APPLICATION-LAYER DDOS DETECTION BASED ON A ONE-CLASS SUPPORT VECTOR MACHINE

Chuyu She[1, 2, 3], Wushao Wen[1,2], Zaihua Lin[1], and Kesong Zheng[1]

[1]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, PR China
[2]SYSU-CMU Shunde International Joint Research Institute, Shunde 528300, PR China
[3]School of Mathematics and Statistics, Guangdong University of Finance & Economics, Guangzhou 510320, PR China

ABSTRACT

Application-layer Distributed Denial-of-Service (DDoS) attack takes advantage of the complexity and diversity of network protocols and services. This kind of attacks is more difficult to prevent than other kinds of DDoS attacks. This paper introduces a novel detection mechanism for application-layer DDoS attack based on a One-Class Support Vector Machine (OC-SVM). Support vector machine (SVM) is a relatively new machine learning technique based on statistics. OC-SVM is a special variant of the SVM and since only the normal data is required for training, it is effective for detection of application-layer DDoS attack. In this detection strategy, we first extract 7 features from normal users' sessions. Then, we build normal users' browsing models by using OC-SVM. Finally, we use these models to detect application-layer DDoS attacks. Numerical results based on simulation experiments demonstrate the efficacy of our detection method.

KEYWORDS

Application-layer DDoS attack, One-Class Support Vector Machine, machine learning, feature, browsing model

1. INTRODUCTION

Denial of Service (DoS) attack is a significant security challenge on the Internet. The ease of conducting DoS attack comes from the existing limitations on the Internet protocols such as TCP, UDP and the ready availability of attack tools [1]. In traditional DoS attacks, an attacker uses only one node. However, attackers nowadays can control multiple nodes to launch attacks. These attacks are classified as Distributed Denial-of-Service (DDoS) attacks [2]. DDoS attacks may be carried out at the network layer. These kinds of attacks mainly exploit vulnerabilities of protocols of network layer [3]. However, with the progress of defence method, the DDoS attackers are gradually targeting the application layer. Application-layer DDoS attacks are more complicated to detect. Lots of methods have been proposed to defend against DDoS, such as statistical approaches, algorithms based on signatures and so on. Statistical approaches take account of packet attributes such as source IP and destination IP address, time to live (TTL), and so on. These methods often assume that the distribution of network traffic characteristics will change when a DDoS attack happens. Actually, these methods are effective in protecting DDoS attack at IP or TCP layers. However, they can't distinguish the application-layer DDoS attack packets from normal packets because they lack the analysis capability of application layers. Algorithms based on signature may detect attacks on any communication layer. However, they can detect known attacks only.

DOI: 10.5121/ijnsa.2017.9102

Literature on detecting application-layer DDoS attacks can be also found. Ramamoorthi et al.[4] proposed an anomaly detection mechanism to detect DDoS attacks by using Enhanced Support Vector Machine (ESVM) with string kernels. SVM is a relatively new machine learning technique based on statistics. Due to its excellent generalization ability compared with the traditional intelligent methods, e.g. neural network, SVM has been famous and popular in many areas [5]. However, the primal SVM only focuses on two class classification problem. When the problem is imbalance, the performance deteriorates [6].

In this paper, we use One-Class Support Vector Machine (OC-SVM) to detect application-layer DDoS attack. OC-SVM is a special variant of the SVM. OC-SVM which is proposed to deal with the unbalanced problems of classification, intends to find the smallest hyper-sphere containing the positive data. As for the test point, OC-SVM only judges it whether the test point belongs to that class [7]. Since only the normal data is required for training, it is effective for detection of application-layer DDoS. Our contributions in this paper are summarized as:

1) Seven features are proposed and extracted from users' sessions based on the differences between attack users and normal users.

2) A normal user's browsing model is built by OC-SVM to detect application layer DDoS attack.

3) Numerical results demonstrate the effectiveness of the algorithm based on a real website.

