EMITTER International Journal of Engineering Technology
Vol. 6, No. 1, June 2018
ISSN: 2443-1168

Botnet Detection Using On-line Clustering with Pursuit Reinforcement Competitive Learning (PRCL)

Yesta Medya Mahardhika, Amang Sudarsono, Aliridho Barakbah

Postgradaute Applied Engineering of Technology
Division of Information and Computer Engineering, Department of Information and Computer Engineering, Electronic Engineering Polytechnic Institute of Surabaya
EEPIS Campus, Jalan Raya ITS, Sukolilo 60111, Indonesia
Email: yestamedya@pasca.student.pens.ac.id, {amang, ridho}@pens.ac.id

Abstract

Botnet is a malicious software that often occurs at this time, and can perform malicious activities, such as DDoS, spamming, phishing, keylogging, clickfraud, steal personal information and important data. Botnets can replicate themselves without user consent. Several systems of botnet detection has been done by using classification methods. Classification methods have high precision, but it needs more effort to determine appropiate classification model. In this paper, we propose reinforced approach to detect botnet with On-line Clustering using Reinforcement Learning. Reinforcement Learning involving interaction with the environment and became new paradigm in machine learning. The reinforcement learning will be implemented with some rule detection, because botnet ISCX dataset is categorized as unbalanced dataset which have high range of each number of class. Therefore we implemented Reinforcement Learning to Detect Botnet using Pursuit Reinforcement Competitive Learning (PRCL) with additional rule detection which has reward and punisment rules to achieve the solution. Based on the experimental result, PRCL can detect botnet in real time with high accuracy (100% for Neris, 99.9% for Rbot, 78% for SMTP_Spam, 80.9% for Nsis, 80.7% for Virut, and 96.0% for Zeus) and fast processing time up to 176 ms. Meanwhile the step of CPU and memory usage which are 78 % and 4.3 GB for pre-processing, 34% and 3.18 GB for online clustering with PRCL, and 23% and 3.11 GB evaluation. The proposed method is one solution for network administrators to detect botnet which has unpredictable behavior in network traffic.

Keywords: Botnet Detection, Maliciouse Software, On-line Clustering, Pursuit Reinforcement Competitive Learning

1. INTRODUCTION

Botnet or Robot Network is one of network security and data threat that often happens in this time [1]. Botnets have become the common channel to conduct cybercrime [2], by causing harmful activities, such as DDOS, Spamming, Phishing, keylogging, click fraud, theft of personal data and important information according to recent Symantec threat reports [3].

Botnets can replicate themselves by infecting other computers that are vulnerable to bots using IRC (Internet Relay Chat)protocol, HTTP (Hypertext Transfer Protocol) and P2P (Peer to Peer) and can be controlled remotely through the network by exploiting a set of computers that have been infected (bots or zombies). Botnets have work like a group of special soldiers who suddenly infiltrate and attack simultaneously, therefore it is difficult to prove whether a computer is infected by a botnet. Botnet have infected computer or zombie computer activity without the owners concern.

Another danger of misuse of the internet by zombie bots, is the blocking of domain and public IP, consequently data communication can only be done between the local or internal network. This is certainly make the loss of cyberspace community confidence and trust.

One of example of botnet protocol types is IRC. IRC is a traditional botnet communication protocol to perform command and control after successfully infect a computers target [1]. Servers that gives command is bot master. To keep the connection with its bots, it will use IRC command and Control protocol to make communication through the network, so both of bots and bot master can connect and update bots behavior [4]. In this couples of year, IRC botnet is left and switch to botnet with peer to peer protocol. Peer to peer comunication can still communicate between botmaster and bots if only have one lost host in network topology.

Several methods for detecting botnet have been done with classification method. Classification methods have high precision, but it needs more effort to determine appropiate classification model.

We applied reinforcement learning as the new paradigm in machine learning to detect botnet using PRCL (Pursuit Reinforcement Competitive Learning). Unfortunately botnet ISCX is an unbalanced dataset, to implemented using PRCL it needs additional rule detection. PRCL have characteristic to cluster dinamic data which is come unpredictably likes networking dataset.

2. RELATED WORK

Various techniques have been implemented in this botnet detection researches. Some researches that insipire us are briefly described, such as:

Elaheh et al. [5], proposed flow based feature to detect variouse types of botnet, the features consist of 21 features and have 17 types of botnet in the testing data and 5 types of botnet in the training data. This feature was generated by combining three dataset (ISCX IDS dataset [6], ISOT dataset [7], and Botnet Traffic generated by Malware Capture Facility Project [8]) with overlay methodology [9], it is one of the most popular method to create synthethic dataset. The result of this experiments is high with detection rate 99.5% using limited number of botnet types, such as Storm, Nugache, and Waledac. But to detect all botnets dataset it has 75% of detection rate.

Saad et al. [1], used flow based features to classify P2P traffic and non-P2P traffic, while to identify the hosts with similar C&C communication used host-based features. Comparing five classification methods of Nearest Neighbor Classifier, Analitical Neural Network, Support Vector Machine, Naive Bayes Classifier, and Gausian Based Classifier. The accuration is above 90% for Support Vector Machine, Artificial Neural Network and Nearest Neighbors Classifier, then 88% true rate detection for Gausian Based Classifier and Naive Bayes Classifier.

Francisco Villegas Alejandro, Nareli Cruz Cortes, and Eleazar Aguirre Anaya[10], proposed botnet detection using clustering algorithms with K-medoids and K-means clustering. The accuration is 73.37% of K-medoids and 69.99% of K-means.

Khalid Huseynov, Kwangjo Kim and Paul D. Yoo [11], proposed a bio inspired computing technique called ant colony clustering compare to K-means clustering. It used ISOT dataset with eleven features to detect botnets attacks. The accuracy rate is above 68.8% for ant colony clustering and 82.1% for K-means clustering.

Dhavid Zhao, Issa Traore, Ali Ghorbani, Bassam Sayed, Sheerif Saad, and Wei Lu [12], proposed botnet detection based on flow interfal which work by classifying network traffic behavior using machine learning classification techniques. They used Decision Tree with REPTree to increase the detection accuracy and compared to Bayesian Network. The accuracy is above 90% for both algorithms.

G. Kirubavathi and R. Anita [13], proposed novel approach which have similar technique with David Zhao, et al. [12], to detect botnets based on traffic flow behavior analysis and machine learning techniques. To compare with three algorithm and four selected features for 14 of bots sample, the accuracy rate is 95.86% for Adabost+J48, 99.14% for Naive Bayes classification, and 92.02 for Support Vector Machine.

Sean Miller and Curtis Busby Earle[14], used supervised and unsupervised learning to know the impact of different botnet flow feature subsets on prediction accuracy which implemented using machine learning method. They proposed K-means clustering for unsupervised learning approach and Naive Bayes for supervised learning and divided three sets of features. The K-means accuration is above 90 % for Feature Set1, 99% for Feature Set2, and 90% for Feature Set3, while Naive Bayes accuration is above 90.775% for Feature Set1, 60.205% for Feature Set2, and 93.735% for Feature Set3.

