Memoirs of the Graduate Schools of Engineering and System Informatics Kobe University
No. 7, pp. 8–13, 2015. doi:10.5047/gseku.e.2015.001

[Research Paper]

Distributed Denial of Service (DDoS) Backscatter Detection System Using Resource Allocating Network with Data Selection

Siti-Hajar-Aminah ALI[1], Nobuaki FURUTANI[1], Seiichi OZAWA[1], Junji NAKAZATO[2], Tao BAN[2], Jumpei SHIMAMURA[3]

[1] *Graduate School of Engineering, Kobe University, 1-1 Rokko-dai, Nada-ku, Kobe, Japan*
[2] *National Institute of Information and Communications Technology (NICT), Japan*
[3] *Clwit Inc. Japan*

(Received April 13, 2015; Accepted May 29, 2015; Online published June 8, 2015)

Keywords: Distributed Denial of Service, Backscatter, Resource Allocating Network, Darknet, Locality Sensitive Hashing

In this paper, we propose a fast detection system for Distributed Denial of Service (DDoS) backscatter using packets from various protocols and port numbers, which is not restricted to only the following two types of packets that can be labeled with simple rules called *labeled packet*: Transmission Control Protocol (TCP) Port 80 (80/TCP) and User Datagram Protocol (UDP) Port 53 (53/UDP). Usually, it is not easy to detect DDoS backscatter from the *unlabeled packets*, which an expert needs to analyze packet traffic manually. To deal with *unlabeled packets*, first, the detection system would learns general rules of DDoS backscatter using information from 80/TCP and 53/UDP. After the learning process, the generalized detection system is used to detect the DDoS backscatter from *unlabeled packets*. This detection system consists of two main modules which are pre-processing and classifier. In the pre-processing module, the incoming packets are transformed into feature vectors. As for the classifier module, since it is important to detect DDoS backscatter from all protocols as early as possible, we use Resource Allocating Network (RAN) with data selection. Using this classifier, the learning time is shortened because the classifier only learns essential data. Here, essential data means the data located in "well learned" regions, in which the classifier gives trustable predictions. To quickly search for the regions closest to given data, the well-known Locality Sensitive Hashing (LSH) method is used. The performance of the proposed detection system is evaluated using 9,968 training data from *labeled packets* and 5,933 test data from *unlabeled packets*. They are collected from January 1st, 2013 until January 20th, 2014 at National Institute of Information and Communications Technology (NICT), Japan. The results indicate that the detection system can detects the DDoS backscatter with high detection rate within a short time.

1. Introduction

As many people greatly rely on the Internet these days, potential threats to have cyber attacks are increasing. One of these serious threats is Distributed Denial of Services (DDoS) attack. DDoS attack has been used by cyber criminals to cause damage virtually to the targeted organization especially organizations which rely on Internet such as news sites, political parties, financial institutions, gaming sites and online businesses. DDoS attack can cause material losses (i.e., finance and resources bandwidth), ruin the reputation of the targeted company or organization and decrease the consumers confidence. By saturating the targeted bandwidth with fake requests, this attack can cause the legitimate users to experience a slowed down internet services when requesting connection to targeted websites, or even in the worst cases, the targeted websites would suffer a total blockage for a minutes, several hours or even a week. The worst attack in recent history was recorded on 20th November 2014, when 500 gigabits per second of junk traffic were pounded at Apple Daily and Pop Vote which are two independent news sites at Hong Kong[1)]. Given the devastating effects of DDoS attack on our cyber infrastructure, it is very important to detect the DDoS attack as quickly as possible before the Internet traffic is totally down.

Backscatter is the after-effect of DDoS attack when the source Internet Protocol (IP) addresses are spoofed by the attacker. Backscatter happens when the target server sends unsolicited response packets to the spoofed IP addresses. Some of these random IP addresses are directed to the darknet (i.e., a set of unused IPs). By spoofing the source IP addresses, the attackers not only can amplify the DDoS attack, but also are able to conceal their identity. According to the reports on the Cloudflare website[2)], 92% of DoS attacks using Transmission Control Protocol (TCP) at Cloudflare are through Port 80 (Hypertext Transfer Protocol (HTTP)), and 97% of the Denial of Services (DoS) attacks using User Datagram Protocol (UDP) are through Port 53 (Domain Name System (DNS)). Basically, it is expected that the companies and organizations with service-based websites would have a trend of attack similar to Cloudflare. It is because web servers often use TCP Port 80 (80/TCP), whereas DNS servers usually use Port 53 (53/UDP).