The rest of this paper is organized as following. In Section 2, we describe the related work of our research. In Section 3, we introduce our detection model and algorithm. In Section 4, we validate the efficiency of our detection method. We conclude our work in Section 5.

2. RELATED WORK

Most studies about DDoS focus on IP layer or TCP layer [8]. Numerous variables such as IP address, port, or TCP flags can be analysed to detect Net-DDoS [9]. Lakhina et al. [10] use traffic feature distributions to detect anomalies. Xue et al. [11] present the histogram of the maxima of bounded traffic rate on an interval-byinterval basis as a traffic feature for exhibiting abnormal variation of traffic under DDOS flood attacks. Lee et al. [12] cluster IP addresses and TCP and UDP ports on backbone routers to find DDoS attacks. Simmross et al. [13] proposed a method to detect two anomaly types, namely floods and flash-crowds in network traffic, based on a non-restricted α-stable first order model and statistical hypothesis testing. Stavros et al. [14] proposed a method for DDoS detection by using fuzzy estimators. Neural networks [15] and [16] detect DDoS attacks combine with machine learning.

Literature on detecting application-layer DDoS attacks can be also found. Some researchers have used "puzzle" or identification codes that need users' interaction, which may disturb users. Park et al. [17] proposed that web servers can reply with probe scripts which detect whether there are mouse motion and other normal user's behaviours. Yu et al. [18] discriminated DDoS attack flow from flash crowd by the similarity of flows. Since attack flows are generated by a fixed program, attack flows are similar. Oikonomous et al. [19] found that request dynamics, request semantics and ability to process visual cues can be used to differentiate humans and attack agents. Many studies on web user behaviour mining formed four types. Type I is probabilistic model. For example, Brklen et al. [20] characterized page jump-probability with Zipf-like distribution, set double Pareto distribution to model the link-choice, and used log-normal distribution to capture revisiting behaviour. Type II is click-streams and web content mining, e.g., Velsquez et al. [21] used click-streams dataset and page content to mining user's usage patterns. Type III is a kind that detectors monitor user's behaviour such as system-calls by user's processes. Type IV is

based on classical pattern recognition model—Markov model. Xie et al. [3] used Hidden semi-Markov Model to capture users browsing behaviours.

3. DETECTION MODEL AND ALGORITHM

In this section, we introduce OC-SVM algorithm to detect application-layer DDoS based on 7 features (CMBF) we selected.

3.1. Detection Algorithm

We use OC-SVM algorithm to train normal sessions and then build normal users' behaviour models. And new sessions that deviate from the normal users' behaviour models are identified as anomalies. OC-SVM which is proposed to deal with the problems of classification, intends to find the smallest hyper-sphere containing the positive data. OC-SVM is to implicitly map the data points from the input space to the feature space by means of a nonlinear kernel function [22]. After being mapped into the feature space, the training data will be treated as belonging to one class [23]. Then a boundary is found in the feature space that separates the test points into normal and anomalous measurements.

Now, we introduce the detail of our detection algorithms.

Before running the OC-SVM, dataset should be pre-processed. OC-SVM algorithm needs a data structure, called data matrix.

$$X = \begin{bmatrix} X_{11} & \cdots & X_{1k} & \cdots & X_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ X_{j1} & \cdots & X_{jk} & \cdots & X_{jp} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ X_{n1} & \cdots & X_{nk} & \cdots & X_{np} \end{bmatrix} \tag{1}$$

The data matrix denotes the normal data set. It is a $n \times p$ matrix, which n means it has n objects, and p means objects have p features. So we collect normal data and select features first.

3.1.1 Generate Sessions of Users

For a user, his or her browsing behaviour is a request sequence. We assume that one client IP, two consecutive requests are less than m (default is set as 1800) seconds away are treated as in the same session [24].

According to the definition above, we consider $S_u = \{IP_u, \langle r_1, t_1 \rangle, \cdots, \langle r_n, t_n \rangle\}$ as a session describing the interaction between user u and web server, where $t_j - t_{j-1} \leq 1800s$.