Indah et al, [15], used LVQ to compare her main proposed method On-Line Clustering using PRCL and implemented it with different case study which detect various intrusion such as R2L, U2R, Probe and DoS. The dataset used is KDD Cup 1999 dataset. On-Line Clustering can clasify in realtime, have a high acuration in detecting intrusion. Based on the experiment PRCL has learning rate 0.0001, time 65 ms, and accuracy rate 99.948% for U2R, 99.865% for R2L, 99.939% for Probe dan 99.996% for DoS. This method has not been implemented in real system, it was just a simulation.

According to the related work we inspire to take up this research On-line Clustering Method with Pursuit Reinforcement Competitive Learning (with additional rule) to detect botnet.

3. ORIGINALITY

Botnet is a part of malicious software that used to harm personal data or computer and also perform cyber crime in this recent years. Recently several systems of botnet detection has been done by using classification methods. Classification methods have high precision, but it needs more effort in determinating between appropiate classification model, it takes long time to classify botnet and needs minimum 40% of training data to reach the succesfull detection. In our paper, we get data from research forum of New Brunscwick University. The data represent network traffic that contains of botnet. Botnet have unique behavior in traffic, which is the bot master always keep connecting to an address[1]. On other hand botnet dataset is a kind of data which comes every time unpredictably so that we can not stop it in a while in order to clustering. We used 6 classes of botnet types (Neris, Rbot, SMTP_SPAM, Nsis, Virut, and Zeus), which have unbalance number of data between 7 types of botnet in training data. We proposed competitive learning approach to detect botnet with On-line Clustering using Pursuit Reinforcement Competitive Learning with additional rule that execute data as realtime classification with high accuration in detecting botnet. PRCL has characteristic that the data with minimum number would be classified into another class and it would made high error rate detection. Therefor the system needs additional rule for classifying class data that have minium number (less than 5000 data) and reduce the error rate. PRCL, is supposed to detect new botnet in realtime with higher speed and accuracy than previous research and can help network administrators to detect botnet which has unpredictable behavior in network traffic.

4. ADOPTED PREVIOUS METHOD

The system design of botnet detection with Pursuit Reinforcement Competitive Learning adopted 3 method from the previous research. The further explanation of the adopted methods are described in pont 4.1, 4.2 and 4.3.

4.1 Flow Based Features

Several researches used flow based method to detect botnet [5],[6],[12],[13],[14], which the features are extracted based on network flow in a period time. The definition of flow, is a set of pair of connection between two host and port in a period time. This pair are source ip-destination ip and source port-destination port.

Previously, research with flow based features is not clear[5], therefor feature selection and grouping unique feature was done in order to eliminate the redundant feature. There are four groups of feature, as follow:

a.	Byte-based	: TBT, APL, DPL, PV
b.	Time-based	: BS, PS, PPS, AIT
c.	Behavior-based	: Reconnect, Duration, FPS
d.	Packet based	: PX, NNP, NSP, IOPR, PSP

The groups of feature would be combined with protocol types, while for source ip, destination ip, source port and destination port is not choosen because in universal botnet detection it has known as inefficiency feature. This phase shown which is the worst group would be romoved to increase the accuracy of botnet detection.

4.2 Support Vector Machine (SVM) Algorithm

Support Vector Machine is part of pattern recognition. SVM is a machine learning method to find the best hyperplane which separate two classes in inout space. To find the best hyperplane between two classes can be did by measuring hyperplane margin dan get the maximal point. Margin is a distance between hyperplane and the nearest pattern in each class. The nearest pattern is called as support vector. The main phase of SVM learning process is to find hyperplane location, is shown in Figure 1.

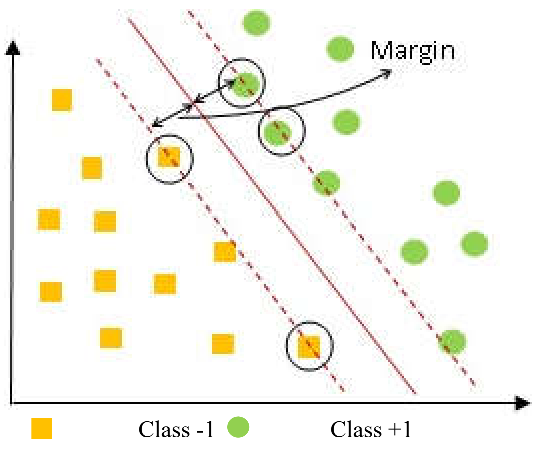

Figure 1. Finding the best hyperplane between two classes in SVM

Solid line in Figure 1. shows the best hyperplane. The best hyperplane is in the middle of two classes, while the yellow and green color in black circle is support vector.

4.3 Learning Vector Quantization Algorithm

LVQ (learning Vector Quantization) is one example of competitive learning, working to find how the data is clustered using network discover structure in data. Learning Vector Quatization also called Simple Competitive Learning [15], which is a simple On-line Clustering method. The algorithm of LVQ are describes as follow:

1. Choose the number of clusters N
2. Initialize the prototypes $w_1 * \ldots w_n *$ (1)
3. Repeat until stoping criterion is satisfied:
4. Randomly pick an input x
   - Determine the winning unit for k for all i using vector:

     $$|w_k - x| \leq |w_i * - x| \quad (2)$$

   - Update the winning prototype weights as follows:

     $$w_k * = w_k + \alpha(x - w_k *) \quad (3)$$

   Where α is learning rate and x is new data.

5. SYSTEM DESIGN

The system design is divided by 3 phases, which are the first is Data Pre-procesing, the second On-Line Clustering and the third is Evaluation. Figure 2. shows the proposed block diagram system.

