General framework for detection of botnet using Random forest in real time

Nandhini Selvam , Dr. V.Vanitha , Ms. V.P.Sumathi

Abstract— Botnet is a distributed malware which spreads widely without the knowledge of the end user. The effect of botnet increases very fast today. Researches on detection of botnet in real time are less. It is necessary to detect the presence of botnet instantly to avoid the affects of botnet. Storm is a real time, distributed fault tolerant system and supports online machine learning techniques. Random Forest classifier can be used to produce higher accuracy rate for massive data. This paper presents a new framework for detecting the presence of botnet in real time using the Storm tool. The framework consists of three main components. First, preprocessing the data and extracting features for classification. Second, Training and testing the dataset in the classification algorithm and finally, predicting the presence of botnet in Storm. Experimental results show that the presence of botnet can be predicted with higher accuracy rate.

Index Terms— Botnet, Storm, C&C, Random Forest, DNS, HTTP, P2P

1 INTRODUCTION

Enormous number of malwares rises day to day. Botnet is such a raising malware which have various impacts. Botnet is derived from the word "Robot – Network". It is a distributed network where a leader (Bot master) takes control of the entire network. Computers which are under the control of bot master are known as bot which request and reply messages to the bot master through command and control (C&C) server without knowing to the end user. Initially a centralized server was used for sending the command and request later many detection methods were implemented to identify the centralized server which leads to track the bot master. Here five states occur in the life cycle of botnet. First state is the Injection state where the malware gets into the host system. This may be achieved by downloading the malwares through e-mail, trojan software, click fraud techniques. The second state is the connection state, here the bot will be get in connect with the C&C server. The third state is the waiting state where the bot will be waiting for the request from its master. The next stage is the execution state, here the received request will be performed by the bot. Finally, the maintenance and upgrading state, here the bot master upgrades their attacking techniques in order to ignore the detection method.

Peer to Peer (P2P) architecture had overcome this problem but later many researchers have also found detection techniques for this architecture too. To enhance the architecture hybrid methodology was used by the bot master, which combines the centralized and P2P architecture. Till now only few approaches have been proposed for detecting the botnet in real time.

Many machine learning techniques have been proposed for detecting the specific types of botnets and also some general framework have been proposed for detecting all types of botnets. Affects of botnet leads to financial loses, steals secured information and reduces national security. It also reduces the computation power of the CPU and system memory. The percentage of the victim's rate occurs majorly in US and UK. Whereas the occurrence of bot attackers are massively present in countries like India, Brazil, Russia and Germany. Recently Arbor networks have reported that 33% of bot server was located in France.

In this proposed architecture a general framework for detecting the botnet has been implemented to predict the presence of botnet in real time. This paper consists of the following sections. Section II describes the history of botnet. Section III gives the description about detection of botnet systems. Section IV consolidates and describes the related work of the botnet detection systems. Section V briefly explains the proposed architecture of this paper. Section VI shows the experimental results for the proposed architecture. Finally, Section VII summarizes the remarkable points which can be carrying forward further.

2 HISTORY OF BOTNET

The table 1 describes the raise of botnet and companies under taken down the botnets for past few years. The first bot is the Eggdrop which is a centralized botnet raised in the year 1993. The percentage of bots occurrences increases every year and spreads widely throughout the world. Though the detection technique get enhances, the way of spreading the bot and the way of attacking the victims get change. Though the characteristic for botnet varies with each other, there are some common behaviors for the botnet. Like botnet connections occur periodically where as normal connections are aperiodic. Also packet length for botnet request and reply messages are constant where as it will change incase for normal instances.

TABLE 1

- *Nandhini S is currently pursuing masters degree program in computer scienceand engineering in Kumaraguru College of technology,India. E-mail: nandhini542@gmail.com*
- *Dr. V.Vanitha is working as a professor in computer science and engineering in Kumaraguru College of technology, India. E-mail: vanitha.v.cse@kct.ac.in*
- *Ms.V.P.Sumathi is working as an asstitant professor in computer science and engineering in Kumaraguru College of technology, India. E-mail: sumathi.vp.cse@kct.ac.in*