3.1.2 Feature Selection

There are many differences between normal users' sessions and attackers' sessions, such as request rate, requested resources, request order, session time and so on. Based on these differences, we can select features from sessions. Before selecting session features, several definitions need to be introduced.

1) Resource Popularity

   Resources have different access frequency in a website. The word resource mentioned here has a flexible meaning: it can be a webpage, a picture, a video, a voice or a text, etc. Jung et al. [25] said that 10% of webpages account for approximately 80-90% of requests. They

proved that resources popularity follow Zipf-like distribution. So we define popularity of resource i , as following:

$$POP_i = \frac{AC_i}{AC_{all}} \quad (2)$$

Here, AC_i is the number of the resource i accessed in a period of time, while AC_{all} is the number of all the resources accessed in this period of time. Resource A's popularity is higher than B's, if A's access frequency is more than B's.

2) Transition Probability

On a website, the transition probability between every two resources is different. The following formula shows how to calculate this probability.

$$P_{ij} = \frac{Trans_{i \to j}}{Trans_{i \to all}} \quad (3)$$

$Trans_{i \to j}$ means transition times from resource i to j , and $Trans_{i \to all}$ is transition times from i to others.

3) History Transition Matrix

This matrix records transition times from one resource to any other resource, defining as follow:

$$M_{trans} = \begin{bmatrix} t_{11} & \cdots & t_{1i} & \cdots & t_{1n} \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ t_{i1} & \cdots & t_{ii} & \cdots & t_{in} \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ t_{n1} & \cdots & t_{ni} & \cdots & t_{nn} \end{bmatrix} \quad (4)$$

Here, t_{ij} is a transition time from resource i to resource j .

As concepts and definitions have been introduced above, the following features are selected for OC-SVM algorithm. And these features can be extracted from web server logs.

1. $N_{session}$: The total number of requests in a session.
2. Obviously, for HTTP flood application-layer DDoS, it's very important that a session should have enough requests for attack purpose. So we consider $N_{session}$ as a very important feature for detection of this kind of application-layer DDoS attack.
3. $POP_{session}$: The average popularity of all requests in a session.
4. $POP_{session}$ represents average popularity of all requests in a session. Some application-layer DDoS use random request sequence to launch an attack because it is very easy to implement. And random request attack is more complicated to detect than the attack of simply request one or some particular resources. As study [25] shows that most resources have relatively low popularity, the $POP_{session}$ of a session generated by random request attack is lower than normal users' sessions. So this character is proper for detection of random request attack.

5. $P_{session}$: The average transition probability of all adjacent requests in a session.

6. $P_{session}$ represents average transition probability of all adjacent requests in a session. For random request attack, its request sequence is randomly generated. Compared to normal users' sessions, generally, $P_{session}$ of attack session is lower than most normal sessions'. That is to say, this feature is very useful to distinguish normal users' session and random attack session.

7. $Size_{session}$: The total size of all request in a session.

8. An attacker can just request large resources such as a large video resource, which is a simple request but may cost a large volume of bandwidth. We consider $Size_{session}$ as session load. So session size is a very important feature for detection of attack focusing on large resources.

9. $D_{session}$: The duration of a session.

10. It's the time range from the first request to the last request in a session. Generally, normal users won't stay in a website for a long time. For stealthy attacks, it may take longer time to achieve attack effect. So this feature is useful for detection.

11. $replycode_n$: Reply code

12. Statistically, HTTP reply codes from web server have different frequencies. Generally, a better attack effect can be gained by more effective request. So attack session may have more reply code 200 than normal one, unless it simulates normal session reply code distribution, but this may weaken the attack effect in turn.

13. $dynamic$: The times of appearance of dynamic pages.

14. Normally, dynamic pages involve database query or insertion, computation of complicated scripts and other operations with heavy load. A little "smarter" attack tool will attack this vulnerability of a website. So this feature is useful for detection.