The most common DDoS attack is using "flood" attack in which a large number of packets containing useless information are sent by this flood through various internet protocols such as TCP and UDP[3)]. The main purpose of the flood attack is to occupy the server resources so that the server is unavailable to the legitimate user. To send DDoS attack through 80/TCP, firstly, the attacker would request a connection by sending packets with SYN flag set to '1' to the World Wide Web (www) of the targeted server using a spoofed source IP address. Then, the targeted server would send a response packet with SYN-ACK flags to the source IP addresses. The targeted server is under the DDoS attack when the attacker sends a SYN flood to the targeted server using botnet and consumes the available resources by refusing to respond the SYN-ACK flags. Since the available resources are consumed by these overwhelming connection requests and waiting period for reply packets with ACK flag, the targeted server would denies any request from the legitimate users. Another frequent attack is using 53/UDP in which the attacker manipulates the operation of

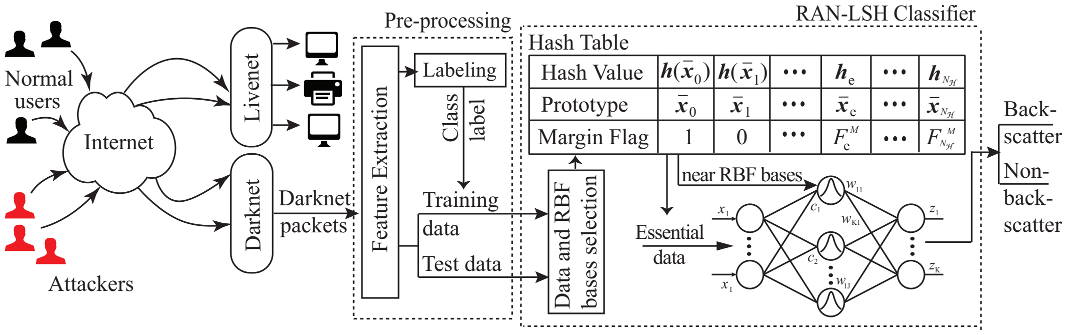

Fig. 1 The architecture of DDoS backscatter detection system.

the DNS server to amplify the DDoS attack. DNS server is used to search the corresponding IP address of the requested internet domain name. Internet Control Message Protocol (ICMP) "Destination Unreachable" message would be generated the moment after the system had noticed inactive ports. Thus, the additional packets containing these messages would cause traffic congestion at the targeted server when a large number of UDP packets are sent to these inactive ports.

Both attacks from labeled packets (80/TCP and 53/UDP) are easy to be detected based on the flags in these packets. However, for other protocols with different port numbers, usually an expert is hired to analyze the characteristics of packet traffic manually. This method is time consuming and is costly because a professional needs to be hired. Therefore, we propose an autonomous DDoS backscatter detection for all protocols and port numbers using Resource Allocating Network with Locality Sensitive Hashing (RAN-LSH) classifier. The Resource Allocating Network (RAN) classifier is embedded with the well-known Locality Sensitive Hashing (LSH) technique which is used to select essential data. The LSH method indeed has proven to be able to quickly find similar objects in a large database[4)].

Although extensive research has been carried out on DDoS attack detection using machine learning model[5)], to our knowledge, no attempt was done to explore the potential of data selection to speed up the learning time of the DDoS attack detection system. To address this gap, we use RAN-LSH classifier which embedded with data selection to learn only essential data. Learning irrelevant data would not only lead to a slow learning time[6)] but also can cause the deterioration in detection performance of a classifier. Oyang et. al.[7)] suggest that the training samples that are located near to the boundaries between different classes would carry more crucial information. By calculating the difference between the highest network output and the second highest network output, we can judge whether the training data are located near the class boundaries or not. With the expectation that if the difference between them is large, it means that the given data is "well-learned", whereas the small difference indicates that the data is "not well-learned". We assume that the "not well-learned" data is located at the class boundaries and is essential to be learned. In this study, we define an output margin by the difference between the two highest network outputs. By combining the margin-based data selection with LSH technique, the learning time is expected to be accelerated.

To evaluate our detection system, darknet packets collected by National Institute of Information and Communications Technology (NICT) in Japan are used in this study. Darknet is a group of IP addresses which is not connected to any active services such as computer, printer or router. Thus, it is unusual for darknet to receive any packets since darknet is not capable to start any communication with other server. Based on this fact, darknet is an effective medium for collecting data set of DDoS attack because a large number of true positive data (i.e., DDoS backscatter packets) can be collected with only small number of false positive data (i.e., DDoS non-backscatter packets). There are at least two possibilities of false positive data, which are from misconfigurated IP setting and the scan activity by botnet to find available IP addresses to be spoofed. Since NICT has the largest network monitoring system in Japan that assembled 140 thousands unused Internet Protocol version 4 (IPv4)[8)], we expect to collect a good data set using this resources.