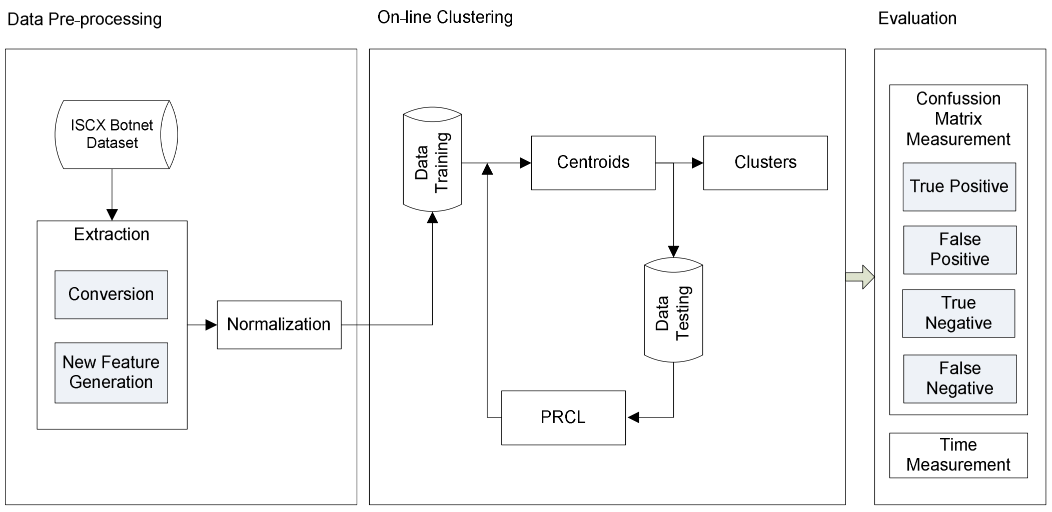

Figure 2. Block diagram system

Preprocessing data is the first step that have to do which take more than 50% effort in datamining. The result of preprocessing data become the input in online clustering. In this phase we applied rule detection in PRCL methods. The evaluation is done after finishing the first and second process.

5.1 Data Pre-processing Phase

The process of pre-processing consists of three sub-sections, they are: PCAP dataset, extraction, and Normalization.

The dataset used is dummy Botnet ISCX dataset, is made by some researcher from New Brunswick University in Canada[5]. Dataset Botnet ISCX is created by merging existing datasets are ISOT datasets, ISCX IDS and Botnet Traffic generated by Malware Capture Facility Project, using the overlay methodology. Overlay methodology is considered as a method that can provide the accuracy of the classification dataset and to estimate the level of precision and recall, making it attractive for researchers to be implemented in the research. Implementation of the overlay method is to combine datasets and benign malicious data by mapping the data to either existing machines. Because a very wide range of IP, the scenario for mapping the datasets using Bitwist, then replayed using TCP replay and capture using a TCPdump into a single dataset. This dataset localized for research which consist of two sets of data, namely the training data capacity of 5 GB and Data Testing capacity of 2 GB. The testing dataset consist of seventeenth classes, while the training data contains seven types of botnets are Neris, Rbot, Virut, NSIS, SMTP Spam, Zeus and Zeus Control.

Features extraction are the major steps that must be performed on the data pre-processing explained in Figure 3.

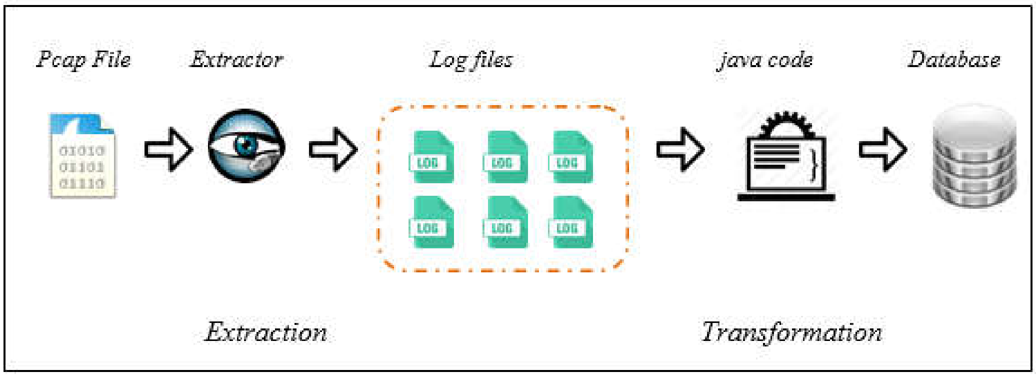

Figure 3. Features extraction and transformation data

The features extraction determine how consistence and well structure dataset are modelled. The pcap file is extracted to log files using extractor tools with connection logs approach on the dataset. The log files must be stored to database in purposed for creating new features. The features creation is implemented based on network flow. First of all we made the data model using pair of source ip, destination ip and source port, destination port in a period time to extract new features. The new created features are the result of reading and analyzing packet headers and payload data using some formulas that is explained in Table 1.

The new created features will be merge with the exctracted features to be unsupervised dataset. Based on the proposes method, the data requires a label that can represent each botnet behavior. The labeling process is done by comparing amount of any attack in dataset, calculating and analyzing features that are related to each other. The dataset with label or supervised dataset ready for clustering after validating process.

Table 1. The formulas of features creation

No	Attributes	Descriptions	Formulas	Type
1	PX	Total Number of Packet Exchange	*(src_pkts+ dst_pkts)*	flow
2	NNP	Number of Null Packet Exchange (packet with zero payload size)	*(if (src_byte+ dst_bytes=0) then nnp=1, else nnp=0)*	flow
3	NSP	Number of small packet exchange	*(if (src_bytes+ dst _bytes)>= 63 && (src_bytes+ dst_bytes) <= 400, then nsp=1, else nsp=0)*	flow
4	PSP	Precentage of small packet exchange(have length between 63 and 400 bytes)	*(nsp/px)*	flow
5	lOPR	The ratio between the number of Incoming Packets Over the number of outgoing packets	*(src_pkts/ dst_pkts)*	flow
6	Reconnect	Number of reconnection	*if history like 'Sr%', then reconnection=1*	flow
7	FPS	The length of the first packet	*(src_bytes/src_pkts)*	flow
8	TBT	Total number of bytes per flow	*(src_bytes+ dst_bytes)*	flow
9	APL	Average packet length per flow	*(src_bytes+ dst_bytes)/total_packet*	flow
10	DPL	The total number of subsets of packets of the same length over the total number of packets in the same flow	*(number packet with same length divided by all packets in group)*	flow
11	PV	Standard deviation of payload packet length	*(sqrt((1/ total data in group) x ((src_bytes+ dst_bytes)-APL)^2))*	flow
12	BS	Average bit per second	*((src_bytes+ dst_bytes) x 8) / duration*	flow
13	PS	Average packets persecond in time window	*(src_pkts+dst_pkts)/ duration*	host
14	AIT	Average inter arrival time of packets	*(total(src_pkts+ dst_pkts))/(max time- min time)*	host
15	PPS	The total number of bytes of all the packets over the total number of packets in the same flow	*(src_pkts+ dst_pkts)/ duration*	flow

Normalization is used for making the dataset into the same value, smoothing the noisy data, transforming and reducting data. The purposes of data normalization is to create a uniform value of a large data range, by scaling the value attribute of the data, so it can fall in a certain range . We used sigmoidal normalization method, which have range of data between -1 and 1 to transform the dataset. The reason for implementing sigmoidal on proposed method, because the minimum dataset value is less than 1 for duration features and the maximum value is more than 1000 for average packets per-second feature. Therefor we transform the data in order to get higher rate of detection.