15. Now all the features above can be combined into a vector, like this:

    i. $$Features = (N_{session}, POP_{session}, \cdots, dynamic) \quad (5)$$

3.1.3 One-Class Support Vector Machine (OC-SVM)

As mentioned above, X represent training set consisting of n normal sessions. $\{x_i, i = 1,2,\cdots,n\}$,

$x_i \in R^p$, (p is the dimension of x_i).

Let $\Phi(x_i)$ denote the image of x_i in the feature space. For the classification problem of two categories, the data sets are not always linearly separable in the original space. Φ projects the original data sets into a higher dimensional space which called feature space and the non-separable data sets become linearly separable in this space [5]. The training instances are mapped into the feature space and separated from the origin by the hyperplane with the maximum margin [26]. In order to obtain the boundary, an optimization model is considered as follows:

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 + \frac{1}{nv}\sum_{i=1}^{n}\xi_i - \rho \\ &s.t \qquad w \bullet \phi(x_i) \geq \rho - \xi_i, \xi \geq 0 \end{aligned} \quad (6)$$

where n is the number of the data points; ξ_i is the non-negative slack variable of x_i; v is a regularization parameter and $v \in (0,1)$ controls the fraction of outliers. w and ρ are the parameters which determine the decision boundary.

Φ is a kind of mapping. Φ is inexplicit, however, the inner product of the image $\Phi(x_i)$ can be computed via kernel function. Such as linear kernel, polynomial kernel, radial basis function kernel and sigmoidal kernel [6]. These kernel functions are listed as:

1) Linear kernel: $K(x_i, x_j) = x_i^T x_j$ (7)

2) Polynomial kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + C)^p$ (8)

3) Radial basis function kernel: $K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2 / (2\sigma^2)\right)$ (9)

4) Sigmoidal kernel: $K(x_i, x_j) = \tanh(k x_i^T x_j - c)$ (10)

Where i and j range over $1,...n$; γ and c are constants; p is the degree of the polynomial and σ is the width of radial basis function kernel. In this paper, we selected radial basis function kernel as the kernel function $K(x_i,x_j)$ because the radial basis function kernel can approximate most kernel functions if the parameter σ is chosen appropriately [27].

In order to solve the optimization problem (6), Lagrange multipliers α_i, $\beta_i \geq 0$ are introduced for the constraints $w \cdot \Phi(x_i) - \rho + \xi_i \geq 0$ and $\xi_i \geq 0$. The Lagrange equation is formed as:

$$L(w, \xi, \rho, \partial, \beta) = \frac{1}{2}\|w\|^2 + \frac{1}{nV}\sum_{i=1}^{n}\xi_i - \rho - \sum_{i=1}^{n}\partial_i(w \bullet \phi(x_i) - \rho + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \quad (11)$$

In order to get the optimal value, the partial derivatives of the Lagrange equation (11) with respect to w, ξ and ρ are computed and set to zero.

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\partial_i \bullet \phi(x_i) = 0 \quad (12)$$

$$\frac{\partial L}{\partial \xi_i} = \frac{1}{nV} - \partial_i - \beta_i = 0 \quad (13)$$

$$\frac{\partial L}{\partial \rho} = \sum_{i=1}^{n}\partial_i - 1 = 0 \quad (14)$$

Then, we can obtain the following formulas:

$$w = \sum_{i=1}^{n}\partial_i \bullet \phi(x_i) \quad (15)$$

$$\partial_i = \frac{1}{nV} - \beta_i \quad (16)$$

$$\sum_{i=1}^{n}\partial_i = 1 \quad (17)$$

Substitute formulas (15), (16) and (17) into Lagrange equation (11), and its dual form is presented as follow:

$$\begin{aligned} &\min \partial^T Y \partial \\ &s.t\ 0 \le \partial_i \le \frac{1}{nV}, \sum_{i=1}^{n} \partial_i = 1 \end{aligned} \tag{18}$$

