Scientific Research Publishing

Journal of Computer and Communications, 2018, 6, 119-129
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Comparative Study of Machine Learning Algorithms and Their Ensembles for Botnet Detection

Songhui Ryu, Baijian Yang

Purdue University, West Lafayette, USA
Email: ryu26@purdue.edu, byang@purdue.edu

How to cite this paper: Ryu, S. and Yang, B. (2018) A Comparative Study of Machine Learning Algorithms and Their Ensembles for Botnet Detection. *Journal of Computer and Communications*, **6**, 119-129.
https://doi.org/10.4236/jcc.2018.65010

Received: April 13, 2018
Accepted: May 28, 2018
Published: May 31, 2018

Abstract

A Botnet is a network of compromised devices that are controlled by malicious "botmaster" in order to perform various tasks, such as executing DoS attack, sending SPAM and obtaining personal data etc. As botmasters generate network traffic while communicating with their bots, analyzing network traffic to detect Botnet traffic can be a promising feature of Intrusion Detection System. Although such system has been applying various machine learning techniques, comparison of machine algorithms including their ensembles on botnet detection has not been figured out. In this study, not only the three most popular classification machine learning algorithms—Naive Bayes, Decision tree, and Neural network are evaluated, but also the ensemble methods known to strengthen classifier are tested to see if they indeed provide enhanced predictions on Botnet detection. This evaluation is conducted with the CTU-13 public dataset, measuring the training time of each classifier and its F measure and *MCC* score.

Keywords

Machine Learning, Ensemble Method, Botnet, CTU-13

1. Introduction

As a network of compromised devices called bots, a botnet executes malicious tasks under the control of the attacker, a botmaster. A botnet has been a threat to cybersecurity [1] [2]. According to Ref. [3], the primary goals of botnets are as follows:

- **Information dispersion:** sending SPAM, executing Denial of Service (DoS) attack, distributing false information from illegal sources.

- **Information harvesting:** obtaining identity, password and financial data.
- **Information processing:** processing data to crack the password for access to additional hosts.

A botnet has been grown as a menace since the first botnet. EggDrop was reported in 1993 [1]. For example, [4] it was reported that the Necurs botnet was one of the most active distributors of malware in 2016. More than 2.3 million spam emails carrying JavaScript downloaders, VBS, and WSF attachments were sent out by Necurs in just one day on November 24, 2016.

The Mirai botnet, according to the same report [4], drove the largest DDoS attack ever recorded in 2016 on the French hosting company OVH peaking at 1Tbps mostly targeting IoT devices, such as home routers and IP cameras. As Gartner predicted that there would be more than 20 billion IoT devices by 2020 [5], it is important that botnets like Mirai should be addressed.

The speed of growth of botnet threat is also rapid. According to the report from Spamhaus [6], the number of IP addresses that was figured out as a Control and Command (C & C) server hosted by Amazon in 2017 increased 6 times against that of 2016.

A botnet usually has distinguishable architecture features [7]. A botmaster usually sets up a Control and Command server to easily communicate with his bots. For this type of centralized architecture, IRC protocol has been the most popular, but HTTP or POP3 have been used as well. The centralized botnets like Rxbot, Festi, and Bobax using IRC, HTTP, TCP can be depicted as **Figure 1**. On the other hand, a botnet can feature a Peer-to-peer architecture. In a P2P botnet, a bot can bypass connections to other bots in case there are firewalls in some of the connections. The decentralized botnets like TDL-4 utilizing P2P protocol can be depicted in **Figure 2**.

To detect botnet, approaches focusing on anomalies in bot(net)s' network behavior with or without temporal behavior have been proposed. Most of the previous studies adopted machine learning technologies along with heuristic rules [1] [2]. Even though the previously proposed detection systems utilized both supervised and unsupervised machine learning algorithms, the ensemble methods have not been discussed yet.

In this paper, focusing on the ensemble methods, supervised machine learning algorithms popularly used in previous studies were evaluated with their ensembles. As the ensemble methods were designed to strengthen weak classifiers, it would be meaningful to figure it out if the ensemble methods are indeed beneficial when it comes to botnet detection.