The rest of this paper is organized as follows: Section 2 describes the overall architecture of the proposed DDoS detection system. Section 3 validates the effectiveness of our detection system to detect the DDoS attack from different protocols and port numbers, other than labeled packets. Finally, conclusions and future work are discussed in Section 4.

Algorithm 1 DDoS Backscatter Detection

```
Input: Darknet packets from 80/TCP and 53/UDP.
Output: The class label of given darknet packets.
1: Do Pre-processing to generate feature vectors of training data
   $X' = \{(x_i, d_i)\}_{i=1}^{N_{d'}}$ from darknet packets.
2: Do the cross-validation for $N_0$ initial training data $X_0 = \{(x_i, d_i)\}_{i=1}^{N_0}$
   to determine the RBF width $\sigma$, the number of partitions $P$, error threshold
   $\varepsilon$ and distance threshold $\delta$. Determined the other four parameters
   empirically; accumulation ratio $\theta_a$, output margin $\theta_m$, network output
   $\theta_z$ and tolerant distance $\theta_p$.
3: Do RAN-LSH Learning to construct initial RAN.
4: loop
5:     Input: New darknet packets from other than 80/TCP and 53/UDP.
6:     Do Pre-processing to generate feature vectors of test data
       $X'' = \{x_i\}_{i=1}^{N_{d''}}$ from darknet packets.
7:     Test $X''$ using the constructed RAN and obtain the class label of the test data
       $X'' = (x_i, d_i)$.
8:     Do RAN-LSH Learning to learn new training data $X' \leftarrow X''$.
9: end loop
```

2. DDoS Backscatter Detection System

2.1 System Architecture

Figure 1 shows the overall architecture of DDoS backscatter detection system. IP addresses can be divided into the following two categories: livenet and darknet. Livenet consists of a group of IP addresses that are connected to actual hosts and devices (i.e., desktop, laptop and printer). While the remaining IP addresses other than livenet are included in darknet category. In general, there should not be any legitimate packets sent to a dark-

Algorithm 2 Pre-processing

```
Input: Darknet packets.
Output: DDoS backscatter data set: training data $X' = \{(x_i, d_i)\}_{i=1}^{N_{d'}}$ or test data $X'' = \{x_i\}_{i=1}^{N_{d''}}$.
 1: for all darknet packets do
 2:     Sort the darknet packet by the source IP addresses.
 3:     if time interval between two sequence of packets ≥ one hour then
 4:         Define a new data set $i \leftarrow i + 1$.
 5:         Extract 17 features from the packets of the first one minute time-window.
 6:         if the darknet packets are training data then
 7:             Do Labeling
 8:             Create ith feature vector $(x_i, d_i)$.
 9:         else if the darknet packets are test data then
10:             Create ith feature vector $x_i$.
11:         end if
12:     end if
13: end for
```

Algorithm 3 Labeling (for 80/TCP and 53/UDP)

```
Input: Input data containing 17 features $x_i$.
Output: Class label $d_i$.
 1: loop
 2:     if the packets are from 80/TCP then
 3:         if (SYN=1 and ACK=1) || (RST=1) then
 4:             Response flag $F^R = 1$.
 5:         end if
 6:     else if the packets are from 53/UDP then
 7:         if the Query/Response flag QR=1 & (domain name contains http, www, com) then
 8:             Response flag $F^R = 1$.
 9:         end if
10:     end if
11:     if Number of packets sent in one minute ≥ 40 & $F^R = 1$ then
12:         Label the packet as DDoS backscatter $d_i = (1, 0)$.
13:     else
14:         Label the packet as DDoS non-backscatter $d_i = (0, 1)$.
15:     end if
16: end loop
```

net. However, there is a possibility of configuration errors during data transmission, which mistakenly end up to reach darknet. Nevertheless, there are users who intentionally use darknet to perform cyber criminal activities such as DDoS attack. In this study, we used darknet packets collected by NICT as data set. There are two main modules of the detection system, which are pre-processing and classifier. Firstly, the raw darknet packets are passed to the pre-processing module where the packets are transformed into feature vectors. Then, these feature vectors are given to the classifier for learning and testing. Algorithm 1 shows the overall procedures of the detection system which consists of the following two learning phases: initial learning (Lines 1-3) and incremental learning (Lines 4-9).

