Anomaly Detection for Application Layer User Browsing Behavior Based on Attributes and Features

Xiong Luo\textsuperscript{1,2}, Xiaoqiang Di\textsuperscript{1,2}, Xu Liu\textsuperscript{1,2}, Hui Qi\textsuperscript{1,2}, Jinqing Li\textsuperscript{1,2}, Ligang Cong\textsuperscript{1,2} and Huamin Yang\textsuperscript{1,2}

\textsuperscript{1} School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, 130022, China
\textsuperscript{2} Jilin Province Key Laboratory of Network and Information Security, Changchun, 130022, China
Email: dixiaoqiang@cust.edu.cn

Abstract. Application layer distributed denial of service (App-DDoS) attacks have posed a great threat to the security of the Internet. Since these attacks occur in the application layer, they can easily evade traditional network layer and transport layer detection methods. In this paper, we extract a group of user behavior attributes from our intercept program instead of web server logs and construct a behavior feature matrix based on nine user behavior features to characterize user behavior. Subsequently, principal component analysis (PCA) is applied to profile the user browsing behavior pattern in the feature matrix and outliers from the pattern are used to recognize normal users and attackers. Experiment results show that the proposed method is good to distinguish normal users and attackers. Finally, we implement three machine learning algorithms (K-means, DBSCAN and SVM) to further validate the effectiveness of the proposed attributes and features.

1. Introduction

Distributed denial of service (DDoS) attacks have seriously threatened the Internet service security, which consume server resources to prevent legitimate users from visiting the target system [1, 2] and are recognized as one of the most damaging attacks on the Internet security today [3].Conventionally, DDoS attacks occur in the network layer and aim to overwhelm the victim’s bandwidth. For instance, ICMP flooding, SYN flooding and UDP flooding [2, 4]. Since many methods have been proposed to detect network layer DDoS attacks, it is not easy for attackers to launch DDoS attacks in the network layer as before [5]. Therefore, attackers turn their attention to the application layer. In this layer, the connatural vulnerabilities of the Internet architecture provide opportunities for malicious users to launch various attacks.

Since application layer distributed denial of service (App-DDoS) attacks are carried out in the application layer, they can easily escape the traditional detection methods based on network layer and transport layer [6]. Moreover, the victim server cannot distinguish which one is a normal request and which one is an abnormal request, because attackers utilize legitimate HTTP requests to access the target system [1]. In order to solve this problem, numerous methods have been proposed. To the best of our knowledge, the existing studies about App-DDoS attacks detection can be roughly divided into two categories: (1) analysis of user behavior differences between normal users and attackers based on web server logs [5, 7, 8, 9] and (2) detection of App-DDoS attacks during a flash crowd event based on aggregate-traffic analysis [2, 10, 11].
In this paper, we propose an App-DDoS attacks detection method for user browsing behavior based on attributes and features. First, we extract a group of user behavior attributes and construct a user behavior feature matrix based on nine user behavior features to characterize user browsing behavior. Next, we apply PCA to profile the potential user behavior pattern and outliers from the pattern are used to identify anomaly users. In addition, for extracting user behavior attributes and computing user behavior features, we design and develop an application layer intercept program which is based on Java Platform Enterprise Edition (J2EE) framework to collect user access data instead of collecting them from web server logs. The advantage of this is that we can get more detailed access data such as the number of request parameters. What’s more, we do not have to consider the redundant records in web logs which are caused by the embedded objects in web pages.

The contributions of our work in this paper are as follows: (1) we extract a group of user behavior attributes and construct nine user behavior features; (2) we design and develop an intercept program to collect data; (3) we validate the effectiveness of our scheme on real data from a dynamic government website.

The rest of the paper is organized as follows. In section 2, we discuss the related work on our research. Section 3 describes the process of data collection, attribute extraction, feature construction and the feature matrix construction. In Section 4, we introduce our detection method. Experimental results and analysis are shown in Section 5. Finally, we conclude our work in Section 6.

2. Related Work

Last decade, many detection methods for App-DDoS attacks have been put forward. In [2, 5, 12, 13], Xie Y et al. use Hidden Semi Markov Model (HsMM) to predict user browsing behaviour, and the user request sequence is regarded as a criterion to distinguish normal users and attackers. They assume that normal users visit web pages in some sequential orders, which is different from attackers. Therefore, they utilize the extended HsMM to predict whether the visitor is a malicious user based on the request sequence. Nevertheless, this assumption cannot be always true and for some extremely large popular websites like www.tmall.com and www.sina.com, the complexity of HsMM algorithm is very high.