In the following chapter, popular classification algorithms that have been used in serval botnet detection proposals and the ensemble technology are explained.

2. Related Works

2.1. Machine Learning for Botnet Detection

In the previous studies where they used supervised machine learning algorithms,

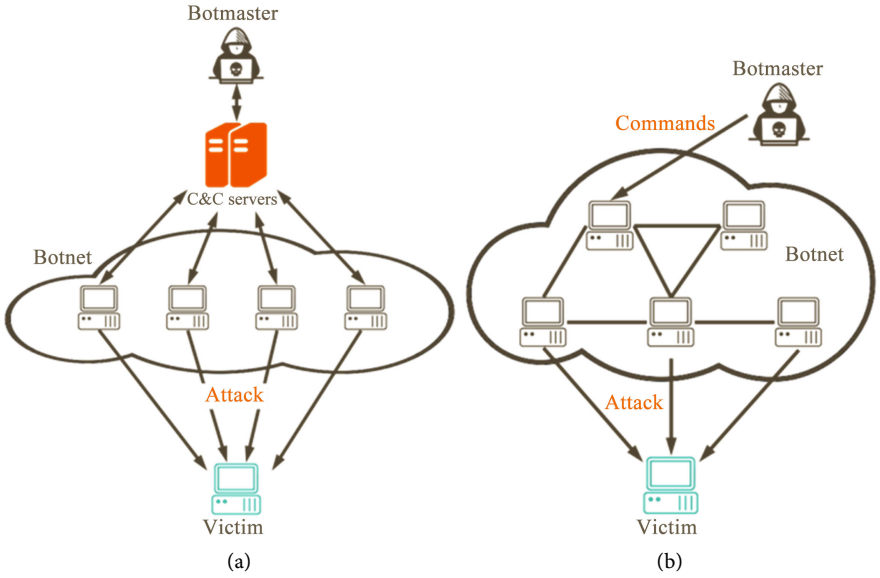

Figure 1. Examples of botnet architecture. (a) Centralized C & C architecture; (b) Decentralized P2P C & C architecture.

Id	Duration(hrs)	# Packets	# NetFlows	Size	Bot	# Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.26	50,888,256	1,925,150	34GB	Virut	1

Figure 2. Amount of data on each botnet scenario [17].

three algorithms—Naive Bayes, decision tree, and (artificial) neural networks were popularly adopted [8] [9] [10] [11].

2.1.1. Naive Bayes

Naive Bayes algorithm is a simple and intuitive classification technique based on the Bayes theorem assuming each feature contributes independently to the probability of an event [12]. Specifically in machine learning, Naive Bayes classifier calculates all the probability for all classes for a target feature and selects one with the highest probability. Furthermore, Gaussian Naive Bayes (GNB) assumes that the values associated with each class of each feature follow a Gaussian distribution. Although those assumptions do not happen often in real life, Naive Bayes shows relatively better results than other models like logistic regression. Also, it can generate models very quickly with very little computation overhead . It is therefore a popular choice for SPAM filters and other real-time anomaly detection algorithms [13].

2.1.2. Artificial Neural Networks

Neural networks, analogous to the human brain, refer to large connections of simple units called neurons. Consisting of three layers—input layer, hidden layer(s) and output layer, Neural network takes each record to pass its features onto input layer, and then the model makes decisions calculating weights of hidden neurons to get the single highest value at the output layer. A feed-forward neural network where the output of one layer is used as input to the next layer does iterate for the same data to compare the output to true value so that it adjusts the weights in the hidden neurons with its error term. Recurrent neural networks, however, adopts feedback loops between neurons that resemble human brains [14].

2.1.3. Decision Tree

As another popular classification method, decision tree generates a tree-like model of decisions based on decision rules inferred from the data. The goal is to create a model that predicts the value of a target variable based on several input variables. In classification decision tree, dependent variables can be categorical. Unlikely other machine learning algorithms, a decision tree is easy to interpret with tree visualized.