History Of Botnet

Year	Botnet name	Comment	Size	Company Taken down
2009	Waledac	1.5 Billion spam mail/day	80,000	Microsoft
2008	Srizbi	One of the world's largest botnet	40% of all the spam	McColo
2011	TDL-4	P2P	4.5 millions	Kaspersky
2012	Grum	18000 Spam mail/day World's 3rd largest	560,000–840,000	Fireeye
2012	Carna	Internet census of 2012	420,000	-
2013	Zeus	Bank malware $70 million	163,812	Microsoft
June 2013	Citadel	Steal more than half a billion dollars	More than 5 million	Microsoft
February 2013	Bamital	-	More than 8 million computers	Microsoft
2013	Zeroaccess	$2.7 million/month Bit coin miner	2 million victims	Microsoft

3 Botnet Detection

Botnet detection technique can broadly classify into three main categories. They are 1) Bot detection 2) C&C detection 3) Botmaster detection

3.1 Bot detection

The detection technique will spot the bot system in the network. Here the attacker cannot be tracked. Hence, the attack cannot be stopped entirely but the system under the control of the leader can be dismantled.

3.2 C&C detection

In this type of detection technique, the centralized server will be hacked which leads to track the bot master. Many researches were done in detecting the centralized server. By hacking this server the botnet can be entirely dismantled.

3.3 Bot master detection

This kind of detection technique is an extreme end of spotting the botnet. Though this method is robust some researchers have done in tracking the bot master.

Botnet detection system can also be classified into two based on the network traffic anomalies (Silva et al. (2013)). They are host – based and network – based.

In the host – based approach, the machine behavior will be analyzed. To find whether the end system was attacked by bot can be found using this technique. In the network – based approach, the network traffic will be analyzed. Some botnets are specifically created for some protocols and architectures. Such bots can be detected by analyzing the network traffic.

The proposed architecture deals with the network – based approach and as a result the presence bot in the network flow will be predicted.

4 Related Work

Zhao et al. (2013) [2] proposed machine learning technique for detecting botnet using decision tree classifier and have achieved 90% of detection rate and 5% of false positive. Initially network flows were captured and then multiple time windows were chosen for the flow. Then twelve attributes were used to classify malicious and non malicious flows. In their approach, they have implemented two phases. First is the training phase, here they trained the algorithm using honeypot dataset. Next is the detection phase, here the presence of botnet can be predicted using decision tree. As the result the authors have proved that it is possible to detect the unknown bot using their technique.

Lu et al. (2011) [3] proposed a clustering algorithm for detecting the botnet. The author had implemented a payload analysis technique for detecting the botnet. This approach deals with the bit strings characteristics of the payload packet. A clustering framework was designed to detect botnet which consists of three main components. First in feature analysis, the response time of bots and also the behavior of bot master commands were analyzed though it will be automated. Finally two clusters were formed in which one contains the malicious data and the other has normal instances. A high detection rate with low false positive rate was achieved.

Dietrich et al. (2013) [4] proposed the machine learning approaches in which initially the known malware were traced. Three features were chosen for analyzing the network traffic and then C&C protocol, packet length and then number of distinct byte. By calculating the centroids for the known malwares the unknown malwares were found and then bots were detected.

A new framework is proposed by Choi et al. (2012) [5] for detecting DNS botnet namely BotGAD (Botnet Group Activity Detector). The BotGAD framework consist of five main components namely data collector, data mapper, correlated domain extractor, matrix generator and similarity analyzer. By calculating the similarity score for the matrixes, the botnet were detected. The BotGAD can be used to detect the botnet in real time and also unknown bots can be detected here. But the botnet using DNS protocol were only be detected by this framework. In this paper P2P botnet were detected by analyzing the communicational behavior of the command and control request packets.

Zhang et al. (2014) [6], initially identifies the P2P clients in the network. Then statistical fingerprints of the P2P communications were analyzed for the detection purpose. A flow clustering method is used for detecting the P2P botnet. The evaluation results shows 100 % detection rate for P2P botnet.

5 System Design

The proposed architecture consists of three main components. They are 1) Preprocessing and extracting the features for the botnet 2) Training the dataset using Random forest algorithm and 3) Testing the dataset and predicting the botnet in Storm. Figure 1 describes the proposed architecture.