5.2 On-line Clustering

On-line Clustering is one of clustering method for dynamic data [16]. We applied Reinforcement learning using Pursuit Reinforcement Competitive Learning with additional rule detection. The rule is used for detecting botnet which have class less than 5000. The rule have many variation according to the particular botnet type, so if there is a different type of botnet it will add a new rule. Figure 4. describes how the rule takes contribution to detect botnet.

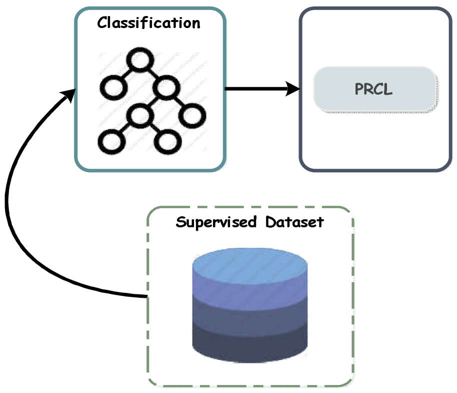

Figure 4. Additional rule to detect botnet with class less than 5000

The rule uses classification method with decision tree algorithm. In this phase the supervised dataset will be classified into 4 class, class 1 for nsis , class 2 for virut, class 3 for zeus, and class 4 for combination of Neris, Rbot and SMTP_ spam. Class 1, 2 and 3 is botnet that has small class less than 5000 class. While class 4 is botnet that has class more than 5000.

The class 4 is implemented to Pursuit Reinforcement Competitive Learning (PRCL) [17], which used winning weight factors to determine the cluster. The algorithm of PRCL are describes as follow:

1. Determining the winning unit i^* from:

$$d_{i^*} = \arg\min_i d(x, w_i) \tag{4}$$

2. Update the reward track of x for all weights :

$$r(x, w_{ij}) = r(x, w_{ij}) + \beta(1 - r(x_j, w_{ij})) \text{ if } i = i^* \tag{5}$$

and

$$r(x, w_{ij}) = r(x, w_{ij}) + \beta(0 - r(x_j, w_{ij})) \text{ if } i \neq i^* \tag{6}$$

3. Choose the winning i^* from maximizing the reward as:

$$w_{i^*} = \arg\max_i r(x, w_i) \tag{7}$$

4. Update the winning prototype weights as follows:

$$\Delta w_{ij} = \alpha(x_{ij} - w_{ij}) \text{ if } i = i^* \tag{8}$$

and

$$\Delta w_{ij} = 0 \text{ if } i \neq i^* \tag{9}$$

where α is learning rate, β is reward rate, d is distance, r is reward, and x is new data.

SVM and LVQ algorithms in previous sections, are used to compare the experimental results of PRCL. PRCL has the same algorithm in equation no 4, 8 and 9 with LVQ, which distinguishes it on the side of PRCL there is reward and punishment.

5.3 EVALUATION

Evaluation is estimation stage of the proposed system whether it is consistence to the goals to be achieved. Evaluation process in a research is used to find out how accurate the classification method used and the computational level of the experimental results. Measurement of evaluation using confusion matrix such as True Positive (TP), False Negative (FN), True Negative (TN), and False Positive (FP). The experimental and evaluation result of proposed systems shows that PRCL can achieve high accuracy and fast measurement time with the average less than 1 second.

6. IMPLEMENTATION

The desain system that has been made is implemented in this phase. The implementation describes the system requirement and specification of computer and libraries that is used. We implemented the proposed system to the requirement in this following Table 2.

Table 2. System requirement

Requirement	**Specification/ Version**
Operating System	Windows 7 64-bit
Processor	Intel core i5-3230M
RAM	8 GB
Libraries : 1. Postgresql jdbc 2. ALI lib	 9.3-1103.41 11.01.16

All of the proposed algorithm is implemented in computer with Windows 7-64 bit operating system, intel core i5 processor and 8 GB RAM. The postgresql library is java library for extracting feature and helping to store the extraction result to postgresql database. The ALI lib is java library for inteligent computing such as measure the distance between two vector, calculating centroid and etc.

7. EXPERIMENT AND PERFORMANCE EVALUATION

Confussion matrix is used to evaluate the performance of the proposed system by counting the accuracy [15]. A confusion matrix of binary classification is a two by two table formed by counting of the number of the four outcomes of a binary classifier. Table 3 and 4 explains the definition and description of confussion matrix, such as: True Positive (TP), False Negative (FN), True Negative (TN), and False Positive (FP).

Table 3. Confusion matrix measurement

Confusion Matrix			
		Predicted Class Botnet	
		Yes	**No**
Actual Class Botnet	**Yes**	True Positive	False Negative
	No	False Positive	True Negative

Table 4. Confusion matrix definitions

Confussion Matrix	**Definitions**
TP	TP occurs if the message is true value in diagnosing the correct data
TN	TN occurs if the message is incorrect value in diagnosing the wrong data
FP	FP occurs if the message is true value in diagnosing the wrong data
FN	FP occurs if the message is incorrect value in diagnosing the correct data

Generally to evaluate the performance of classification using F-measure, recall and precision. Precision and recall is used for measuring the performance of binary classification test. Recall is the presentage of positive labeled which are corectly recognized. The followings formulas 10, 11, 12, and 13 decribes how F-Measure, recall, precison and accuracy equation are.

$$\Pr ecision = \frac{TP}{TP + FP} \tag{10}$$

$$\operatorname{Re} call = \frac{TP}{TP + FN} \tag{11}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$F - Measure = \frac{2 * \mathrm{Pr}ecision * \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call} \tag{13}$$

To evaluate the proposed system, we used accuracy which defines presentage of corect classified over the total number of instance.

In this paper we compared PRCL with Learning Vector Quantization and SVM that have proposed by previous researcher [1], [13], [14]. The experiment is divided by three phases. The first is experiment uses 3 types of botnet, second is experiment uses 6 types of botnet without additional detection rule, and the third is experiment uses 6 types of botnet with additional detection rule. PRCL and LVQ will be tested using learning rate (alpha)=0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001, 0.000005, 0.000001, and data composition is order by timestamp.

We implemented PRCL, LVQ and SVM using 10% of botnet supervised dataset for training and the composition of supervised dataset consist of 44599 data. The number of cluster for Zeus (50), Nsis (639), Virut (1544), Neris (10042), Rbot (22484), and SMTP_Spam (9840).

7.1 Experiment using Three Types of Botnet

The first experiment predicted three botnet class for Neris, Rbot and SMTP_Spam. The composition of supervised dataset using three types of botnet consist of 43286 data, and the number of cluster 1=Neris, 2=Rbot, and 3=SMTP_Spam.