Where α = [α1, α2... αn] is the vector form of Lagrange multipliers for the constraints. Y is the kernel matrix of the training set, it can be expressed as follow:

$$Y_{ij} = k(x_i, x_j) = \phi(x_i) \bullet \phi(x_j) \tag{19}$$

The instances $\{x_i \mid \alpha_i > 0, i = 1... n\}$ are called support vectors. The function $f(x)$ is as follow:

$$f(x) = \operatorname{sgn}\left(\sum_{i \in SV} \partial_i K(x_i, x) - \rho \right) \tag{20}$$

Where SV is the indices of support vectors; ρ is computed by the following formula:

$$\rho = w^T \phi(x_i) = \sum_{j \in \{j \mid \partial_j \neq 0\}} \partial_j K(x_i, x_j) \tag{21}$$

For a new session x, $f(x)$ returns 1 if x is a normal session; $f(x)$ returns -1 if x is an abnormal session.

Algorithm 1 shows that the set of normal sessions are trained by OCSVM based on the selected features. Firstly, we extract features from the set of sessions. Secondly, we normalize the feature vectors. Finally, we use OC-SVM method to train the normal behavior model.

Algorithm 1 training the normal sessions

1: **Input**: Sessions
2: **Output**: boundary
3: **Method**:
4: Step1: extract features from the set of normal sessions.
5: Step2: normalize feature vectors and get the training set $\{x_i, i = 1, 2, ..., n\}$.
6: Step3: use OC-SVM algorithm to get the boundary of the training set.
7: (1)use the kernel function $K(x_i, x_j) = exp(-\|x_i - x_j\|^2/(2\sigma^2))$.
8: (2)solve the corresponding OC-SVM optimization problem.
9: (3)get the optimal boundary of the training set.

3.1.4. Detection Process

As the normal users' behavior models above have been built, these models can be used to detect anomaly of behavior. For a new session x, detection algorithm calculates the result base on the formula (20). If the result returns 1, the session will be recorded as normal; If the result returns 1, the session will be recorded as abnormal, and the corresponding user IP will be added to blacklist.

Algorithm 2 shows the detection process

Algorithm 2 Detection Process

```
1: Input: Sessions, boundary of the training set
2: Output: anomaly user IP
3: Method:
4:         i=0
5:         WHILE i < sessions.size
6:                 IF f(x_i) returns -1
7:                         normal=false
8:                 ENDIF
9:                 Else IF f(x_i) returns 1
10:                        normal=true
11:                        break
12:                ENDIF
13:                IF normal == false
14:                   discard current request
15:                   block this session[i].IP and add this IP to blacklist
16:                ENDIF
17:                increase i by 1 to calculate next session
18:        ENDWHILE
```

3.2. Detection Architecture

This paper trains normal sessions and build normal users' behaviour models by OC-SVM. As we have built the normal users' behaviour models, we can use these models to detect anomaly of behaviour. Fig.1 shows the detection architecture. Firstly, we select some features according to the different browsing behaviour between attack users and normal users. Secondly, we use OC-SVM to get the boundary of normal sessions set and build the normal users' behaviour models based on the features we selected. When the detection begin, the system constantly gets requests from the HTTP request queue, and adds the requests to the corresponding user's session. And then detection algorithm calculates whether the current session is in a normal session class. If the session is found to deviate from the normal session class, we record the session as abnormal, and add the corresponding user IP to blacklist. If the session is in the normal session class, we record the session as normal one.

