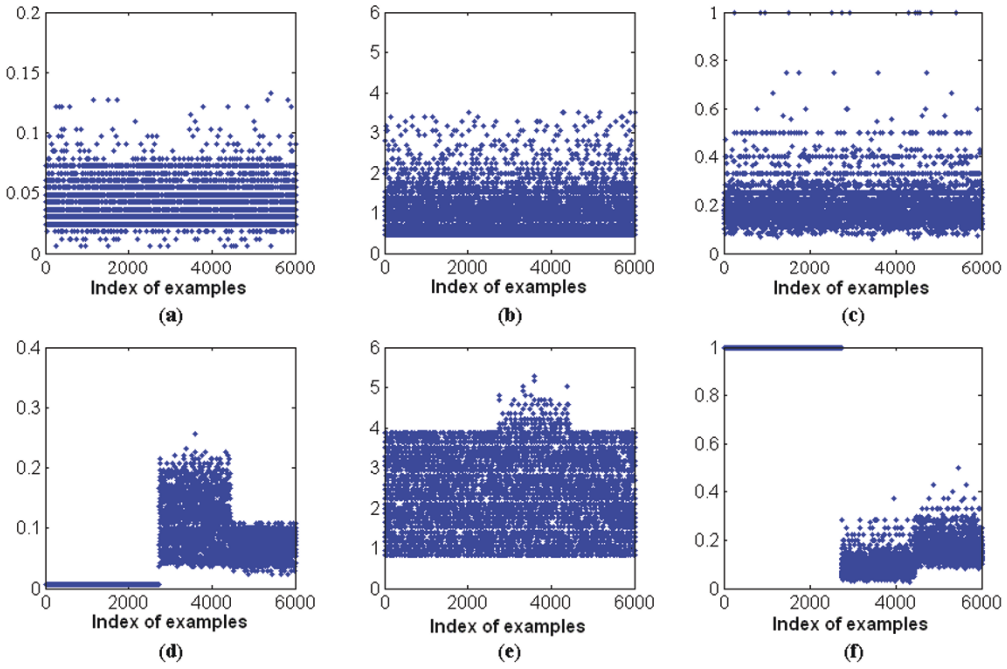

Figure 4 Values of the proposed attributes for normal and attack connections: the α_i (a), the τ_i (b) and the γ_i (c) of normal are shown in the first row; the α_i (d), the τ_i (e), and the γ_i (f) of attacks are shown in the second row.

receiver operating characteristic (ROC) curves. For that reason, we constructed a sequence-order-based defense method, which is modeled by the HsMM [8]. Webpage sequences for the HsMM were extracted from our weblogs. The best parameters were selected after experiments with a variety of parameters were conducted; initial parameters such as prior probabilities, transition matrix, and observation matrix were randomly set.

For the purpose of comparison, we first conducted experiments on various decision thresholds. The DR and the FPR were measured on each dataset and averaged. Table 2 shows the results of the experiment. As the threshold became large, the models yielded a relatively low DR and a low FPR, and vice versa. Our method tended to detect more App-DDoS attacks on all thresholds but its FPR was slightly higher than that of the HsMM. Figure 7 shows the ROC curves comparing our method and the HsMM. In this figure, we can see that the DR of our method is higher than that of the HsMM at most FPRs. When we carefully take into

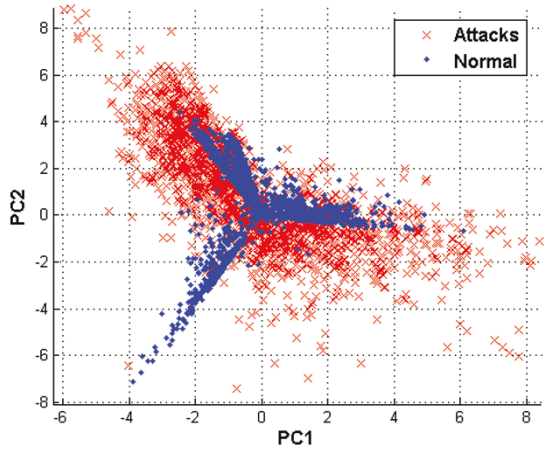

Figure 5 Plot of the h_i for normal and attack connections after dimension reduction.

