Asian Journal of Information Technology 15 (4): 670-675, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Scalable Real Time Botnet Detection System for Cyber-Security

V. Vanitha, V.P. Sumathi, Sindhu Arumugam and Nandhini Selvam
Department of Computer Science and Engineering,
Kumaraguru College of Technology, Coimbatore, India

Abstract: Malicious malware can exploit vulnerabilities in the internet computing environment without the user's knowledge. Today, different types of malware exist in the Internet. Among them one of the malware is known as botnet which is frequently used for many cyber attacks and crimes in the Internet. The aim of this study is to develop a scalable botnet detection framework which will be able to identify and remove stealthy botnets from the real-world network traffic. 'Storm' real time, distributed, reliable, fault-tolerant software is used in this work for analyzing the streams of data. Experimental results show that random forest has higher accuracy rate than fuzzy c-means but clustering algorithm is useful to detect the botnet in real time processing.

Key words: Security, botnet, random forest, fuzzy c-means, traffic analysis

INTRODUCTION

The evolution of information technology has tremendously changed our generation's life style. Nowadays, this greatest invention gives big influence to human life covering simplest activities like buying the grocery to the big issues like developing the regional economic or fighting for the global peace. Due to this, over dependence of human on the use of the Internet has not only lead to excitement and amenities but also exposes users to unpredictable criminals. This virtual threat not only attacks individuals but may also be targeted at larger communities like countries.

Cyber criminals are capable of launching a sudden attack on any network infrastructure by isolating the hosts depending on their interests such as financial institution or political bodies. An internet threat occurs when a group of compromised computers or botnets have been controlled by a mastermind from an unknown destination around the globe through the dominance of distributed behavior to launch a cyber attack.

Botnets have become the biggest threats on the Internet and are used for launching attacks and committing fraud by the hackers. A study shows that on a typical day, about 40% of the 800 million computers connected to the Internet are connected to some botnet. Those infected machines engage in many illegitimate activities, like distributing spam, stealing sensitive information, launching denial-of-service attacks and spreading new infections. This infection can sneak in a number of ways including opening an email attachment, visiting a dangerous web site or clicking a pop-up window while visiting a web site. A botnet can cover an extremely large geographic area, limited only by the availability of internet access. Hence, it is imperative to detect and prevent the bots in order to avoid the malicious behaviors both in the host and network.

The architecture of botnet is based on how they communicate with the bots. It is classified into three types, they are centralized, decentralized and hybrid (Silva *et al*., 2013). In a centralized architecture, all the bots report to and receive commands from a single C and C server whereas in decentralized architecture, the communication between bots and server or among the bots will be of peer to peer contact. Hybrid architecture is the combination of the centralized and decentralized structure. Bots are classified into four types based on their architectural design namely, Internet Relay Chat (IRC) bot, Pear-to-Pear (P2P) bot, HTTP bot and DNS bot. The IRC bots, follow the PUSH approach as they connect to selected channels and remain in the connect state till it receives command from botmaster. The P2P bot architecture, instead of having a central C and C server, the botmaster attack the target host by sending a command to one or more bots and they deliver it to their neighbors. The HTTP bot uses the HTTP protocol to send the commands via web servers which enables these bot to periodically visit certain web servers to get updates or new commands. The DNS bots are the bots which make use of DNS to receive commands to perform malicious activities.

When malicious attacks have fixed patterns, they can be easily identified by matching these patterns. However, sophisticated attacks with changing patterns are

Corresponding Author: V. Vanitha, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, India

distributed over the internet and have fewer common characteristics. The advances in transmission methods, anti-detection techniques and means of concealment have made botnets more difficult to detect and prevent. In addition, Botnets make use of encryption and P2P protocols to evade traditional pattern matching-based detection techniques.

Methods for detecting bots can generally be divided into two categories, those that involve static analysis or checking computers' characteristics against a list of known threats and those that involve behavioral analysis or monitoring communications in a network for behaviors that are known to be exhibited by known botnets. Static analysis results in more reliable judgments but requires threat signatures that are current and available. Behavioral analysis potentially allows for much broader detection methods especially by aggregating information from multiple sources but is more likely to result in false positives. Effective botnet detection strategies generally involve aspects of both static analysis and behavioral analysis. The researchers (Dietrich *et al*., 2013; Zhao *et al*., 2013; Lu *et al*., 2011) used machine learning for detection which require labeled P2P botnet data to train a statistical classifier. Unfortunately, acquiring such information is a challenging task, thereby drastically limiting the practical use of these methods.