Sequence-order-independent is considered by Lee S et al. [7]. They think the request sequence orders of web users may be more harmful than helpful in the profiling of browsing behavior, because different persons have different browsing habits. They extract four sequence-order-independent attributes to characterize user browsing behavior. However, abstracting these four attributes which reflect the document popularity needs to know the total number of web pages of a website in advance. This is a very difficult work for those enormous websites.

To avoid the above problems, many scholars put their attention to the distinctions of user browsing behavior between normal users and attackers [1, 8, 9]. Najafabadi M M et al. [1] extract URL, IP address and request time to model user browsing behavior. However, they only focus on the HTTP GET attacks with flooding behavior. Yadav S et al. [8] extract 8 features from web server logs and construct other 9 features to better detect App-DDoS attacks. But the feature Src_IP_Addr is likely to be invalid, because attackers could use the IP address deception technology to evade the anomaly detection.

With the same idea, a series of user behavior features are extracted from web server logs by Liao Qin et al. [9]. And for achieving better test performance, they also construct other features based on the extracted features. Then, three classical data mining classification algorithms, which are Naïve Bayes Classifier (NBC), Radial basis Function (RBF) and C4.5, are used to verify the effectiveness of their method. Nevertheless, these features are not always suitable for differentiating normal users and attackers. For example, the feature diffReqTimes means times of different requests sent by a user in a single session. They think a normal user will not repeatedly visit the same page, but an attacker always initiates repeated requests. When the user’s network is bad, he or she maybe refresh the current page all the time, then the user will be deemed as an attacker.

As the same as the most of previous works, we also put attention to user browsing behavior. The difference of our research is that we collect user browsing data from our intercept program, which includes more detailed information than web server logs. Besides, we extract a group of more
important user behavior attributes and construct nine features based on the extracted attributes to form the user behavior feature matrix.

3. Attribute Extraction and Feature Construction

In this section, we first introduce data collection and session identification. Then we detailly describe the process of extracting user behavior attributes, constructing user behavior features and constructing the user behavior feature matrix. Figure 1 shows the whole procedure of data collection and pre-processing, which contains data collection, session identification, attribute extraction, feature construction and feature matrix construction.

![Figure 1. Procedure of data collection and pre-process.](image)

### 3.1. Data Collection and Session Identification

Data collection is a very important and critical link, which determines the correctness of our work. We capture user request and server response data from the intercept program instead of collecting them from web server logs. Because web server logs include much redundant information and cannot completely record the user request information. For example, the web server logs only include the length of request content but not include the number of request parameters.

Then, session identification is performed based on the session identifier (SESSIONID) to recognize who initiate the request. SESSIONID is the unique identification of a user session. The purpose of this is to map HTTP requests and HTTP responses to the corresponding user session for extracting user behavior attributes.

### 3.2. User Behavior Attribute Extraction

User behavior has been used to represent the potential browsing pattern of a web user [8, 9] and it reflects the user’s habit of accessing the system. For this, we extract eleven behavior attributes to characterize user browsing behavior. Table 1 illustrates the eleven attributes of a single user session.

| attribute | description               |
|-----------|---------------------------|
| un        | the number of user requests without login |
| ft        | the first user request time |
| lt        | the last user request time  |
| t         | the time of user browsing web server |
| dq        | duration between two continuous requests |
| c         | the total number of user requests |
| a         | the number of user requests invoking an action |
| m         | the number of user requests invoking a method |
| s         | the HTTP status codes of user requests |
| pa        | the number of parameters in a request |
| pg        | the number of user browsing a page |
3.3. User Behavior Feature and Matrix Construction

For detecting App-DDoS attacks, nine user behavior features are constructed based on the above eleven user behavior attributes and the detailed descriptions of them are as follows.

\[
\alpha = t = |lt - ft| \tag{1}
\]

Where \(\alpha\) represents the user browsing time. The browsing time of normal users should be within a reasonable range.

\[
\beta = un \tag{2}
\]

Where \(\beta\) indicates the number of unsigned requests in a session. For those internal websites, this feature is very important. Because only the logged user can browse the system.

\[
\gamma = \left[ \frac{1}{n} \sum_{j=1}^{n} \left( dq_j - \frac{1}{n} \sum_{j=1}^{n} dq_j \right)^2 \right]^{1/2} \tag{3}
\]

Where \(n\) is equal to \(c-1\) and \(j\) means the \(j\)th request. \(\gamma\) is a digital feature calculated as the standard deviation of the attribute \(dq\) in a session. This feature is useful to find abnormal requests, which is owing to that attackers usually send requests by malicious software, regularly.