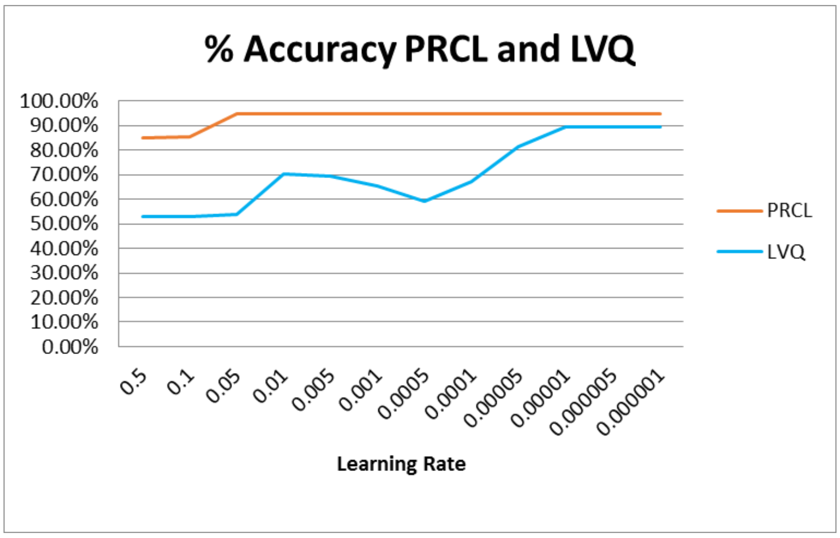

Figure 5. Accuracy of botnet detection using PRCL and LVQ

The accuracy evaluation of botnet detection using PRCL and LVQ is ilustrated in Figure 5. The experimental result explaines that the accuracy of all class of botnet stable when the learning rate 0.00001, shows in Figure 5. The average accuracy of botnet using PRCL is the most stable for each learning rate, which is start to be stable when learning rate 0.05. The experimental result of botnet with LVQ is always have lower average accuracy for each rate and stable when learning rate 0.00001. The experimental results of botnet using PRCL and LVQ explaines that the smaller learning rate, the more stable the accuracy of botnet detection.

Figure 5. shows the average accuracy for each botnet type using PRCL and LVQ, by measuring the total accuracy of a method over total number of botnet data. Figure 6. shows detail accuracy for each type of botnet using PRCL and LVQ algorithm.

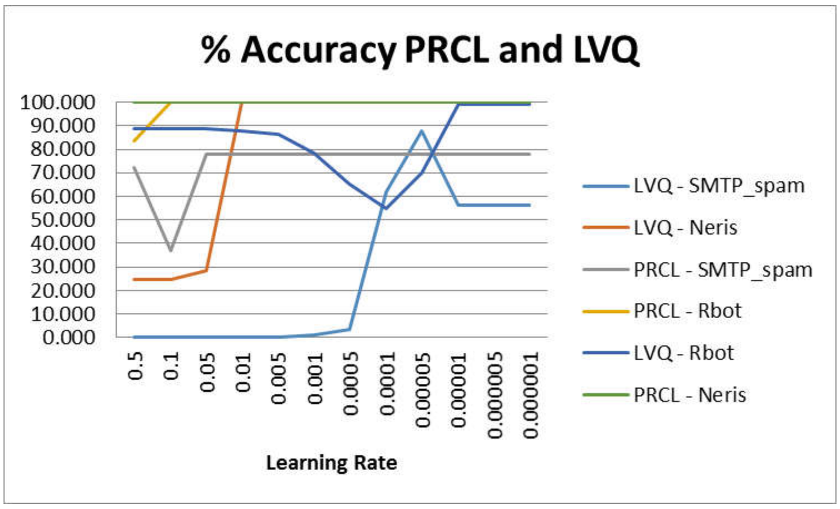

Figure 6. Accuracy of three types of botnet using PRCL and LVQ

The experimental results using three types botnet show that accuracy for PRCL is more stable than LVQ. PRCL achieves high rate accuracy when learning rate 0.5 (for Neris), 0.1 (for Rbot), and 0.05 (for SMTP_spam). While LVQ has a very fluctuating accuracy rate, it is showed by a new valley before it achieves stable accuracy. LVQ achieves stable accuracy when learning rate is 0.01 (for Neris) and 0.00001 (for Rbot and SMTP_spam). The accuracy of SMTP_spam is the lowest compared to other botnet types, while Neris is the highest.

7.2 Experiment using Six Types of Botnet Without Additional Rule Detection

The second experiment predicted six class for Neris, Rbot, SMTP_Spam, Nsis, Virut and Zeus without additional rule. The composition of supervised dataset consist of 44599 data, and the number of cluster 1=Zeus, 2=Nsis, 3=Virut, 4=Neris, 5=Rbot, and 6=SMTP_Spam.

The experimental results using six types botnet in Figure 7., show that accuracy for PRCL is more stable than LVQ. PRCL achieves high rate accuracy when learning rate 0.5 (for Zeus), 0.05 (for Rbot), and 0.00001 (for Neris).

While for SMTP_spam, Virut, and Nsis, PRCL achieves low accuracy which is not more than 10% rate.

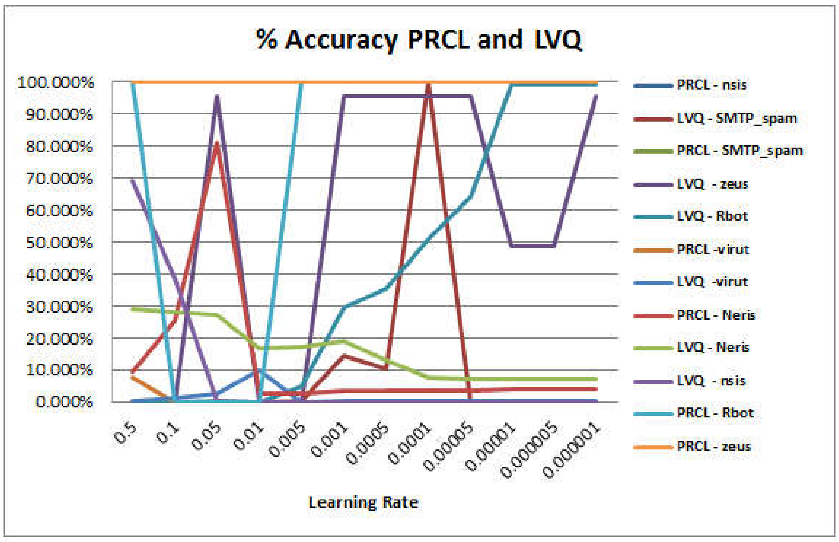

Figure 7. Accuracy of six types of botnet using PRCL and LVQ without rule addition

LVQ achieves stable accuracy when learning rate is 0.01 (for Virut), 0.00001 (for Rbot) and 0.00005 (for SMTP_spam). There are two characteristic of data implementation using LVQ, for Zeus and Rbot the data get higher accuracy in the triple experiments. The accuracy of SMTP_spam is the highest with 100% accuracy and the get down to 0.034 %.