5.1 Features selection

Initially the known botnet traces were gathered for the experiment. Then the in the preprocessing steps missing values were replaced and duplicates were also eliminated. Then the dataset will be normalized for future process. Ten features were extracted for training the algorithms. The features were listed in the table 2 as they were chosen from previous studies.

The time interval was chosen as 180s (Zhao et al. (2013) [2]).

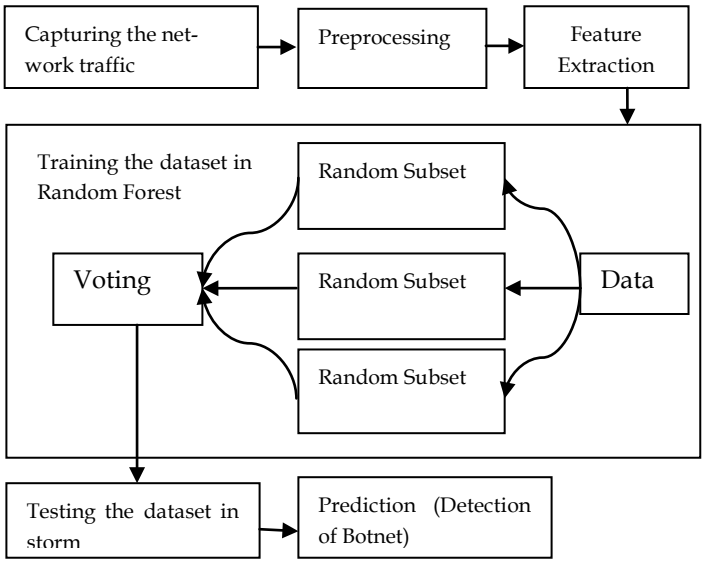

Figure 1. Proposed architecture

TABLE 2
Features for Classification

S.No.	Features
1	Source address
2	Source Port number
3	Destination address
4	Destination Port number
5	Protocol
6	Average payload length for time interval
7	Variance of payload packet length for time interval
8	Number of packets exchanged for time interval
9	Number of packets exchanged per second in time interval T
10	The size of the first packet in the flow

5.2 Classifier

The training dataset consist of the known and unknown malwares. Using the random forest algorithm the dataset was trained and PMML file was generated for the trained dataset which will be the input file for Storm. The spout program contains the stream of input to be processed and the bolt performs the task for classifier. The workers execute the task for the bolt which was scheduled by the zookeeper. Many workers may share the same take by which parallelism can be achieved. In the testing, new PMML file of the network traffic was given for predicting the dataset and finally the classifier predicts the presence of botnet in the network flow.

6 System Implementation

6.1 Apache Storm

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.

The three components of storm are nimbus, zookeeper and supervisor. Nimbus is the master node and supervisor is the worker node. Master node distributes the application code across various worker nodes, and assigns
tasks for different machines. It also monitors if any tasks get failures and then restarts them as and when required. Master node is stateless and stores all of its information in ZooKeeper and only single master node will be present in a Storm cluster. Supervisor nodes are will create, start, and stop the worker processes to execute the tasks assigned to that node in the storm cluster.

Storm topology consists of streams, spout, and bolt. A stream is an unbounded sequence of tuples that can be processed in parallel by Storm and can be processed by one or more number of bolts. A spout is the root for the tuples which read or listen data from an external origin. A bolt is the processing station which is responsible for executing the tuples.

6.2 Classification

Random forest classification algorithm is used for training the dataset. Random forest algorithm provides high accuracy rate for huge dataset. It is the collection of decision trees. 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.

7 Experimental Results

For testing, known malware dataset and live captures are integrated and then send as input for the storm spout. The percentage of accuracy obtained is 98% where the true positive rate is 0.96 and false negative rate is 0.04. The figure Fig.2 shows the tree generated by random forest classifier for a sample of 20 trees.