\[
\delta = \left[ \frac{1}{c} \sum_{j=1}^{c} \left( pa_j - \frac{1}{c} \sum_{j=1}^{c} pa_j \right)^2 \right]^{1/2} \tag{4}
\]

Where \(\delta\) is a numerical feature calculated as a standard deviation of the attribute \(pa\). For attackers, it is very difficult to exactly simulate the request parameters.

\[
\mu = \sum_{i=1}^{c} (s4 + s5) \frac{1}{c} \tag{5}
\]

Where \(s4\) represents the HTTP status code 4xx and \(s5\) represents the HTTP status code 5xx. Feature \(\mu\) means the ratio of the number of abnormal requests to the number of all requests. Ordinarily, legitimate user requests do not return the 4xx and 5xx status code. Thus, this is a significant feature to identify spiteful requests.

\[
\rho = \max(a) \frac{1}{c} \tag{6}
\]

Where \(\rho\) is a numerical feature calculated as the ratio of the max \(a\) to \(c\). Normal users usually utilize different functions to finish their tasks. However, attackers may only repeatedly invoke one or several functions. Hence, attackers’ \(\rho\) are higher than normal users.

\[
\sigma = \left[ \frac{1}{N} \sum_{i=1}^{N} \left( a_i - \frac{1}{N} \sum_{i=1}^{N} a_i \right)^2 \right]^{1/2} \tag{7}
\]

Where \(N\) is the number of actions invoked by users. \(i\) represents NO. of the action and \(a_i\) represents the number of invoking the action. This feature reflects the case of calling actions in another way.

\[
\tau = \max(pg) \frac{1}{c} \tag{8}
\]

Where \(\tau\) is the ratio of the max \(pg\) to \(c\). Generally, a normal user will not repeatedly browse the same page too many times. Therefore, for single URL repeated attack or multiple URL repeated attack, it is efficient to identify them.

\[
\varphi = \max(m) \frac{1}{c} \tag{9}
\]

Where \(\varphi\) is the ratio of the max \(m\) to \(c\). Though attackers can mimic the user browsing behavior between different pages to evade detection, it is very difficult for them to imitate the methods in the pages.

As mentioned above, each user is characterized by nine features, which are good enough to detect App-DDoS attacks for a dynamic web application. For this, let us assume that \(N\) users access a
website. We define a user behavior feature vector \( b = [\alpha, \beta, \gamma, \delta, \mu, \rho, \sigma, \tau, \varphi] \), and construct a user behavior feature matrix \( F = [b_1 \ldots b_N]^T \).

4. Detection Method

4.1. Principal Component Analysis
PCA is a multivariate statistical analysis technique, which is a way of identifying inner feature patterns in high dimensional data and expressing the data in such a way to find the similarities and differences of them [14]. The standard transformation process of PCA to get the principal components of high dimensional data is as follows:
1. subtracting the mean;
2. calculating the covariance matrix;
3. calculating the eigenvectors and eigenvalues of the covariance matrix;
4. choosing components and forming a feature vector;
5. deriving the new data set.

The greatest advantage of PCA is that it can be used to reduce the number of dimensions of the data and hold the main information at the same time. This is achieved by preserving low order principal components and ignoring high order principal components. These low order principal components usually retain the most important aspects of data and reflect the original characteristics of data. Therefore, PCA has been applied in face recognition, image compression and anomaly detection.

4.2. Anomaly User Behavior Detection Based on PCA
Because PCA is a non-parametric method and better suitable for explaining the given data by reducing it to a lower dimension matrix, we utilize PCA to profile user browsing behavior and reveal the hidden behavior pattern \( S \) in the feature matrix \( F \). Where \( S \) can be calculated through solving Eq. (10).

\[
\min \| F - S \|_F \tag{10}
\]

Let \( A = F - S \), so \( \| A \|_F \) represents the Frobenius norm of the matrix \( A \). Then we use singular value decomposition (SVD) [8, 15] to compute the principal component \( PC \) of \( X \), which denotes the covariance matrix of the feature matrix \( F \). After that, \( S \) can be defined as Eq. (11).

\[
S = PC^T \times F \tag{11}
\]

For judging whether the given user session is normal, we define a threshold \( \xi \) based on central limit theorem, as solved in Eq. (12).

\[
\xi = \mu[\varepsilon] + k\sqrt{\mu[\varepsilon - \mu[\varepsilon]]^2} \tag{12}
\]

Where \( \mu[\varepsilon] \) and \( \mu[\varepsilon - \mu[\varepsilon]]^2 \) are the mean and variance of the L2 norm of the vector in the matrix \( A \), respectively. The value \( k \) is a positive weight parameter for the threshold range. We use \( \xi_t \) to express the L2 norm of the vector of the \( i \)th user in the matrix \( A \), which we call outlier. Our key idea is that outliers between user browsing pattern \( S \) and original data \( F \) of attackers are higher than those of normal users.