7.3 Experiment using Six Types of Botnet With Additional Rule Detection

The third experiment predicted six class for Neris, Rbot, SMTP_Spam, Nsis, Virut and Zeus with additional rule. The composition of supervised dataset consist of 44599 data, and the number of cluster 1=Zeus, 2=Nsis, 3=Virut, 4=Neris, 5=Rbot, and 6=SMTP_Spam.

The third experiment is impelemented PRCL combine with rules to increase the accuracy of botnet detection. The rules is used to detect Zeus, Nsis, and Virut wich has the total data less than 5000.

The accuracy evaluation of botnet detection using PRCL and LVQ with additioanl rule is ilustrated in Figure 8. The experimental results show that PRCL started to stable for learning rate 0.05 and accuracy more than 90%. PRCL achieves 85% accuracy when learning rate 0.5 and increase to 85.5% when learning rate 0.1. The graph of LVQ accuracy results is in valleys and hills forms. LVQ has a significant increase of accuracy rate from 70% when learning rate 0.5 to 87.6% when learning rate 0.00001. LVQ achieves 69.8% when learning rate 0.05 and get higher rate for 78% when learning rate 0.01 before decrease to 72.4% when learning rate 0.0005. Accuracy rate of LVQ started to increase again when learning rate 0.0001 achieves 76.4% and when learning rate 0.00005 achieves 83.6%.

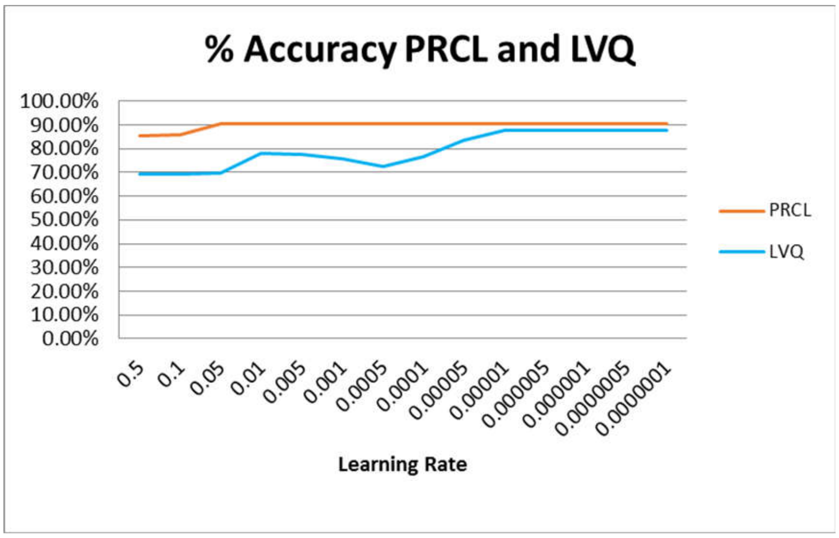

Figure 8. Accuracy of six types of botnet using PRCL and LVQ with additional rule

The detail accuracy for each botnet types of Neris, Rbot and SMTP Spam is similar to Figure 6, while the detail accuracy of Zeus, Nsis and Virut is described in Figure 9.

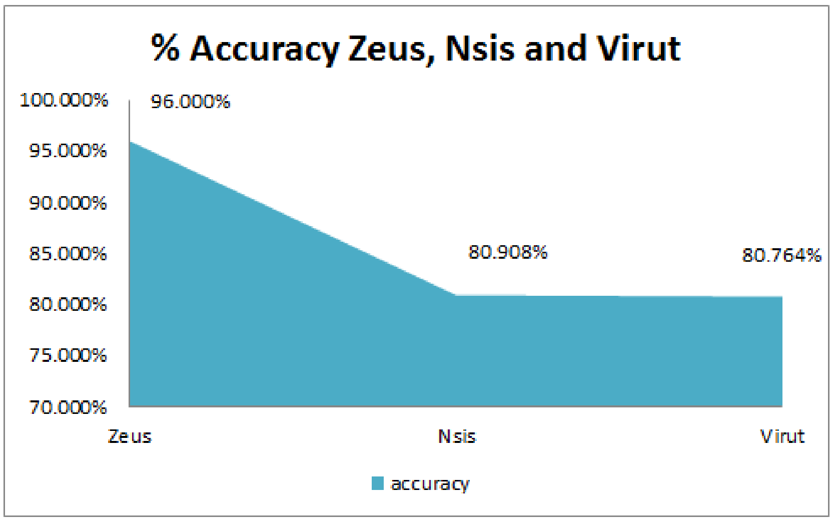

Figure 9. Accuracy of Zeus, Nsis, and Virut using PRCL and LVQ with additional rule

The experiment result of PRCL and LVQ is asumed simillar for Nsis, Virut and zeus botnet types. The reason, Botnet ISCX is unbalanced dataset therefor to detect it needs additional rule using classification method. PRCL and LVQ is implemented using the same rules to detect botnet types of Nsis, Vrut and Zeus. The highest accuracy rate is Zeus, it achieves 96% and the lowest is Virut which achieves 86.76%. While Nsis achieves 80.9% accuracy rate. The analysis result describes the implemented rule, accuracy rate is not affected by number of data but the character attributes used.

The accuracy results show that PRCL is more stable than LVQ, and the smaller the learning rate the more accurate the result of accuracy. The average accuracy of PRCL and LVQ stable when learning rate 0.00001, it achieves 90% accuracy rate for PRCL and 87.6% accuracy rate for LVQ.

The following Table 5., explained the comparison accuracy and processing time between PRCL, LVQ and SVM result.

Table 5. Comparison of accuracy and processing measurement (ms) time of botnet detection using PRCL, LVQ and SVM

Technique	Neris (%)	Rbot (%)	SMTP Spam (%)	Nsis (%)	Virut (%)	Zeus (%)	Time (ms)
PRCL (On-Line Clustering)	100%	99.9%	78%	-	-	-	158
LVQ (On-Line Clustering)	99.9%	99.3%	56.3%	-	-	-	130
SVM (Classification) with Three Types Botnet	99%	99.6%	99.8%	-	-	-	4000
PRCL (On-Line Clustering) without Rule	3.84%	99.96%	0%	0%	0%	100%	165
LVQ (On-Line Clustering) without Rule	7.269%	99.8%	0.03%	0%	0.36%	95.6%	142
SVM (Classification) 10% training data	99%	99.6%	99.8%	0%	34.6%	0%	12000
PRCL (On-Line Clustering) with Rule	100%	99.9%	78%	80.9%	80.7%	96.0%	175
LVQ (On-Line Clustering) with Rule	99.9%	99.3%	56.3%	80.9%	80.7%	96.0%	156
SVM (Classification) 80% training data	99.9%	99.9%	99.6%	0%	26.5%	0%	22740