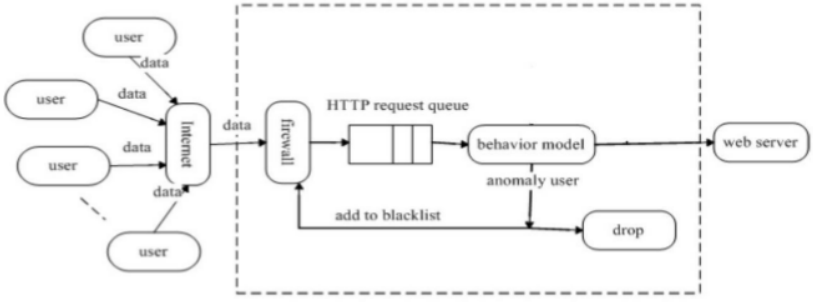

Figure 1. Detection architecture

4. NUMERICAL RESULTS

4.1. Datasets

To validate our defence method, we used the web-logs of the website of Sun Yat-sen University. The logs were collected on 10/4/2013. There are 20978 IP, 1,281,876 requests, and 2480 resources through the whole day. Timestamps have one-second precision.

In this experiment, the dataset above are seen as normal dataset. The requests of normal datasets last from 0s to 86382s. We launch a random attack from 40000s to 55000s. So attackers' requests are mixed with the normal users' requests from 40000s to 55000s. Fig.2 shows the arrival rate of normal traffic. And Fig.3 shows the arrival rate of attack traffic.

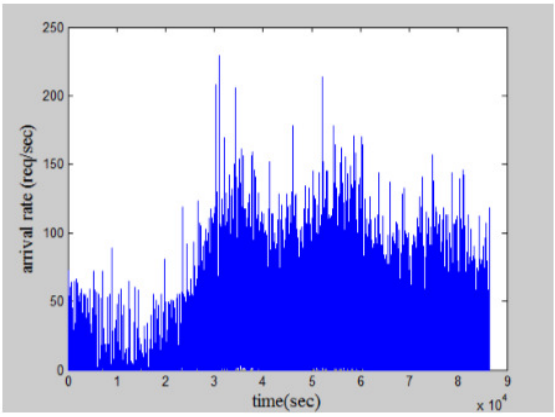

Figure 2. Normal traffic arrival rate

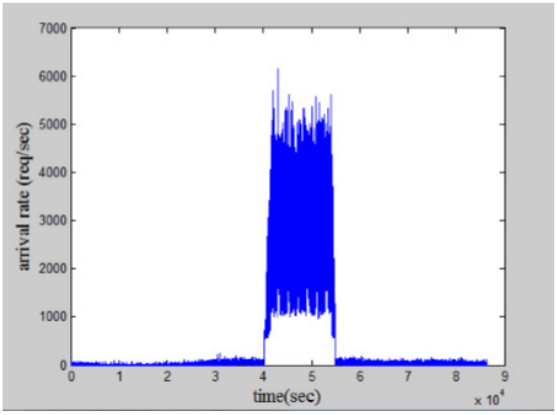

Figure 3. Attack traffic arrival rate

As mentioned above, request distribution is very important for our detection system. When random attack begins, request distribution will change. Let's compare the entropy of request distribution between normal datasets and attack datasets. As the definition above, the popularity of resource i is POP_i. So, the entropy of request distribution is calculated as following.

$$entropy = -\sum POP_i \log_2(POP_i) \tag{22}$$

We compute the entropy every 10 seconds. As we can see in Fig.4 Clearly, entropy changes after 40000s, and returns to be stable after 55000s. It matches the period of our simulation attack.

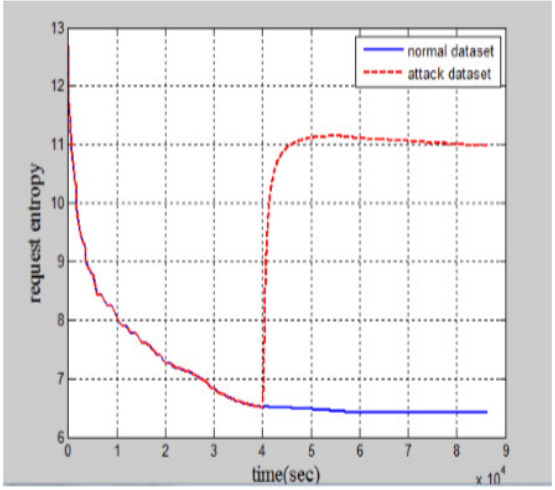

Figure 4: Normal and attack request entropy

4.2. Detection results

Our detection system uses OC-SVM to train the normal data from 0s to 40000s and then build normal users' model.