When a new user \( t \) browses the target system, we capture user requests and record access data. After this, we extract the user behavior attributes described in Table 1 and compute the behavior features by Eq. (1)-(9). Then, we construct the behavior feature vector based on the nine features and add it into the normal sample matrix \( F \). Next, we decompose the feature matrix \( F \) and calculate the threshold \( \xi \) and the outlier \( \xi_t \). Finally, comparing \( \xi_t \) with \( \xi \), if \( \xi_t > \xi \), our detection model thinks the user is an attacker. Figure 2 shows the procedure of our detection method.
Figure 2. Procedure of detection method.

5. Experiment Results and Analysis

5.1. Normal Dataset
We evaluate our detection method on a real dataset of a dynamic government website whose architecture is based on J2EE to provide the service of project management. First of all, 96-hour HTTP requests and HTTP responses were captured by our intercept program. Then 74411 records were extracted from the captured data. Finally, 2551 useable user sessions were identified from the HTTP requests and HTTP responses.

5.2. Attack Dataset
As similar as the majority of previous works [1, 5, 8, 9], we synthesized the App-DDoS attacks dataset in an isolate network to validate our approach. Here, we chose JMeter [16] to simulate App-DDoS attacks instead of existing DDoS attack tools, because JMeter is a load benchmark tool for testing web applications and is open source software. Besides, JMeter allows concurrent sampling by many threads and simultaneous sampling of different dynamic web functions by separate thread groups. In this paper, every thread launched by JMeter is regarded as an individual attack session. Initially, we started 200 threads to imitate different attack scenarios by four thread groups and every thread was circulated many times. In order to identify the data generated by different threads and map them to different sessions, cookie management was used to ensure the requests of a thread have the same cookie. In the meantime, attack data was captured by the intercept program.

5.3. Results and Analysis
In order to validate that our attributes and features are good to detect App-DDoS attacks, we first compare the user behavior attributes and the user behavior features between normal users and attackers. Figure 3 shows user request time interval (URTI) cumulative distribution function (CDF) of normal users and attackers. CDF is the integral of the probability density function and it can be used to describe the probability distribution of URTI, so we can find that the URTIs of normal users and attackers are different and the URTIs of attackers are very similar with each other. This is because most App-DDoS attacks are launched through automatic computer programs or malicious software. In Figure 4, $\gamma(a), \rho(b), \varphi(c)$ of normal sessions are displayed in the first row and $\gamma(d), \rho(e), \varphi(f)$ of attack sessions are displayed in the second row. We can see that the normal sessions differ from the attack sessions in term of feature values which have been normalized.
Figure 3. CDF for user request time interval.

Figure 4. Comparison of normal and attack sessions: the $\gamma(a)$, $\rho(b)$, $\varphi(c)$ of normal sessions are displayed in the first row; the $\gamma(d)$, $\rho(e)$, $\varphi(f)$ of attack sessions are displayed in the second row.

Figure 5. ROC curve for PCA detection method.
Figure 5 is the receiver operating characteristics (ROC) curve of our scheme, which shows the performance of our anomaly detection method on App-DDoS. We can see that when the FPR is more than 0.2%, the TPR will reach 100%. In figure 6, we plot the outliers of normal sessions and attack sessions, which are indicated by circles and asterisks, respectively. It is obvious that the outliers of attack sessions are higher than those of normal sessions.

![Figure 6. Outliers from the user browsing pattern.](image)

Table 2 gives the specific experiment results of PCA with different detection thresholds. When we set the threshold between 1.7293 and 3.4477, the DR is more than 95%, the FPR is less than 2%. And when the detection level is set to $\theta + \pi$, the threshold is 2.5885, we can get the best detection result. The DR is 100% and the FPR is 0.25%. This indicates that our detection method with the nine user behavior features is good to detect App-DDoS attacks.