The experimental result of PRCL and LVQ method explaines the smaller learning rate, the accuracy will be higher. PRCL achieves high accuracy (100% for Neris, 99.9% for Rbot and 78% for SMTP_Spam) when learning rate (alpha) =0.05, beta=0.5 and data composition order by timestamp ascending using three types of botnet in the first experiment. LVQ achieves high accuracy (99% for Neris, 99.3% for Rbot and 56.3% for SMTP_Spam) when learning rate (alpha) =0.00001. SVM achieves high accuracy (99% for

Neris, 99.6% for Rbot and 99.8% for SMTP_Spam). The seccond experiment without Rule addition PRCL achieves unstable accuracy (3.84% for Neris, 99.96% for Rbot, 0% for SMTP SPAM, 0% for Nsis, 0% for Virut, and 100% for Zeus). LVQ also achieves unstable accuracy (7.269% for Neris, 99.8% for Rbot, 0.03% for SMTP SPAM, 0% for Nsis, 0.36% for Virut, and 95.6% for Zeus). SVM achieves high accuracy using 10% training data (99% for Neris, 99.6% for Rbot and 99.8% for SMTP SPAM) but lower accuracy (0% for Nsis, 34.6% for Virut and 0% for Zeus). The third experiment using additional rule PRCL achieves high accuracy (100% for Neris, 99.9% for Rbot, 78% for SMTP_Spam, 80.9% for Nsis, 80.7% for Virut, and 96.0% for Zeus). LVQ achieves high accuracy (99% for Neris, 99.3% for Rbot, 56.3% for SMTP_Spam, 80.9% for Nsis, 80.7% for Virut, and 96.0% for Zeus). SVM achieves high accuracy (99.9% for Neris, 99.9% for Rbot and 99.6% for SMTP SPAM) but lower accuracy (0% for Nsis, 26.5% for Virut and 0% for Zeus) using 80% of training data. The result of comparison explains that:

- The accuracy of botnet detection using LVQ almost same with PRCL but PRCL is more stable than LVQ beside LVQ is faster than PRCL.
- The Accuracy of botnet detection using PRCL (Neris, Rbot and SMTP SPAM botnets types) is lower than SVM but PRCL faster than SVM.
- The accuracy of Botnet Detection using PRCL for Neris and Rbot is the highest but PRCL is lower for SMTP_spam than SVM.
- The accuracy without additional rule to detect six types of botnet is lower than using additional rule, especially in detecting Nsis, Virut and Zeus botnet types.
- The additional Rule experiment achieves the same accuracy for PRCL and LVQ to detect Nsis, Virut, and Zeus botnets types because the rule which is used also same for both PRCL and LVQ.

The proposed method system PRCL outperforms other methods because it gains better performance evaluation than LVQ and SVM in terms of accuracy and time processing. PRCL have punishment and reward system, therefore the system in PRCL will always study automaticaly to update the winning centroid to gains better result of prediction.

Other experimental result of botnet detection is the resource usage of memory and CPU data, which is one of performance testing in software engineering. The resource usage of each process have different data, this is infuenced by how each process works. The system design of the proposed method have 3 phases, which are the first is data pre-procesing, the second is online clustering and the third is evaluation. Resource Usage retrievel needs 10 second to know the graph fluctuation of memory and CPU usage. Figure10., describes the resource usage of CPU and figure 11., describes the memory usage when the process is running in each step of pre-processing, online clustering and evaluation.

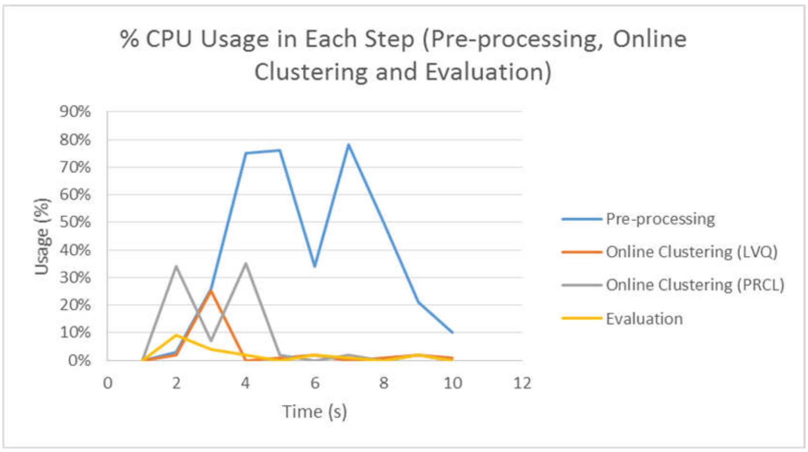

Figure 10. The CPU usage when the process is running in each step (pre-processing, online clustering and evaluation)

The Pre-processing step has fluctuatif graph and takes higher CPU usage than the other steps which is the highest percentage 78% while the time is 7 second. The PRCL online clustering takes the highest percentage 34% while time 2 and 5 second. The LVQ takes lower CPU usage than PRCL which is the highest percentage 25% while the time is 3 second and become stable in 4 second. The lowest percentage among all steps is evaluation, it takes the highest percentage 9% while time is 2 second and become stable in 4 second.

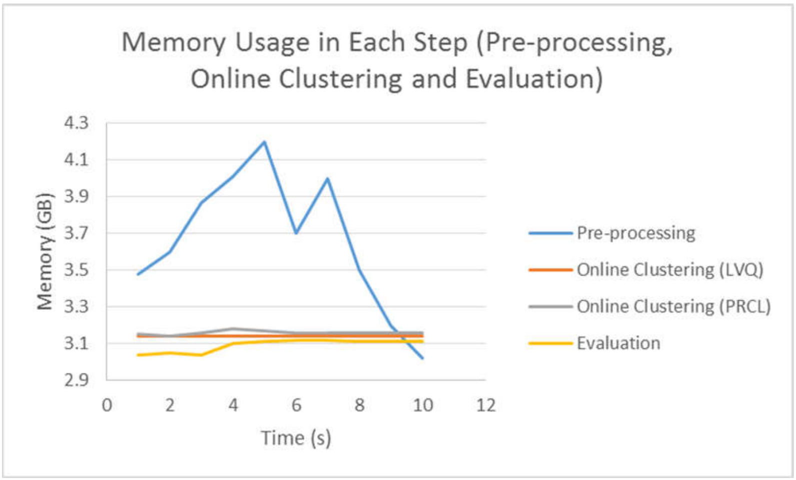

Figure 11. The memory usage when the process is running in each step (pre-processing, online clustering and evaluation)

The memory usage of pre-processing step has a fluctuatif graph which there are hill and valey in this graph. The highest memory usage of pre-processing step is 4.2 GB while in 5 second, then it has decrease become 3.7 GB in 6 second. The pre-processing step increases in 7 second before will decreaces again become 3.0 GB in 10 second. The memory usage of online clustering (LVQ) step is the most stable among all the steps which is from 1 second untill 10 second the usage has 3.14 GB. The online clustering (PRCL) step has the highest memory usage 3.18 in 4 second and get decrease to 3.16 GB in 6 second. The lowest memory usage of evaluation step is 3.04 GB in the first time and get increase to 3.11 GB in 4 second.