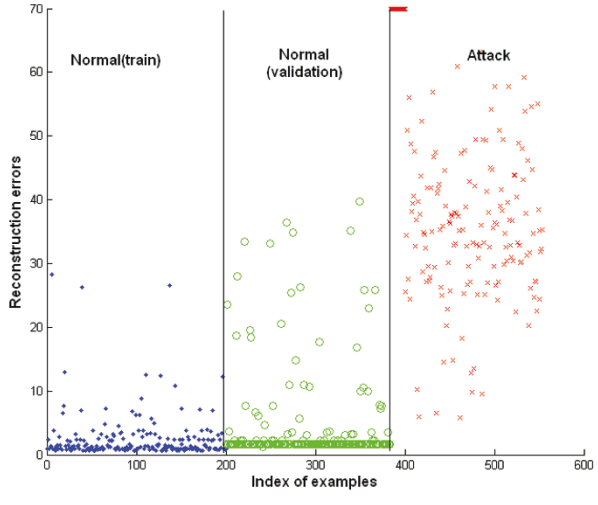

Figure 6 Plots of PCA coefficients in the model k.

Table 2 Averaged performances of the methods for various error thresholds

	Proposed		HsMM	
Threshold	DR(%)	FPR(%)	DR(%)	FPR(%)
$\mu + \sigma$	95.4	20.9	89.9	14.7
$\mu + 1.5\sigma$	94.3	11.8	83.8	7.8
$\mu + 2\sigma$	91.4	9.1	74.3	5.4
$\mu + 2.5\sigma$	86.7	4.5	64.3	3.4
$\mu + 3\sigma$	83.8	4.0	38.5	1.7
$\mu + 3.5\sigma$	77.0	3.2	33.7	0.5

account the DR, the FPR, and ROC curves, the threshold can be set between $\mu + 2\sigma$ and $\mu + 3\sigma$. This result is the same as that of studies based on outlier detection: the thresholds are usually set to two or three standard deviations from the mean [16]. In this experiment, our detection model has a DR of 86.7% and an FPR of 4.5% for a threshold of $\mu + 2.5\sigma$, and the HsMM has a DR of 74.3% and an FPR of 5.4% for a threshold of $\mu + 2\sigma$.

As previously mentioned, we performed experiments with three types of App-DDoS attacks. The DRs for each type of attack are shown in Table 3. Performance evaluations were conducted on each dataset because browsing behaviors can be different for different websites. As we expected, random page attacks tend to be detected well by defensive methods and the dominant page attacks tend to be detected less. We can see that our method outperforms the HsMM in most cases. The HsMM shows good performance for random page attacks but poor performance for main page and dominant page attacks. This means that random page attacks generate disordered sequences, while other attacks generate natural sequences by mimicking normal behaviors. In addition, as shown in dataset3 of this table, we can find that sequence orders can be broken in complicated

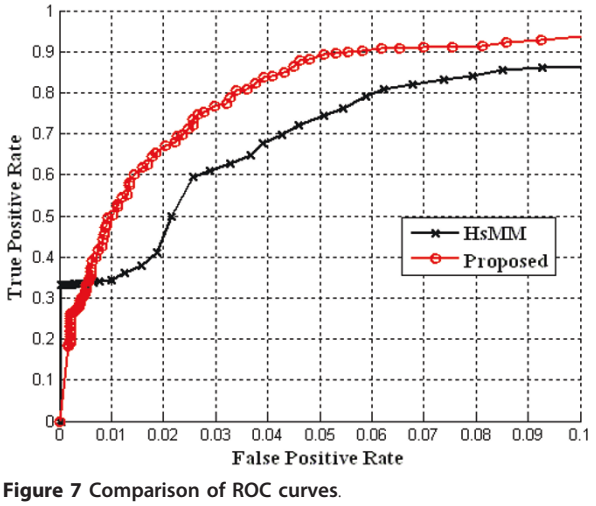

Figure 7 Comparison of ROC curves.

Table 3 DRs of the proposed method (Prop.) and the HsMM on various datasets

Attack type	Dataset1		Dataset2		Dataset3	
	Prop.	HsMM	Prop.	HsMM	Prop.	HsMM
Main page	92.3	85.5	83.7	48.3	90.5	84.2
Random page	95.5	98.1	97.2	99.8	82.9	68.7
Dominant page	86.1	65.5	82.5	67.2	69.5	51.6
Average	91.3	83.0	87.8	71.7	80.9	68.1

websites that have a lot of hyper links. On the other hand, our method can detect the attacks better than the HsMM. One of the reasons is that we utilize specific numerical values for detecting App-DDoS attacks and we employ the divide and conquer approach.