An adaptive blacklist-based packet filter using a statistic-based approach was developed by Yuxin Meng and Kwok (2014) to improve the performance of signature-based botnet detection. A behavior-based botnet detection system based on fuzzy pattern recognition technique was proposed by Wang *et al.* (2011) to identify bot-relevant domain names and IP addresses by analysing network traces.

The objective of this study is to develop a botnet detection framework that is able to identify and remove stealthy botnets from real-world network traffic. In addition, as the volume of network traffic grows rapidly, the detection system is required to process a huge amount of information efficiently. Hence, in this study a scalable and distributed botnet detection system has been proposed which can handle heavy network traffics. To achieve this objective big data analytics tool called 'storm' is used. Storm is distributed and fault tolerant tool capable of doing parallel computation over streaming real time data.

MATERIALS AND METHODS

The main objective of this study is to detect different types of bot infections such as IRC bot, HTTP bot, P2P bot and DNS bot by analyzing the network traffic behavior. The framework of the proposed system is shown in Fig. 1. First network traffic is captured using wireshark tool then it is preprocessed to remove irrelevant packets. The filtered packets are passed to the feature extraction stage. The feature extraction is the process of extracting the fields from the packet which are necessary to detect bot behavior. Using the features extracted, the detection method analyzes and finds whether the packet is infected or not.

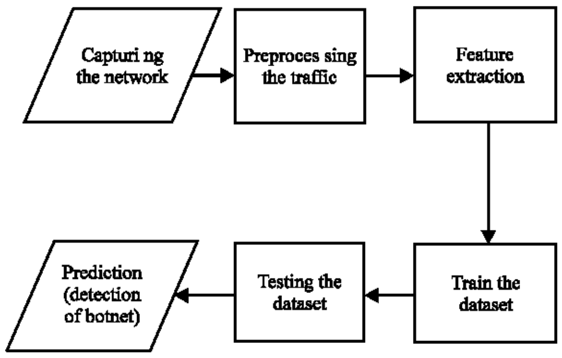

Fig. 1: Framework for botnet detection system

Dataset: Three different botnet dataset each consists of around 15,000 instances are collected. The normal dataset is captured in real time using wireshark tool.

Preprocessing: Preprocessing is used to improve the efficiency and ease of the detection process. It consists of the following:

- Data cleaning: detecting and removing errors, filling missing values and inconsistencies from data in order to improve the quality of data
- Data integration: malicious and non-malicious datasets are integrated
- Normalization: IP addresses are converted to unique numerical values. For example, the normalization function converts the IP address 147.32.1.101 as 100 and this value will be assigned for every occurrences of this IP address
- Duplicate elimination: editcap program is used to remove duplicate packets exists in the traffic. This program compares the current entry with the previous 'n' packets in the log ('n' specifies the window size) and eliminates this current entry if it exists in the 'n' packets

Feature extraction: In a large set of features, the subset of features necessary for detection of bots is identified

Table 1: Feature subset

Features for classification	Features for clustering
Source address	Source address
Source port number	Source port number
Destination address	Destination address
Destination port number	Destination port number
Protocol	Protocol
Average payload length for time interval	Duration
Variance of payload packet length for time interval	The number of packets transferred in forward directions
Number of packets exchanged for time interval	The number of packets transferred in backward directions
Number of packets exchanged per second in time interval T	
The size of the first packet in the flow	

based on the literature survey (AsSadhan and Moura, 2014; Chen and Lin, 2015; Choi and Lee, 2012). If all these unnecessary features are considered then the detection rate will get decreased. Table 1 shows the features necessary for detecting botnets.

First four features given in the Table 1 are used to find IP and port addresses of the communication. Once the detection system finds the malware hosts that target the victim host then these addresses are blacklisted in the victim which means it blocks the packets from those addresses. Fifth feature is a protocol which is used to find communication protocol (TCP, UDP, HTTP) in botnet. Moreover, botmaster does not change a carrier protocol during the lifetime of a botnet. The remaining temporal features are important in botnet detection due to two empirical observations of botnets behavior:

- The response time of bots is usually immediate and accurate once they receive commands from botmaster while normal human behavior might perform an action with various possibilities after a reasonable thinking time and
- bots basically have pre-programmed activities based on botmaster's commands and thus all bots might be synchronized with each other

First five features listed in the Table 1 are necessary basic features for both classification and clustering algorithms. Classification algorithm requires more temporal features compared to clustering because detection accuracy will be high in classification.