To further verify the effectiveness of the proposed user behavior attributes and user behavior features, we carry out three machine learning algorithms (K-means, DBSCAN and SVM) on our datasets. The detection results are presented in table 3. We can see that the detection results of K-means and SVM are very close to the detection result of PCA. The detection result of DBSCAN is not as good as PCA, but the DR of DBSCAN is also more than 90%.

| Detection level | Detection threshold | DR  | FPR  |
|-----------------|---------------------|-----|------|
| $\theta + 0.5\pi$ | 1.7293              | 1   | 0.0130 |
| $\theta + \pi$   | 2.5885              | 1   | 0.0025 |
| $\theta + 1.5\pi$| 3.4477              | 0.9550 | 0  |
| $\theta + 2\pi$  | 4.3070              | 0.8050 | 0  |
| $\theta + 2.5\pi$| 5.1662              | 0.6550 | 0  |

| Detection algorithm | DR  | FPR  |
|---------------------|-----|------|
| K-means             | 1   | 0.0020 |
| DBSCAN              | 0.9400 | 0    |
| SVM                 | 1   | 0.0028 |

6. Conclusion
Application layer DDoS attacks detection is of great importance for the web application and the server provider. In this paper, we first extract eleven behavior attributes through analyzing the user browsing behavior in the application layer. Then we construct nine behavior features to characterize user
browsing pattern and construct a user behavior feature matrix based on them. We introduce our
detection method based on PCA in detail and verify it on a real dataset. When the detection level is set
to $\theta + \pi$, our detection scheme achieves a good performance, the DR is 100% and the FPR is 0.25%.
Finally, we implement three machine learning algorithms (K-means, DBSCAN and SVM) on our
datasets. Experimental results further prove that the proposed attributes and features are good to
distinguish normal users and attackers.

7. Acknowledgement
This work is supported in part by the Science and Technology Development Project (20150204081GX)
and Science and Technology Project (JJKH20170628KJ) of Jilin Province of China.

8. References
[1] Najafabadi M M, Khoshgoftaar T M, Calvert C, et al. User Behavior Anomaly Detection for
Application Layer DDoS Attacks[C] IEEE International Conference on Information Reuse and
Integration. IEEE Computer Society, 2017:154 -161.
[2] Xie Y, Yu S Z. Monitoring the Application-Layer DDoS Attacks for Popular Websites [J].
IEEE/ACM Transactions on Networking, 2009, 17(1):15-25.
[3] Stevanovic D, Vlajic N, An A. Detection of malicious and non-malicious website visitors using
unsupervised neural network learning [J]. Applied Soft Computing, 2013, 13(1):698-708.
[4] Zargar S T, Joshi J, Tipper D. A Survey of Defense Mechanisms against Distributed Denial of
Service (DDoS) Flooding Attacks [J]. IEEE Communications Surveys & Tutorials, 2013,
15(4):2046-2069.
[5] Xie Y, Yu S Z. A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User
Browsing Behaviors [J]. IEEE/ACM Transactions on Networking, 2009, 17(1): 54- 65.
[6] Mantas G, Stakhanova N, Gonzalez H, et al. Application-layer denial of service attacks: taxonomy
and survey [J]. International Journal of Information & Computer Security, 2015, 7(2/3/4):216-
239.
[7] Lee S, Kim G, Kim S. Sequence-order-independent network profiling for detecting application
layer DDoS attacks [J]. Eurasip Journal on Wireless Communications & Networking, 2011,
2011(1): 50.
[8] Yadav S, Selvakumar S. Detection of application layer DDoS attack by modeling user behavior
using logistic regression [C] International Conference on Reliability, INFOCOM Technologies
and Optimization. IEEE, 2015:1- 6.
[9] Liao Q, Li H, Kang S, et al. Feature extraction and construction of application layer DDoS attack
based on user behavior[C] Control Conference. IEEE, 2014: 5492-5497.
[10] Yu S, Zhou W, Jia W, et al. Discriminating DDoS Attacks from Flash Crowds Using Flow
Correlation Coefficient [J]. IEEE Transactions on Parallel & Distributed Systems, 2012,
23(6):1073-1080.
[11] Oikonomou G, Mirkovic J. Modeling Human Behavior for Defense against Flash-Crowd
Attacks[C] IEEE International Conference on Communications. IEEE, 2009:1- 6.
[12] Xie Y, Tang S. Online Anomaly Detection Based on Web Usage Mining[C] Parallel and
Distributed Processing Symposium Workshops & Phd Forum. IEEE, 2012:1177-1182.
[13] Xie Y, Tang S, Xiang Y, et al. Resisting web proxy-based http attacks by temporal and spatial
locality behavior [J]. IEEE transactions on parallel and distributed systems, 2013, 24(7): 1401-
1410.
[14] Smith L I. A Tutorial on Principal Components Analysis [J]. Information Fusion, 2002, 51(3): 52.
[15] Abdi H, Williams L J. Principal component analysis [J]. Wiley interdisciplinary reviews:
computational statistics, 2010, 2(4): 433- 459.
[16] JMeter, A., https://jmeter.apache.org/