One of the advantages of our detection method is to enable additional information to be easily inserted into the model. For the purpose of adding attributes, our method can use any types of numerical attributes, while the HsMM requires only types of ordered data. Thus, network experts' opinions can easily be applied to our method for improving the detection performance. For example, it was difficult to detect main page attacks on the community website because the same page was also repeatedly requested by normal users. In this case, we can utilize information about the HTTP status codes and parameters of server scripts. Thus, we extracted this information from weblogs and simply attached two additional values to the attribute vector: $v_1 = e_i/L_i$ and $v_2 = p_i/L_i$, where the e_i is the number of requests whose status code includes an error (i.e., 404 Not Found) in a sequence i. The value p_i is the number of the pages that include the same script parameters (i.e., "id = 1") in a sequence i. Figure 8 shows the results of measurement for normal traffic and attacks. The main page attack includes three types of script parameter patterns: (1) no script parameter, (2) random script parameters, and (3) increasing script parameters. Adding new attributes helps improving the DR for the main page attacks. For main page attacks on dataset2, the DR is originally 49.1% but we can achieve a DR of 83.7% as shown in Table 3 after attaching two new attributes into the attribute vector.

4.4. Early warning system

The proposed method can be used as an early warning system. A detection method that utilizes sequences should determine the sequence length at which the algorithm is launched. The general principle for determining the sequence length is as follows: the shorter the sequence, the earlier the response–albeit with less reliability. Figure 9 shows the average reconstruction errors as the sequence length varies. The reconstruction error for legitimate users remains at a certain level whereas

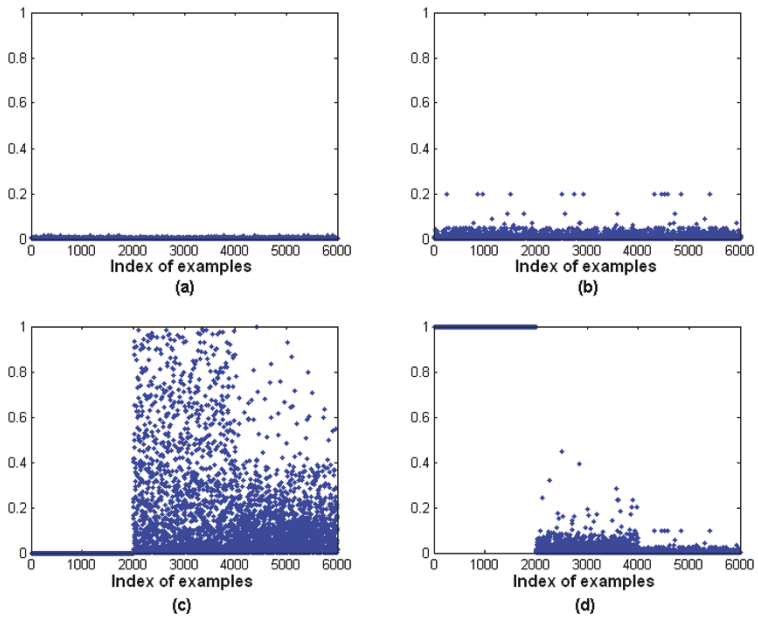

Figure 8 Plots of additional attributes: the v_1 (a) and the v_2 (b) of normal traffic are shown in the first row; the v_1 (c) and the v_2 (d) of attacks are shown in the second row.

the error for attacks becomes large as the sequence length varies from short to long. Therefore, if the DDoS detection model is launched with an appropriately short sequence length, the model can act as an early warning system. Although the proposed model may make incorrect decisions, it gradually refines its decisions as enough requests are obtained.

5. Conclusion

The focus of this article is on detecting App-DDoS attacks. We proposed a new model that utilizes sequence-order-independent attributes rather than the web page sequence order. The model consists of multiple PCAs so that complex browsing behaviors are given maximal consideration. Requiring only the weblog and the simplest of computations, the proposed method is practical and efficient for detecting App-DDoS attacks in real environments. To reliably validate our model, we generated three types of App-DDoS attacks. Our method detects App-DDoS attacks with an averaged DR of 86.7% and an averaged FPR of 4.5% when the error threshold is set at $\mu + 2.5\sigma$. These values demonstrate

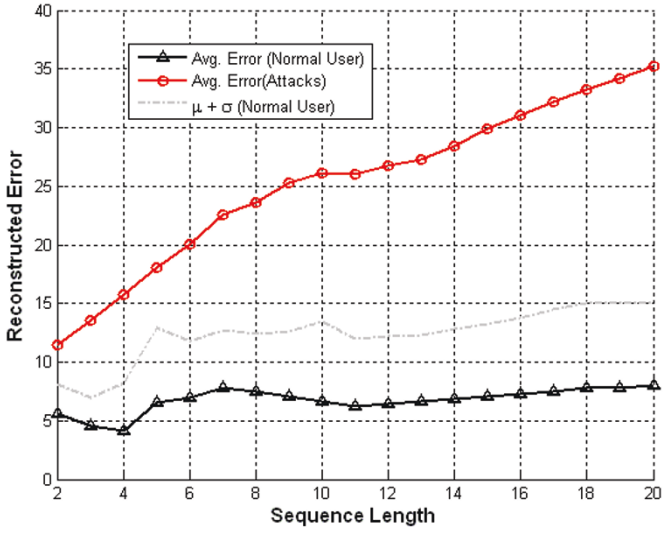

Figure 9 Reconstruction errors in relation to the sequence length.