Training and testing of dataset: Random Forest classification and Fuzzy C-Means Clustering algorithms are used for training the dataset. For testing a tool called 'storm' is used. Apache 'Storm' is a free and open source distributed real time computation system which can be used with any programming language. Some of the use cases of storm are real-time analytics, distributed RPC, online machine learning, ETL, continuous computation and more. Storm executes very fast though a million tuples can be processed per second per node. It is fast, horizontally scalable, fault-tolerant, easy to operate and provides guarantee for the data which needs to be processed.

System implementation: Most of the botnet detection is based on payload analysis methods which inspect the contents of TCP and UDP packets for malicious signatures (Houmansadr and Borisov, 2013; Garcia *et al*, 2014; Rodriguez-Gomez *et al.*, 2014). Payload inspection involves very high identification accuracy when compared to other approaches but it violates the privacy. Furthermore, new bots utilize encryption and other methods to hide their communication and defeat packet inspection techniques.

Hence, traffic analysis exploits the idea that bots within a botnet typically have uniformity of traffic behavior then these behaviors may be analyzed and clustered using a set of features which differentiate malicious and non-malicious traffic. Traffic analysis does not depend on the content of the packets and is therefore unaffected by encryption.

Using the random forest and Fuzzy C-Means algorithms the dataset containing both known and unknown malwares is trained. The PMML file generated for these trained dataset is given as input file to storm. The spout program contains the stream of input to be processed after which the bolt performs the task of classification and clustering. The workers execute the task for the bolt which is scheduled by the zoo keeper. Many workers may share the same task by which parallelism is achieved. In the testing phase, new PMML file of the network traffic is given for predicting the dataset. Finally the storm predicts the presence of botnet in the network flow.

Random Forest (RF): RF is a classification and regression method (Singh *et al.*, 2014) based on the aggregation of large number of decision trees. Specifically, it is an ensemble of trees constructed from a training data set and internally validated to yield a prediction of the response given the predictors for future observations. Figure 2 shows Random Forest algorithm. There are several variants of RF which are characterized by:

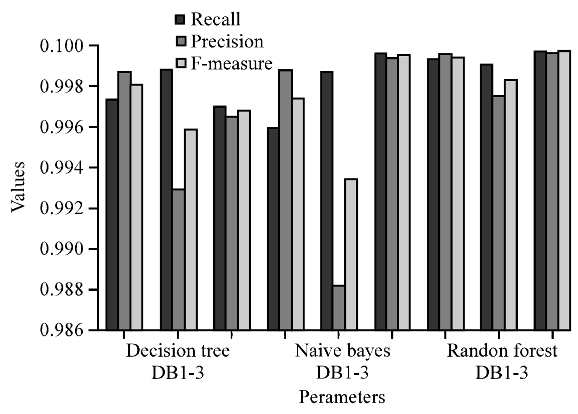

Fig. 2: Performance of classification algorithms

- The way each individual tree is constructed
- The procedure used to generate the modified data sets on which each individual tree is constructed
- The way the predictions of each individual tree are aggregated to produce a unique consensus prediction

Random Forest algorithm
Input; Dataset D:

- From the entire set of dataset 'D' a random subset 'S' is taken as training set
- Clusters the subset 'S' into groups and subgroups
- At each split or node in these groups, the attributes are chosen by calculating the Gini index values choose the best split which has the smallest Gini index value
- Step 3 is repeated for every split node for all groups
- The program makes multiple trees (forest). Each tree is different because for each split in a tree, variables are chosen at random
- Then the rest of the dataset (not the training set) is used to predict which tree in the forests makes the best classification of the data
- The tree T with the most predictive power is shown as output by the algorithm.