Based on the normal users' model we built, the system is ready to detect attacks. If a user is detected as abnormal, the system add this user to blacklist and block this user's requests. otherwise, the system record it as normal. Fig.5 is the Receiver Operating Characteristics (ROC) curves showing the performance of our detection model on application-layer DDoS attack.

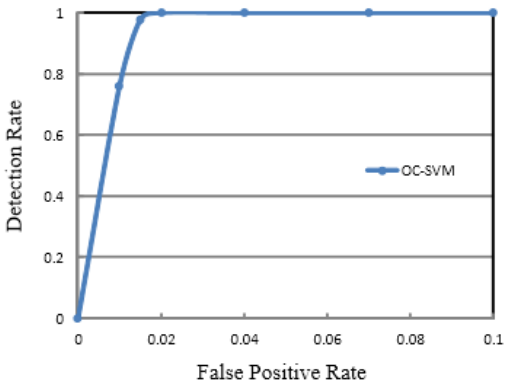

Figure 5: ROC of detection model

5. CONCLUSIONS

This paper proposed an application-layer DDoS detection method based on user behaviour model. To build user behaviour model, we extract features from users' sessions and cluster these sessions by OC-SVM method. And then, we use the model to detect anomaly of user behaviour. Numerical results based on real-traffic simulations demonstrate the efficiency of our detected strategy.

REFERENCES

[1] N.A. Mohammed and J.R. Martin. "Uniform DoS traceback." Computers & Security 2014;45 (2014):17-26.

[2] T. Spyridopoulos, G. Karanikas, T. Tryfonas, G. Oikonomou. "A game theoretic defence framework against DoS/DDoS cyber attacks." Computers & Security 2013;38 (2013):39-50.

[3] Y.Xie and S.Z. Yu. "Monitoring the application-layer DDoS attacks for popular website." IEEE/ACM Transactions on Networking 2009, 17(1): 15-25.

[4] A. Ramamoorthi, T. Subbulakshmi and S.M. Shalinie. "Real Time Detection and Classification of DDoS Attacks using Enhanced SVM with String Kernels," Proc. IEEE International Conference on Recent Trends in Information Technology, ICRTIT, Jun.2011, pp.91-96.

[5] S. Yin, X. P. Zhu and C. Jing, "Fault detection based on a robust one class support vector machine," Neurocomputing. 145(2014):263-268.

[6] F. Zhu, J.Yang, C. Gao, et al, "A weighted one-class support vector machine," Neurocomputing. 189(2016):1-10.

[7] X.Y. Huang and X.Y. Chen."A Novel Clustering Algorithm Based on One-Class SVM." Second WRI Global Congress on Intelligent Systems.2009, pp:486-490.

[8] Y. Xie and S.Z. Yu. "A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors." IEEE/ACM TRANSACTIONS ON NETWORKING 2009, VOL. 17, NO. 1.

[9] P. Park et al., "A Service-oriented DDoS detection mechanism using pseudo state in a flow router." Multimedia Tools and Applications 2014, Aug 2014.

[10] A. Lakhina, M. Crovella and C. Diot. "Mining Anomalies Using Traffic Feature Distributions," SIGCOMM'05, August 22C26, 2005, Philadelphia, Pennsylvania, USA

[11] J. Xue, M. Li, W. Zhao ,and S.Y. Chen. "Bound Maxima as a Traffic Feature under DDOS Flood Attacks," Mathematical Problems in Engineering 2012,Volume 2012, Article ID 419319, 20 pages.

[12] K. Lee ,J. Kim, K. H. Kwon et al., "DDoS attack detection method using cluster analysis," Expert Systems with Applications 2008, 34(3): 1659-1665.