2.2 Pre-processing

To make sure that the classifier of the detection system can learn the decision boundary of different classes (i.e., DDoS backscatter or non-backscatter), only important information are extracted from raw data packets. Algorithm 2 shows the pre-processing procedures to create the data set of the DDoS backscatter. First and foremost, the darknet packets are sorted according to the source IP addresses (Line 2). Next, 17 features are extracted from the first one-minute packets as one data (Line 5). Here, we defined a new data as two sequential packets which are separated by one hour of time interval between them (Lines 3-4). For training data, the feature vectors consist of 17 features and the corresponding class label $(\boldsymbol{x}_i, \boldsymbol{d}_i)$, whereas the test data do not contain the class label in the feature vectors $\boldsymbol{x}_i$. The 17 features which are extracted from raw data packets are as follow:

(i) Number of packets.
(ii) Average time difference between consecutive packets.
(iii) Variance of time differences between consecutive packets.
(iv) Number of Source Port.
(v) Average number of packets to a source port.
(vi) Variance of the numbers of packets to a source port.
(vii) Number of protocols.
(viii) Number of destination IPs.
(ix) Average number of packets to a destination IP.
(x) Variance of the numbers of packets to a destination IP.
(xi) Average difference in destination IPs for consecutive packets.
(xii) Variance of the differences in destination IPs for consecutive packets.
(xiii) Number of destination ports.
(xiv) Average number of packets to a destination port.
(xv) Variance of the numbers of packets to a destination port.
(xvi) Average payload size.
(xvii) Variance of payload sizes.

Meanwhile, the steps to assign a class label to the labeled packets are shown by Algorithm 3. If the source host sends more than 40 packets per minute and its response flag F^R is equals to '1' (Line 11), then the feature vector for the one-minute darknet traffic is classified as DDoS backscatter. To set the response flag as $F^R = 1$, the data from labeled packets need to fulfill the conditions in Line 3 and Line 7, respectively. The remaining data are labeled as non-DDoS backscatter.

2.3 RAN-LSH Classifier

Radial Basis Function Network (RBFN)[9] has at least three desirable properties: (1) the architecture of the model is simple and it can easily be modified from batch learning algorithm to incremental learning algorithm[10], (2) the ability to approximate any function if a sufficient number of hidden units are given[11], and (3) the capability of handling the classification problem of more than two classes in a single run[7]. In this study, we use the incremental model of RBFN so-called RAN[12].

Let the number of inputs, RBF units, and outputs be I, J, and K respectively. The RBF outputs $\boldsymbol{y}(\boldsymbol{x}) = (y_1(\boldsymbol{x}), \cdots, y_J(\boldsymbol{x}))^T$ and the network outputs $\boldsymbol{z}(\boldsymbol{x}) = (z_1(\boldsymbol{x}), \cdots, z_K(\boldsymbol{x}))^T$ of inputs $\boldsymbol{x} = (x_1, \cdots, x_I)^T$ given to the classifier are calculated as follows:

$$y_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_j\|^2}{\sigma_j^2}\right) \quad (j = 1, \cdots, J) \tag{1}$$

$$z_k(\boldsymbol{x}) = \sum_{j=1}^{J} w_{kj} y_j(\boldsymbol{x}) + \xi_k \quad (k = 1, \cdots, K) \tag{2}$$

where $\boldsymbol{c}_j = (c_{j1}, \cdots, c_{jI})^T$, σ_j^2, w_{kj} and ξ_k are the center of jth RBF unit, the variance of the jth RBF unit, the connection weight from the jth RBF unit to the kth RBF unit and the bias, respectively.

Algorithm 4 shows the learning algorithm of RAN-LSH which has been proposed in[13]. This model is the extended model of RAN which adopted the LSH-based data selection to accelerate the learning time. The RBF centers of RAN-LSH are not trained but selected based on the error between the prediction class label obtained by the classifier model using Eq. (2) and actual class label of the training data $\boldsymbol{d}$. Low error means that the existing RBF centers are "well-learned". On the other hand, high error value