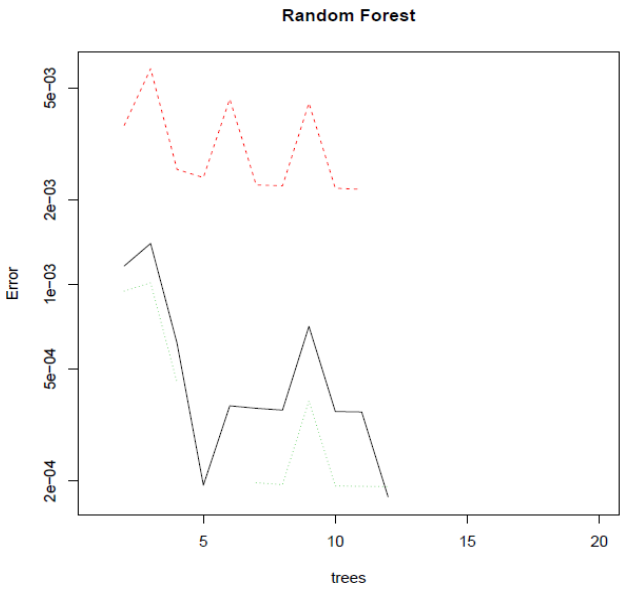

Figure 2. Tree generated by Random Forest

Table 3 shows the experimental results of the dataset. Here tree different classification algorithms are compared with each other. They are Random forest, Decision tree and naïve bayes. Here the true positive rate achieved for random forest is higher than other two classifiers.

TABLE 3
Experimental results

Dataset	Algorithm	TP	FP
Dataset 1	Decision Tree	0.995	0.01
	Naïve Bayes	0.991	0.017
	Random Forest	0.999	0.001
Dataset 2	Decision Tree	0.999	0.002
	Naïve Bayes	0.996	0.004
	Random Forest	0.998	0.002
Dataset 3	Decision Tree	0.997	0.016
	Naïve Bayes	0.999	0.002
	Random Forest	1	0

Figure 3 shows the comparison result of precision, Recall and F-measure values for dataset 1. In this naïve bayes algorithm has low values while comparing with others. Whereas random forest have the more recall value.

Figure 4 shows the comparison result of precision, Recall and F-measure values for dataset 2. In this naïve bayes and decision tree algorithms have low values while comparing with random forest.

Figure 5 shows the comparison result of precision, Recall and F-measure values for dataset 3. Here decision tress algorithm has low precision and recall value. By comparing all dataset results random forest shows high true positive rate with an average of 0.96.