that the proposed method can effectively describe a web user's browsing behavior and detect App-DDoS attacks. In addition, the proposed model has the capability of acting as an early warning system. Future research will focus on efficient updates and automatic learning algorithms, so that we can develop a defense method with online updates.

Abbreviations

App-DDoS: application-layer DDoS; DDoS: distributed denial of service; DR: detection rate; FPR: false positive rate; HsMM: hidden semi-Markov model; KAIST: Korea Advanced Institute of Science and Technology; PCA: principal component analysis; ROC: receiver operating characteristic.

Acknowledgements

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1031-0005).

Author details

[1]Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Republic of Korea
[2]Graduate School of Information Security, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Republic of Korea

Competing interests

The authors declare that they have no competing interests.

Received: 18 November 2010 Accepted: 1 August 2011
Published: 1 August 2011

References

1. Calculating the size of the downadup outbreak–F-secure weblog: news from the Lab. http://F-secure.com. Accessed 16 January 2009
2. Y Xie, S-Z Yu, Monitoring the application-layer DDoS sttacks for popular websites. IEEE/ACM Trans Netw. **17**(1), 15–25 (2009)
3. X Chen, J Heidemann, Flash crowd mitigation via adaptive admission control based on application-level observations. ACM Trans Internet Technol. **5**(3), 532–569 (2005). doi:10.1145/1084772.1084776
4. S Ranjan, R Swaminathan, M Uysal, A Nucci, E Knightly, DDoSshield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Trans Netw. **17**(1), 26–39 (2009)
5. T Yatagai, T Isohara, I Sasase, Detection of HTTP-GET flood attack based on analysis of page access behavior, in *Proceedings IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing*, art. no. 4313218 232–235. (2007).
6. A Shevtekar, N Ansari, Is it congestion or a DDoS attack? IEEE Commun Lett. **13**(7), 546–548 (2009)
7. S Kandula, D Katabi, M Jacob, AW Berger, Botz-4-Sale: surviving organized DDoS attacks that mimic flash crowds. NSDI'05, http://www.usenix.org/events/nsdi05/tech/kandula/kandula.pdf
8. Y Xie, S-Z Yu, A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM Trans Netw. **17**(1), 54–65 (2009)
9. S Ranjan, R Karrer, E Knightly, Wide area redirection of dynamic content by internet data centers, in *Proc IEEE INFOCOM*, 816–826, (2004)
10. S Amari, S Wu, Improving support vector machine classifiers by modifying kernel function. Neural Netw. **12**(6), 783–789 (1999). doi:10.1016/S0893-6080(99)00032-5
11. D Zhao, Formulating LLE using alignment technique. Pattern Recogn. **39**(11), 2233–2235 (2006). doi:10.1016/j.patcog.2006.05.007
12. JA Hartigan, MA Wong, A k-means clustering algorithm. J R Stat Soc Ser C (Appl Stat). **28**(1), 100–108 (1979)
13. J Hu, C Xiong, J Shu, X Zhou, J Zhu, A novel text clustering method based on TGSOM and fuzzy K-means. in *Proceedings of the 1st International Workshop on Education Technology and Computer Science, ETCS 2009*, **1**, art. no. 4958717, pp. 26–30
14. J Ji, TYT Chan, Q Zhao, Comparative advantage approach for sparse text data clustering, in *Proceedings - IEEE 9th International Conference on Computer and Information Technology, CIT 2009* **1**, art. no. 5329159, pp. 3–8.
15. RO Duda, PE Hart, DG Stork, *Pattern Classification* (Wiley, 2000)
16. S Lee, J Sung, D Kim, Incremental update of linear appearance models and its application to AAM: incremental AAM. Lecture Notes Comput Sci. **4633**, 538–547 (2007). doi:10.1007/978-3-540-74260-9_48

doi:10.1186/1687-1499-2011-50
Cite this article as: Lee *et al.*: Sequence-order-independent network profiling for detecting application layer DDoS attacks. *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:50.