Algorithm 4 RAN-LSH Learning

```
Input: Data $X' = \{(x_i, d_i, h(x_i))\}_{i=1}^{N}$, RAN, hash table $\mathcal{H}$ and parameter $\theta_p$.
Output: RAN.
 1: for all $(x_i, d_i, h(x_i)) \in X'$ do
 2:   Calculate (i) the outputs $z$ for $x_i$, (ii) error: $E = \|z(x) - \mathbf{1}_c\|/K$ and (iii) find the nearest center $c^*$ to $x_i$.
 3:   if $E > \varepsilon$ & $\|x - c^*\| > \delta$ then
 4:     Add an RBF unit (i.e., $J \leftarrow J+1$) and set an RBF center, and the connection weights: $c_J = x_i$, and $w_J = d_i - z$.
 5:     Get an index set $C$ of RBFs position in hash table $\mathcal{H}_{\{C_J=e\}}$: $h(c_J) = h(\bar{x}_e)$.
 6:   else
 7:     for $j = 1$ to $J$ do
 8:       Calculate LSH distance $d_j^*$ using Eq. (7).
 9:       if $d_j^* \le \theta_p$ then
10:         Define a set $\mathcal{R}$ of selected RBF centers (i.e $\mathcal{R} \leftarrow j$).
11:       end if
12:     end for
13:     for all $\{(c_j, w_j)\}_{j\in\mathcal{R}}$ do
14:       Calculate the outputs $\mathbf{\Phi}$ for selected RBF centers $c_j$ ($j \in \mathcal{R}$) and a training data $x_i$ by Eq. (1).
15:       Decompose $\mathbf{\Phi}$ using SVD and obtain the matrices $V$, $H$, $U$.
16:       Update connection weights $W' = VH^{-1}U^T D$.
17:     end for
18:     Calculate (i) the outputs $z$ for $x_i$ and (ii) the error: $E = \|z(x) - \mathbf{1}_c\|/K$
19:     if $E > \varepsilon$ then
20:       Add a hidden unit and the connection weights: $c_J = x_i$, and $w_J = d_i - z$.
21:       Get an index set $C$: $C_J = e$.
22:     end if
23:   end if
24: end for
```

indicates that the RBF centers are "not well-learned" and thus, it is necessary to add the data as a new RBF center in the hidden layer node (Lines 4 and 20). Besides the RBF centers selection, another important element in RAN classifier is updating the connection weights. To shorten the learning time, instead of using all RBF centers to update the weights, we only use local RBF centers which are RBF centers that are located near to the training data (Lines 7-12).

RAN-LSH model is based on the idea that the data which are located near to each other would probably have almost similar value of RBF outputs and network outputs. Therefore, data that located near to each other are grouped into a same group which we called as region. LSH technique consists of two main steps: firstly, to group similar data into a same group, and secondly, to store them in a table called hash table where hash values are used as the index of the hash table. With the former step, the data are projected into several projection vectors and each of the projection vectors is divided into equal size of partition or called as partition. With the latter step, the hash values which correspond to buckets comprise of a hash function for every projection vectors. When two different data are always assigned in the same partition while the directions of the projected vectors are changing multiple numbers of times, it means that the data are most probably located close to each other. Nevertheless, assigning to many projection vectors would lead to the construction of a bigger size of hash table that requires not only larger memory but also longer processing time. Therefore, it is important to choose a proper number of hash functions. PCA can generate a proper number of hash functions by controlling the threshold of the accumulation ratio θ_a.

Let l be the subspace dimensions obtained by PCA. Then, the following linear transformation is considered to define a partition in LSH:

$$V = U_l^T x \tag{3}$$

where $V = (v_1, \cdots, v_l)$, $U = (u_1, \cdots, u_l)$ and x are the l-dimensional projection vector, the matrix of l eigenvectors, an I-dimensional input vector, respectively.

A sub-region is defined by a series of assigned partition which are divided based on the l projection values. Each sub-regions is stored as an entry in hash table. A hash table is composed of the following three items: hash value h_e, prototype $\bar{x}_e$ and margin flag F_e^M. The index or the hash values h_e are used as a key to find a matching entry e of a similar item which has been registered previously in hash table (Figure 1: top right). Hash values $h(x) = (H(v_1), \cdots, H(v_l))$ are a series of hash function $H(v_i)$ given as follows;

$$H(v_i) = \max\{\left\lceil \frac{\min\{\max\{v_i, v_i^-\}, v_i^+\} - v_i^-}{v_i^+ - v_i^-} P \right\rceil, 1\} \tag{4}$$

where v_i^+ and v_i^- be the upper and lower values of typical projections v_i on the ith eigenvector u_i, respectively. Whereas P is the number of partition which is used to divide the projection vectors.

The prototype $\bar{x}_e$ is the average value of data in each entry of the hash table. If the hash values have not been registered in the hash table, the margin flag is calculated as follow:

$$F_e^M = \begin{cases} 1 & (\Delta z(x_e) > \theta_m) \ \ \& \ \ (z_{k'}(x_e) > \theta_z) \\ 0 & (\text{otherwise}) \end{cases} \tag{5}$$

where the output margin Δz is given by the difference between the highest network output and second highest network output shown by the following equation:

$$\Delta z = \underset{k' \in \{1,\ldots,K\}}{\text{argmax}}\, z_{k'}(x) - \underset{k'' \in \{1,\ldots,K\}\setminus k'}{\text{argmax}}\, z_{k''}(x). \tag{6}$$

If the flag of the matched prototype is '1', it means the classifier is "well-trained" around the prototype; therefore, there is no need to train a given data. On the contrary, if the flag is '0', it means a given data should be trained. After the learning phase, the margin flag F^M should be updated. Nevertheless, updating the margin flag of every prototype in the hash table would increase the learning time. As mentioned before, the prototype with $F^M = 1$ means the classifier is well trained around the prototype. Thus, this prototype does not need to be updated. Meanwhile, prototype with $F^M = 0$ should be updated because there would probably be regions that have become "well-trained" after the learning phase.