The highest CPU and memory usage of preprocessing step is 78 % and 4.3 GB, while the online clustering is divided into two methods. LVQ is lower (the highest percentage 25% on CPU and 3.14 GB on memory usage) than PRCL (the highest percentage 34% on CPU and 3.18 GB on memory usage). The third step is evaluation, which has 23% on CPU and 3.11 GB on memory usage. Further explaination of the comparison result are:

- The preprocessing step has the highest resource usage because it is consist of three sub step, which is the extraction process needs more resources and has complex computation.
- LVQ method needs lower resource usage than PRCL, because it has more simple algorithm to discover the new centroid based on the shortest distance while PRCL has reward and punishment to discover the new winning centroid.
- The evaluation has the lowest resource usage because of the simple computational process using confusion matrix (True Positive, True Negative, False Positive, and False Negative).

8. CONCLUSION

This paper presents a new method of reinforcement learning to detect botnet using On-Line Clustering which can perform clustering in real time with high accuracy in detecting botnet. To perform On-Line Clustering, we use Pursuit Reinforcement Learning. PRCL use unsupervised dataset to cluster the botnet and to get the accuracy form the evaluation the dataset match with the supervised data or data with label. It means the PRCL itself can perform clustering in real time and combine with pcap dump.

From experimental study, the proposed method achieves high accuracy in the first experiment using PRCL (100% for Neris, 99.9% for Rbot and 78% for SMTP_Spam) when learning rate (alpha) =0.05, beta=0.5 and high speed (158 ms). LVQ achieves high accuracy (99% for Neris, 99.3% for Rbot and 56.3% for SMTP_Spam) when learning rate (alpha) =0.00001 and high speed (130 ms). SVM achives high accuracy (99% for Neris, 99.6% for Rbot and 99.8% for SMTP_Spam) and low speed (4000 ms).

The additional rule detection achieves higher and more stable accuracy than without rule additional to detect six types of botnet. PRCL achieves high accuracy (100% for Neris, 99.9% for Rbot, 78% for SMTP_Spam, 80.9% for Nsis, 80.7% for Virut, and 96.0% for Zeus) when learning rate= 0.05, beta=0.5 and high speed (175 ms). LVQ achieves high accuracy (99% for Neris, 99.3% for Rbot, 56.3% for SMTP_Spam, 80.9% for Nsis, 80.7% for Virut, and 96.0% for Zeus) when learning rate =0.00001 and high speed (156 ms). SVM achieves high accuracy (99.9% for Neris, 99.9% for Rbot and 99.6% for SMTP

SPAM) but lower accuracy (0% for Nsis, 26.5% for Virut and 0% for Zeus) and speed (22740 ms) using 80% of training data. Meanwhile the step of CPU and memory usage which are 78 % and 4.3 GB for pre-processing, 25% and 3.14 GB for online clustering (LVQ), 34% and 3.18 GB for online clustering (PRCL), and 23% and 3.11 GB evaluation

The accuracy of botnet detection using PRCL is higher and more stable than LVQ but LVQ is faster than PRCL. We also compare PRCL and SVM in this paper. The experimental results explained that the accuracy of PRCL is lower than SVM especially to detect SMTP_Spam type but higher for Neris, Rbot, Nsis, Virut and Zeus. PRCL also faster than SVM in detecting botnet with time of processing 158 ms using three types of botnets and 175 ms using six types of botnets.

In this paper our proposed reinforced approach with additonal rule shown high performance in speed and accuracy comparing to the other algorithm in detecting botnet. It is prove that PRCL could be used to data that needs fast detection is like botnet dataset.

9. FUTURE WORK

Our future work is feature selection, then implemented it using PRCL. The feature selection would be increase the accuracy and speed. Therefor it can detect dataset that is need fast classification such as botnet dataset and have high rate accuracy.

Acknowledgements

We would like to thank University of New Brunswick for permits to use the Botnet ISCX Pcap Dataset in our research.

REFERENCES

[1] S. Saad et al., “Detecting P2P Botnets through Network Behavior Analysis and Machine Learning,” *Ninth Annu. Int. Conf. Privacy, Secur. Trust*, pp. 174 – 180, 2011.

[2] C. Chen and H. Lin, “Detecting botnet by anomalous traffic,” *J. Inf. Secur. Appl.*, vol. 21, pp. 42–51, Apr. 2015.

[3] D. Garant and Wei Lu, “Mining Botnet Behaviors on the Large-Scale Web Application Community,” in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, 2013, pp. 185–190.

[4] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, “Botnet Detection Based on Network Behavior,” in *Botnet Detection*, vol. 36, no. August, Boston, MA: Springer US, 2008, pp. 1–24.

[5] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, “Towards

effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security, CNS 2014*, 2014, pp. 247–255.
[6] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, 2013.
[7] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[8] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.
[9] A. J. Aviv, "Challenges in Experimenting with Botnet Detection Systems," *USENIX 4th CSET Work. San Fr. CA*, pp. 1–8, 2011.
[10] F. V. Alejandre and N. C. Cort, "Botnet Detection using Clustering Algorithms," vol. 118, pp. 65–75, 2016.
[11] K. Huseynov, K. Kim, and P. D. Yoo, "Semi-supervised Botnet Detection Using Ant Colony Clustering," vol. 31, no. The 31th Symposium on Chryptography and Information Security Kagoshima, pp. 1–7, 2014.
[12] D. Zhao, I. Traore, A. Ghorbani, B. Sayed, S. Saad, and W. Lu, "Peer to Peer Botnet Detection Based on Flow Intervals," *Inf. Secur. Priv. Res.*, vol. 3, no. 1, pp. 87–102, 2012.
[13] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics R," *Comput. Electr. Eng.*, vol. 50, pp. 91–101, 2016.
[14] S. Miller and C. Busby-earle, "The Impact of Different Botnet Flow Feature Subsets on Prediction Accuracy Using Supervised and Unsupervised Learning Methods," vol. 5, no. 2, pp. 474–485, 2016.
[15] I. Y. P. Tiyas, A. Barakbah, T. Harsono, and A. Sudarsono, "Intrusion Detection with On-line Clustering Using Reinforcement Learning," in *Proceeding The Third Indonesian-Japanese Conference on Knowledge Creation and Intelligent Computing*, 2014, pp. 30–37.
[16] A. Barakbah, "Special Issues on Clustering," Knowledge Engineering Research Group PENS, Ed. EEPIS, 2016, pp. 1–80.
[17] A. Barakbah and K. Arai, "Pursuit Reinforcement Competitive Learning," in *The 2nd International Seminar on Information and Communication Technology Seminar (ICTS)*, 2006.