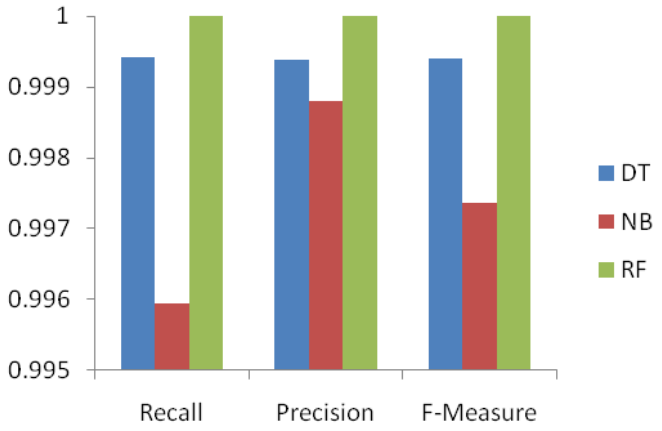

Figure 3. Comparison Precision, Recall and F-Measure score for Dataset 1

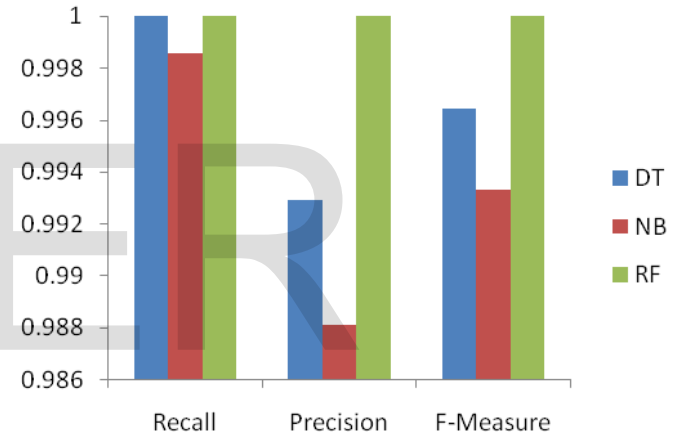

Figure 4. Comparison Precision, Recall and F-Measure score for Dataset 2

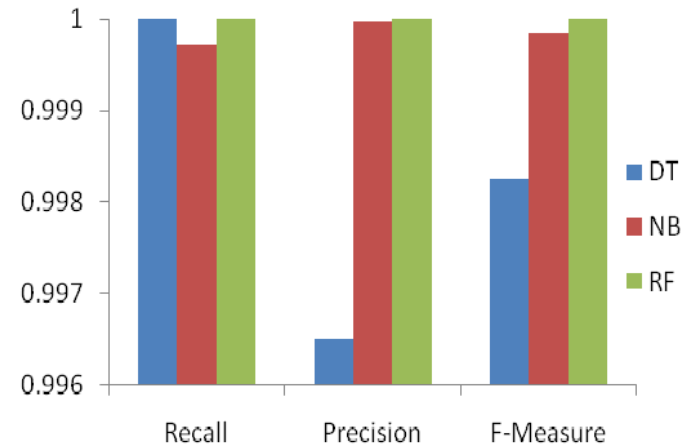

Figure 5. Comparison Precision, Recall and F-Measure score for Dataset 3

Figure 6 shows the comparision of estimated time for deci-

sion tree algorithm for storm and weka tool. Here the average estimated time of weka is 4.682s higher than estimated time of storm.

Likewise Figure 7 shows the comparision of estimated time for random forest algorithm for storm and weka tool. Here the average estimated time of weka is 3.652 s higher than estimated time of storm.

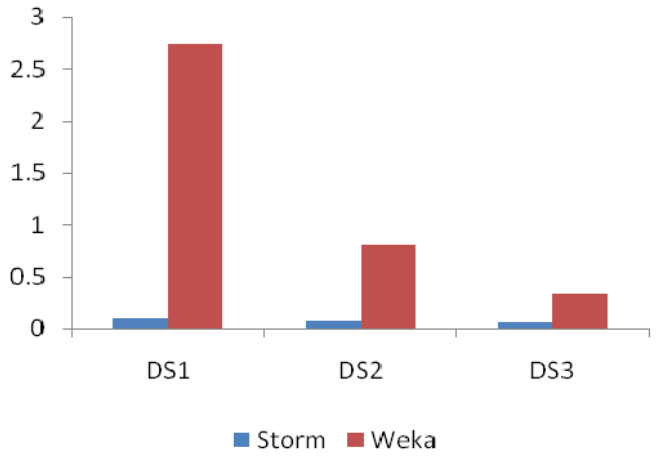

Figure 7. Comparison of estimated time for Random Forest algorithm

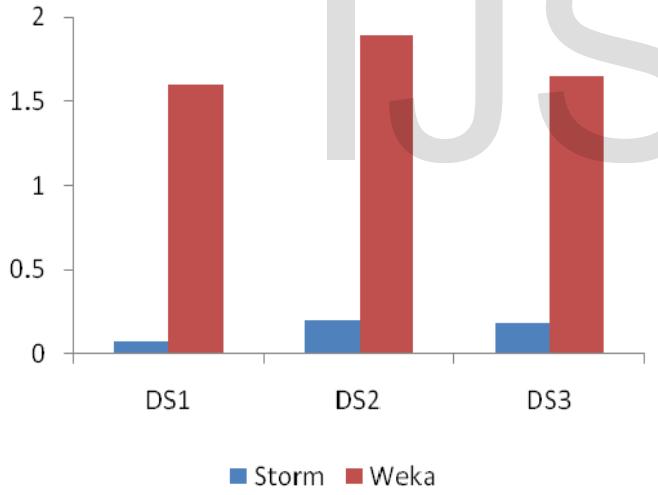

Figure 6. Comparison of estimated time for Decision tree algorithm

Thus the time can be reduced by using the storm tool while comparing with weka tool. For huge dataset random forest can be preferred so that accuracy can be increased with low execution time.

Table 4 shows the confusion matrix of dataset 1 for random forest algorithm. Here the wrongly predicted data are very less comparing with decision tree and naïve bayes algorithm.