The LSH is also used to find near RBF bases to the training data. These RBF bases are used to update the connection weights. Firstly, the hash values of RBF bases are retrieved from the constructed hash table. Next, the LSH distance $d^*{}_j$ for each jth RBFs is calculated as follows:

$$\begin{aligned} d_j^* &= h(x) - h(c_j) \\ &= \sum_{i=1}^{l} \left(\left| H(v_{x\_i}) - H(v_{cj\_i}) \right| \right) \end{aligned} \tag{7}$$

where we calculate the distance between a given training data $h(x)$ and all jth RBF centers $h(c_j)$ by accumulating the difference of hash functions in each ith projection vector v_i in V. Then the RBF centers with LSH distances that are less or equal to θ_p would be selected to solve the linear equation in $\mathbf{\Phi} W = D$.

3. Performance Evaluation

3.1 Experimental Setup

To evaluate the performance of the detection system with data selection mechanism, we carry out the following two experiments on: (1) the influence of the threshold parameters for output margin θ_m and network output θ_z and (2) the effectiveness of incremental learning.

The first experiment is carried out to see the effects of different values θ_m and θ_z using initial training data. In the second experiment, we compare the performance of the three classifier models: RBFN, RAN and RAN-LSH. RBFN is learned with data within a 90-day time window in a batch learning mode, and the time window is shifted at every 1 day for an online learning purpose. For instance, the first learning stage is carried out using data Day 277 until Day 366 as training data and data Day 367 as test data. During the next learning stage, the time-window is shifted by one, as such for the second learning stage, Day 278 until Day 367 is used for training, whereas the performance are tested using data Day 368. On the other hand, RAN and RAN-LSH are trained with data of the first 90 days, and then they are trained incrementally with a chunk of data given everyday. The first learning stage is similar to those in batch learning. The difference is the model is updated only with the data given in a day (i.e., the past data are not kept and used for a learning purpose). For example, in the second learning stage, Day 367 and Day 368 are used as training data and test data, respectively. The learning steps are continued using the next sequence of data until all data have been learned by the model. Since the training data are given based on window-size, the sizes of a data chunk are different at every learning stage. The number of data in a 90-day time window ranges from 8,513 to 13,558, and 11,123 on average. Whereas, the minimum, maximum and average numbers of data collected in a day are 0, 436 and 41, respectively.

In this detection system, four parameters are determined empirically which are accumulation ratio θ_a, output margin θ_m, network output θ_z and tolerant distance θ_p. In the first experiment, we investigate the effect of output margin θ_m and network output θ_z regarding the detection rate and learning time, so that the data selection mechanism is carried out effectively. In the following experiment, the parameters are set to: $\theta_a = 0.9$, $\theta_m = 0.01$, $\theta_z = 0.6$ and $\theta_p = 2$. Meanwhile, the error threshold ε and δ are set to 0.5 and $\sqrt{2}\sigma$, respectively. The other parameters, RBF width σ and the number of partitions P are determined in the initial learning phase through the cross-validation. There are 9,968 data of labeled packets (9,404 backscatter and 564 non-backscatter) collected from 1st January 2013 to 31st December 2013 (1 year). For the test purpose, 5,933 data of unlabeled packets (2,464 backscatter and 3,469 non-backscatter) are collected from 1st January 2014 to 20th January 2014 (20 days).

Besides measuring the learning time, the performance of the detection system is also evaluated based on recall rate, precision rate and F1 measure. The recall rate measures the ratio correctly classified as DDoS backscatter (True Positive) out of expected DDoS backscatter determined by the classifier. Meanwhile, the precision rate measures the True Positive sample out the total of data with DDoS backscatter class label. The F1 measure is the harmonic mean of both recall and precision rates. For this detection system, it is more crucial to have at least high percentage of recall rate to reduce the risk of misclassifying DDoS attack, where a low recall rate indicates that the detection system has mistakenly classified many backscatter data as non-backscatter. Therefore, in this situation, a right action to salvage a targeted host could be delayed, resulting in a significant loss. However, having too low precision rate is also inappropriate because the detection system tends to have classified most of the data as backscatter.