Output: Tree T

Fuzzy C-Means: The Fuzzy C-Means (FCM) algorithm is a clustering algorithm where each item may belong to more than one group where the degree of membership for each item is given by a probability distribution over the clusters. Let, $X = \{x_1, x_2, x_3, ..., x_n\}$ be the set of data points and $V = \{v_1, v_2,...\}$ be the set of centers. The main objective of Fuzzy C-Means algorithm is to minimize the objective function given in the Eq. 1.

$$J(U,V)^n = \sum_{i=1}^{c}\sum_{j=1}(\mu_{ij})^m \|x_i - v_j\|^2 \qquad (1)$$

where, '$||x_i\text{-}v_j||$' is the euclidean distance between jth data and jth cluster center. There shows Fuzzy C-Means algorithm. It is useful when the required numbers of clusters are pre-determined. The algorithm tries to put each of the data points into one of the clusters. FCM is different because it does not decide the absolute membership of a data point to a given cluster, instead, it calculates the likelihood (i.e., the degree of membership) that a data point will belong to that cluster.

Hence, depending on the accuracy of the clustering that is required in practice, appropriate tolerance measures can be put in place. Since, the absolute membership is not calculated, FCM can be extremely fast because the number of iterations required to achieve a specific clustering exercise corresponds to the required accuracy.

```
Algorithm A: Fuzzy C-Means algorithm
Description of the variables used:
    'n' is the number of data point
    '$v_j$' represents the jth cluster center
    'm' is the fuzziness index $m \in [1, \infty]$. If m value is closer to 1 then it gives good fuzziness. Hence, 2 is assigned for m.
    'c' represents the number of cluster center
    '$\mu_{ij}$' represents the membership of ith data to jth cluster center
    '$d_{ij}$' represents the Euclidean distance between ith data and jth cluster center
    'k' is the iteration step
    'J' is the objective function
1: Randomly select 'c' cluster centers
2: Calculate the fuzzy membership '$\mu_{ij}$' using:
        $\mu_{ij} = 1 / \sum_{k=1} (d_{ij} / d_{ik})^{(2/m-1)} c$
3: Compute the fuzzy centers '$v_j$' using:
        $v_j = (\sum_{i=1} (\mu_{ij})^m x_i)/(\sum_{i=1} (\mu_{ij})^m)$, for all $j = 1, 2, ..., c$
4: Repeat step 2 and 3 until the minimum 'J' value is achieved
Output: Clusters: Malicious and Non-Malicious
```

RESULTS AND DISCUSSION

Three different classification algorithms namely random forest, decision tree and naive bayes are compared with each other. Metrics considered for evaluating the performance of algorithms are recall, precision and F-measure. These three metrics are calculated from the confusion matrix obtained for the datasets by the RF classification and FCM clustering algorithms. Table 2 shows the representation of confusion matrix.

From the confusion matrix, the performance metrics such as recall, precision and F-measure are calculated using the Eq. 2-4.

Table 2: Confusion matrix

Class		Detected: Malicious	Detected: Non-malicious
Actual	Malicious	True positive	False negative
	Non-malicious	False positive	True negative

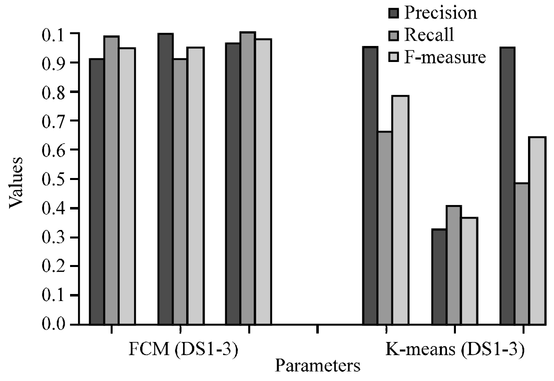

Fig. 3: Performance of clustering algorithms

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive+False negative}} \tag{2}$$

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive+False positive}} \tag{3}$$

$$\text{F-measure} = 2\times\frac{\text{Precision}\times\text{recall}}{\text{Precision+Recall}} \tag{4}$$

It was found that true positive rate achieved for random forest is higher than other two classifiers. Figure 2 shows precision, Recall and F-measure values for dataset 1-3. The result shows that naïve bayes algorithm has lowest values whereas random forest has the highest value. By comparing all dataset results random forest shows high true positive rate with an average of 0.96. For huge dataset random forest can be preferred, so that accuracy can be increased with low execution time.