[13] F. Simmross-Wattenberg et al., "Anomaly Detection in Network Traffic Based on Statistical Inference and α-Stable Modeling," IEEE Transations on Dependable and Secure Computing, Vol. 8, No. 4, pp. 494-509, July/August 2011.

[14] N.S. Stavros, K. Vasilios, S.K. Alexandros, K.P. Basil. "Real time DDoS detection using fuzzy estimators." Computers & Security 2012; 31(2012):782-790.

[15] J. Li, Y. Liu, L. Gu. "DDoS attack detection based on neural network," in: Proceedings of 2nd International Symposium on Aware Computing, IEEE explore, Tainan, Taiwan, 2010, pp. 196-199.

[16] B.B. Gupta, R.C. Joshi , M. Misra. "ANN based scheme to predict number of zombies in a DDoS attack," International Journal of Network Security 2012, 14(1): 35-46.

[17] K. Park , V. Pai, K. Lee and S. Calo. "Securing Web service by automatic robot detection," Proceedings of the Annual Conference on USENIX 06 Annual Technical Conference 2006. Boston, USA. 23-28.

[18] S. Yu , T. Thapngam , J. Liu, et al, "Discriminating DDoS flows from flash crowds using information distance," Network and System Security, 2009. NSS'09. Third International Conference on. IEEE, 2009. 351-356.

[19] G. Oikonomou and J.Mirkovic. "Modeling human behavior of defense against flash-crowd attacks," Proceedings of the 3rd International Conference on Communications. Dresden, Germany, 2009.14-18.

[20] S. Brklen et al., "User centric walk: An integrated approach for modeling the browsing behavior of users on the web," in Proc. 38th Annu. Simulation Symp. (ANSS'05) , Apr.2005, pp. 149-159.

[21] J. Velsquez , H. Yasuda , and T. Aoki. “Combining the web content and usage mining to understand the visitor behavior in a web site,” in Proc.3rd IEEE Int. Conf. Data Mining (ICDM03), Nov. 2003, pp. 669-672
[22] S.M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie. “Highdimensional and large-scale anomaly detection using a linear one-class SVM with deep learning.” Pattern Recognition, 58(2016), pp:121-134.
[23] Y.Liu, B.Zhang, B.Chen and Y.D.Yang. “Robust solutions to fuzzy oneclasss upport vector machine.” Pattern Recognition Letters, 71(2016), pp:73-77.
[24] J. Han and M. Kamber. “Data Mining Conepts and Techniques,” Burnington: Morgan Kaufmann, 2006, 251-351.
[25] J. Jung , B. Krishnamurthy and M. Rabinovich, “Flash crowds and denial of service attacks: Characterization and implications for CDNs and websites,” Proc. The 11th IEEE International World Wide Web Conference, Honolulu, Ha-waii, USA, ACM, 2002, pp.252-262.
[26] B. Scholkopf and J. C. Platt, “J. Shawe-Taylor, et al., Estimating the support of a highdimensional distribution,” Neural Comput. 2001, 13(7): 1443-1471
[27] S.S. Keerthi and C.J. Lin, “Asymptotic behaviors of support vector machines with gaussian kernel,” Neural comput. 2003, 15(7): 1667-1689.

Authors

Chuyu She is a Ph.D candidate in the School of Data and Computer Science at Sun Yat-Sen University. She is also working as a lecturer in the School of Mathematics and Statistics at Guangdong University of Finance & Economics. Her research interests include network security and software engineering.

Wushao Wen is a professor in the School of Data and Computer Science at Sun Yat-Sen University. He is co-appointed as an adjunct professor at SYSU-CMU Shunde International Joint Research Institute. His research interests include computer and network security, universal threat management, telecom networks and cloud computing security.

Zaihua Lin is a student in the School of Data and Computer Science at Sun Yat-Sen University. His research interests include computer and network security, big data, and social engineering.

Kesong Zheng is a graduate student in the School of Data and Computer Science at Sun Yat-Sen University. His research interests include computer and network security, software engineering, big data, spark and data mining.