TABLE 4

Experimental results

Confusion Matrix	Malicious	Non malicious
1	16203	32
2	15	7788

8. CONCLUSION

Apache Storm is used to build real-time data integration system and Random forest classifier is used to produces higher accuracy. From the proposed system, we conclude that the detection of botnet can be possible in real time using Storm tool so that the losses due to botnet can be avoided early. The experimental results show that the percentage of accuracy obtained is 98% where the true positive rate is 0.96 and false negative rate is 0.04.

REFERENCES

[1] Sérgio S.C. Silva, Rodrigo M.P. Silva, Raquel C.G. Pinto, Ronaldo M. Salles,"Botnets: A survey", *Computer Networks* 57, **2013,** pp **378–403**.

[2] David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali horbani and Dan Garant, "Botnet detection based on traffic behavior analysis and flow intervals", *computers & secrity* 39, **2013,** pp **2-16**

[3] Wei Lu, Goaletsa Rammidi and Ali A. Ghorbani, "Clustering botnet communication traffic based on n-gram feature selection", *Computer Communications* 34, **2011,** pp **502–514.**

[4] Christian J. Dietrich, Christian Rossow and Norbert Pohlmann, "CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis" *Computer Networks* 57, **2013,** pp **475–486.**

[5] Hyunsang Choi and Heejo Lee, " Identifying botnets by capturing group activities in DNS traffic", *Computer Networks* 56, **2012**, pp **20–33.**

[6] Yuxin Meng, Lam-ForKwok, "Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection", *Journal of Network and Computer Applications* 39, **2014**, pp. **83–92.**

[7] Kuochen Wang, Chun-Ying Huang, Shang-Jyh Lin, Ying-Dar Lin, "A fuzzy pattern-based filtering algorithm for botnet detection", *Computer Networks* 55, **2011,** pp. **3275–3286**.

[8] Chia-Mei Chen, Hsiao-Chung Lin, "Detecting botnet by anomalous traffic", *journal of information security and applications*, **2014** pp. **1–10.**

[9] Basil AsSadhan, Jose´ M.F. Moura, "An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic", *Journal of Advanced Research* 5, **2014**, pp. **435-448.**

[10] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, Chittaranjan Hota, "Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests", *Information Sciences* 278, **2014**, pp. **488–497**.

[11] Rafael A. Rodríguez-Gómez , Gabriel Maciá-Fernández, Pedro García- Teodoro, Moritz Steiner and Davide Balzarotti, "Resource monitoring for the detection of parasite P2P botnets", *Computer Networks* 70, **2014**, pp **302–311.**

[12] Amir Houmansadra and Nikita Borisovb, "BotMosaic Collaborative network watermark for the detection of IRC-based botnets", *The Journal of Systems and Software* 86**, 2013**, pp **707– 715**.

[13] Shui Yu, Senior Member, IEEE, Song Guo, Senior Member, IEEE, and Ivan Stojmenovic, Fellow, IEEE, "Fool Me If You Can: Mimicking Attacks and Anti-attacks in Cyberspace", *IEEE Transactions On Computers* , **2013.**

[14] Jingyu Hua and Kouichi Sakurai, "Botnet command and control based on Short Message Service and human mobility, *Computer Networks* 57, **2013**, pp **579–597.**

[15] S. García, M. Grill, J. Stiborek and A. Zunino, "An empirical comparison of botnet detection methods", *computers & security* 45 , **2014**, pp **100-123**.

[16] Junjie Zhang, Roberto Perdisci, Wenke Lee, Xiapu Luo, and Unum Sarfraz, "Building a Scalable System for Stealthy P2P-Botnet Detection", *IEEE transactions on information forensics and security*, vol. 9, no. 1, **2014.**

[17] Chun-Ying Huang, "Effective bot host detection based on network failure models", *Computer Networks* 57, **2013,** pp. **514–525**.

[18] Seungwon Shin, Zhaoyan Xu, Guofei Gu, "EFFORT: A new host–network cooperated framework for efficient and effective bot malware detection", *Computer Networks* 57, **2013**, pp. **2628–2642**.

[19] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu, "Design and analysis of a social botnet", *Computer Networks* 57, **2013**, pp. **556–578**.

[20] Weizhang Ruana, Ying Liub, Renliang Zhaob, "Pattern Discovery in DNS Query Traffic", *Procedia Computer Science* 17, **2013**, pp. **80 – 87**