The performance of Fuzzy C-Means algorithm is compared with K-Means for dataset 1-3. Figure 3 shows the precision, recall and F-measure values. Recall rate is high for FCM which means true positive rate is higher compared to K-means clustering.

Finally performance of Fuzzy C-Means algorithm is compared with Random Forest algorithm. The results are tabulated in Table 3.

Figure 4 shows the precision, recall and F-measure values for dataset 1. The performance of FCM is lower compared to Random Forest algorithm. Random Forest algorithm provides high accuracy for huge data set compared to fuzzy c-means because it does classification by analyzing more temporal features extracted from the packet.

Table 3: Confusion matrix for FCM and random forest FCM and random forest

	FCM		Random forest	
Class	Malicious	Non-malicious	Malicious	Non-malicious
Malicious	15109	1126	16203	32
Non-malicious	702	7101	15	7788

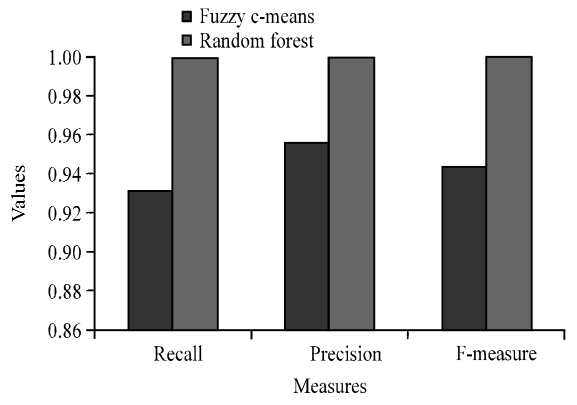

Fig. 4: Perform4ance of FCM and RF

CONCLUSION

Botnet detection is a challenging task, since the creators of botnets continue to adopt innovative means in creating botnets. In this study, RF classification and FCM Clustering algorithms are implemented using storm tool to detect botnets. The experimental results show that the percentage of accuracy obtained in RF is 6.25% higher than FCM. The FCM clustering is applicable for real time detection because features used for detection are directly taken from the network traffic whereas it takes some time to extract features for random forest classification.

REFERENCES

AsSadhan, B. and J.M. Moura, 2014. An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic. J. Adv. Res., 5: 435-448.

Chen, C.M. and H.C. Lin, 2015. Detecting botnet by anomalous traffic. J. Inform. Security Applic., 21: 42-51.

Choi, H. and H. Lee, 2012. Identifying botnets by capturing group activities in DNS traffic. Comput. Networks, 56: 20-33.

Dietrich, C.J., C. Rossow and N. Pohlmann, 2013. CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis. Comput. Networks, 57: 475-486.

Garcia, S., M. Grill, J. Stiborek and A. Zunino, 2014. An empirical comparison of botnet detection methods. Comput. Security, 45: 100-123.

Houmansadr, A. and N. Borisov, 2013. BotMosaic: Collaborative network watermark for the detection of IRC-based botnets. J. Syst. Software, 86: 707-715.

Lu, W., G. Rammidi and A.A. Ghorbani, 2011. Clustering botnet communication traffic based on *n*-gram feature selection. Comput. Commun., 34: 502-514.

Meng, Y. and L.F. Kwok, 2014. Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection. J. Network Comput. Applic., 39: 83-92.

Rodriguez-Gomez, R.A., G. Macia-Fernandez, P. Garcia-Teodoro, M. Steiner and D. Balzarotti, 2014. Resource monitoring for the detection of parasite P2P botnets. Comput. Networks, 70: 302-311.

Silva, S.S.C., R.M.P. Silva, R.C.G. Pinto and R.M. Salles, 2013. Botnets: A survey. Comput. Networks, 57: 378-403.

Singh, K., S.C. Guntuku, A. Thakur and C. Hota, 2014. Big data analytics framework for peer-to-peer botnet detection using random forests. Inform. Sci., 278: 488-497.

Wang, K., C.Y. Huang, S.J. Lin and Y.D. Lin, 2011. A fuzzy pattern-based filtering algorithm for botnet detection. Comput. Networks, 55: 3275-3286.

Zhao, D., I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani and D. Garant, 2013. Botnet detection based on traffic behavior analysis and flow intervals. Comput. Security, 39: 2-16.