3.2 Parameters Tuning for Data Selection

For RAN-LSH classifier which uses LSH data selection, two important parameters have been identified to affect the detection rate and learning time, namely the thresholds for output margin θ_m and network output θ_z. Table 1 shows the result of parameter tuning for θ_m and θ_z using three months of training data and 20-days of test data regarding three evaluation: (a) number of data selected, (b) F1 measure, and (c) learning time. Equation (5) indicates that the higher the threshold values of θ_m and θ_z are, the more stringent the requirements to determine the data as "well-learned". Thus, if high values are used for both parameters, it is expected that only a few data would have margin flag $F^M = 1$. Consequently, the learning time is expected to be increased because many data should be selected to learn the detection system (i.e., data with margin flag $F^M = 0$). To accelerate the learning time, the number of selected data should be small by tuning both θ_m and θ_z.

Table 1. Parameter tuning of data selection using three months training data and 20 days test data. The performance are measured regarding the (a) total number of selected data, (b) F1 measure [%] and (c) learning time [sec.]

(a) Total number of selected data

θ_m	θ_z		
	0.5	0.6	0.75
0.01	2704	3839	6690
0.05	3115	3839	6745
0.1	3319	3839	6891
0.2	3769	3844	6891

(b) F1 Measure [%]

θ_m	θ_z		
	0.5	0.6	0.75
0.01	89.4	91.2	93.1
0.05	89.5	91.3	93.1
0.1	90.1	91.3	93.1
0.2	91.1	91.3	93.1

(c) Learning time [sec.]

θ_m	θ_z		
	0.5	0.6	0.75
0.01	1122.5	5008.1	46071.2
0.05	1915.0	5021.5	46464.9
0.1	2510.4	5284.4	47926.5
0.2	4316.8	5528.8	48184.4

By depending on only the output margin θ_m alone to determine whether the data are located in "well-learned" region or not, it is not enough to reduce the number of selected data effectively. Table 1(a) shows that the number of selected data is high even though a small value of θ_m is used (i.e., $\theta_m = 0.01$) for different values of θ_z. Almost twice the number of selected data is able to be reduced if another threshold is used namely θ_z. To ensure that the selected data provides a high detection rate, the F1 measure of θ_z value should not differ much compared to the other value of θ_z. For RAN-LSH model with $\theta_z = 0.6$ and $\theta_m = 0.01$, almost 10 times of learning time can be shortened and about 0.9% of difference at F1 measure, compared to the model with $\theta_z = 0.75$ and $\theta_m = 0.2$. Taking the tradeoff between short learning time and high detection rate into account (see Table 1(b) and 1(c)), $\theta_z = 0.6$ and $\theta_m = 0.01$ are the most suitable for reducing the learning time of the detection system.

3.3 Performance of DDoS Backscatter Detection System

Table 2 shows the performance comparison between three different classifiers: RBFN, RAN and RAN-LSH regarding recall rate, precision rate, F1 measure and learning time. RBFN uses batch learning, while RAN and RAN-LSH use incremental learn-

Table 2. Comparison of three different models: RBFN, RAN and RAN-LSH. RBFN uses batch learning, whereas RAN and RAN-LSH use incremental learning setting. The performance is measured regarding (a) recall rate [%], (b) precision rate [%], (c) F1 measure [%], and (d) learning time [sec.].

Evaluation	RBFN[a]	RAN[b]	RAN-LSH[c]	RAN-LSH[b]
Recall rate [%]	98.4	98.1	98.1	97.8
Precision rate [%]	96.2	97.2	65.9	97.4
F1 measure [%]	97.3	97.6	77.8	97.5
Learning time [sec.]	58135.5	207.9	3.3	46.2

[a] batch-retrained.
[b] incremental-with update.
[c] incremental-without update.

ing, in which the learning modes are explained in Section 3.1. The results obtained are expected and not surprising. The detection system using RBFN classifier obtains the highest recall rate compared to other classifiers. This is because the RBFN classifier learns using a large number of training data (data collected during the 90-day window size). Since this classifier is retrained in every learning stage, this classifier requires longer learning time than RAN and RAN-LSH. By learning incrementally, RAN and RAN-LSH classifier updated the weights of the classifier using only the recent training data. This can save a lot of time. Moreover, using RAN classifier with LSH-based data selection (RAN-LSH), the detection system is capable to learn faster than RAN, because only essential data from the training data are selected to be learned, instead of learning all given training data using RAN classifier. Based on the result shown in Table 2, to give class labels to 5,581 test data, RBFN and RAN requires approximately 10.4 (sec.) and 0.04 (sec.), respectively in average for each test data. As for RAN-LSH, only 0.008 (sec.) is required which is 130 times faster than RBFN classifier and five times faster than the RAN classifier. Since the recall rate is not significantly different, the detection system with RAN-LSH classifier is expected to detect DDoS backscatter faster, and hence allowing the moderator of the targeted server to take precautious action on the suspicious IP addresses before the Internet traffic of the targeted server is effected by the DDoS attack. Besides that, we also examined the effectiveness of updating RAN-LSH classifier in every learning stage. It is obvious that without updating RAN-LSH classifier using the recent training data, the overall performance is dropped where we can see that the precision rate and the F1 measure decreased about 20%. Based on this result, it gives an evidence that the incremental learning of the detection system is carried out effectively.

4. Conclusions

In this paper, we proposed a fast DDoS backscatter detection system for unlabeled packets using RAN-LSH classifier. It has been shown that the proposed detection system is capable to perform well, comparable to the detection system using conventional classifiers (RBFN and RAN). The current findings add to a growing body of literature on LSH, in which LSH is used to find similar data. However, in this study, LSH is used to select essential data to learn the detection system. The proposed detection system provides a desirable characteristic as a DDoS backscatter detection system. At every minute, darknet packets are transformed into a feature vector and it is provided to the RAN-LSH classifier to detect DDoS backscatter. The time to classify needs less than 0.01 (sec.).

The proposed system has some limitations when deployed under a real environment. First, the labeling of darknet packet features is not easy except for the packets of 80/TCP and 53/UDP. Basically, the labeling must be carried out by expert surveyors through visual observation. Therefore, only unreliable decision by a classifier should be presented to a surveyor for labeling purpose. However, this function is not implemented in the proposed detection system at this preset. Second, there are a few parameters that need to be tuned during initial learning in order to achieve high detection rate. The parameters are RBF width σ, the number of partitions P, error threshold ε, distance threshold δ, accumulation ratio θ_a, tolerant distance θ_p, output margin θ_m and the highest network output θ_z.

The above issues are left as our future work.

Acknowledgements: This work is partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C) 24500173, the University of Tun Hussein Onn Malaysia (UTHM) and the Ministry of Education Malaysia (KPM).

References

1) Olson, P.; The Largest Cyber Attack in History Has Been Hitting Hong Kong Sites, http://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/ (2014)
2) Graham-Cumming, J.; The Wednesday Witching Hour: CloudFlare DoS Statistics https://blog.cloudflare.com/the-wednesday-witching-hour-cloudflare-dos-st/ (2012)
3) Jackson, D.; Understanding and Combining DDoS Attacks: A Threat Analysis http://www.secureworks.com/assets/pdf-store/articles/Understanding_and_Combating_DDoS_Attacks.pdf (2011)
4) Andoni, A. and Indyk, P.; "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Communications of the ACM*, 51(1), 117–122 (2008)
5) Xu, X., Wei, D. and Zhang, Y.; "Improved Detection Approach for Distributed Denial of Service Attack Based on SVM," *Third Pacific-Asia Conference on Circuits, Communications and System (PACCS'11)*, 1–3 (2011)
6) Langley, P.; "Selection of Relevant Features in Machine Learning," *Proceedings of the AAAI Fall Symposium on Relevance*, 1–5 (1994)
7) Oyang, Y. J., Hwang, S. C., Ou, Y. Y., Chen, C. Y. and Chen, Z. W; "Data Classification with Radial Basis Function Networks Based on a Novel Kernel Density Estimation Algorithm," *IEEE Trans Neural Networks*, 16(1), 225–236 (2005)
8) Hishinuma, H.; Information Security by NICT and the Government of Japan, http://www.bic-trust.eu/files/2011/12/slides13.pdf (2011)
9) Broomhead, D. S., Lowe, D.; "Radial Basis Functions, Multi-variable Functional Interpolation and Adaptive Networks (Technical report)," *Royal Signal and Radar Establishment (RSRE)*, 4148, 1–39 (1988)
10) Poggio, T. and Girosi, F; "Networks for Approximation and Learning," *IEEE Trans. on Neural Networks*, 78(9), 1481–1497 (1990)
11) Okamoto, K., Ozawa, S., Abe, S.; "A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory," *Proceedings of International Joint Conference on Neural Networks*, 102–107 (2003)
12) Platt, J.; "A Resource-Allocating Network for Function Interpolation," *Neural Computation*, 3(2), 213–225 (1991)
13) Ali, S. H. A., Fukase, K. and Ozawa, S.; "A Neural Network Model for Large-Scale Stream Data Learning Using Locally Sensitive Hashing," *Neural Information Processing Lecture Notes in Computer Science*, 369–376 (2013)