2.2. Ensemble Methods

Ensemble methods make a set of classifiers into an ensemble by combining the prediction from each classifier either with weight or not. It is regarded as one of the possibilities to improve the accuracy. Typically, there are three types of ensemble methods as introduced below [15].

- **Voting:** as the simplest way to form an ensemble, voting classifier consists of multiple models of diverse types. In the training step, all the models are trained separately with whole training data and it averages the posterior probabilities that are calculated by each model in the recognition step.
- **Bagging:** it, also called bootstrap aggregation, manipulates the training data to generate multiple models. Instead of training the model with the whole training data, bagging randomly samples the training set from the total training data to make sub-models.
- **Boosting:** it also samples out the training data like bagging does but maintains a set of weights on the data. Especially, AdaBoost where the weighted errors of each model update weights on the training data gives more weight on the data with lower accuracy and less weight on the data with higher.

Along with the concept of bagging, random forest is an ensemble of multiple decision trees. By randomly selecting features from the data, it generates decision trees and then for unseen data, the class that the majority of decision trees predict is selected as the prediction for the input. Random forest is known as a way of avoiding overfitting that can happen in a single decision tree [16].

Although ensemble methods are an effective way of reducing variances when it comes to prediction model, it is obvious that they come with more computa-

tion. Thus, a poor model can enhance its accuracy as the cost of the extra computation. For this reason, the ensemble methods are often used with a fast classifier such as decision tree as the case of random forest.

3. Methodology

3.1. Dataset

Finding an appropriate network traffic dataset for machine learning is often challenging. For supervised machine learning, the fact that the data should be properly labeled unless the target feature is already in the dataset makes the task onerous. Addressing this problem, Sebastian Garcia *et al.* created the CTU-13 dataset labeled as botnet, normal and background in the previous research [17]. Even though there had been several botnet datasets downloadable, such as, [18], [19], [20], [21], they were either not representative of the real-world traffic, or not labeled, or not suitable for every detection algorithms that the authors wanted to compare [17]. For those reasons, the CTU-13 dataset was generated with several fundamental design goals, to have real botnets attacks from several types of botnets, to aggregate the packet data to NetFlow flows because of the privacy issue, and to have the data labeled, etc. For more details and characteristics are as shown in **Figure 2** and **Figure 3**. In this comparative study, out of 13 different captures called scenario, the scenario 4, 10 and 11 featuring 1, 10, and 3 Rbot(s) respectively were used.

3.2. Data Features

When the researchers of the CTU-13 creating the dataset, they conducted preprocessing converting pcap files to NetFlow files. In that stage, they configured the data with those following features: start time, end time, duration, protocol, source IP address, source port, direction, destination IP address, destination port, flags, type of services, number of packets, number of bytes, number of flows, and label.

3.3. Metrics

To measure the accuracy of a classifier, taking account confusion matrix is the most common way. Precision meaning the percentage of correctly predicted event from the pool of total predicted event, and recall meaning the percentage of correctly predicted event from the pool of actual events respectively are important.

3.3.1. *F*1 Score

Taking both precision and recall into account, the *F*1 score gives a more balanced view compared to using only precision or recall. The *F*1 score can be between 0 and 1 where 1 means its best accuracy.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9.120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2,48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5.099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7,6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

Figure 3. Distribution of labels in the NetFlows for each scenario in the dataset [17].

3.3.2. Matthews Correlation Coefficient (*MCC*)

Unlike metrics above, *MCC* which is also known as the phi coefficient is considered to be less biased because it incorporates True Negative as well. According to [22], *MCC* is more robust to an imbalanced data where classification methods tend to biased toward the majority class than the *F*1 score or accuracy. The range of *MCC* lies between −1 to +1 where +1 means a perfect prediction, 0 no better than a random prediction and −1 and inverse prediction.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2)$$

To evaluate the classification algorithms along with ensemble methods for the CTU-13 dataset, Scikit-learn on a single core of Intel Xeon-E5 with 64GB of memory was used. Because some of the features were categorical which Scikit-learn cannot handle properly, data preparation including encoding and standardization were conducted.

4. Results and Discussion

The evaluation results are described in **Table 1** and **Table 2**. **Figure 4** also provides graphical views to compare the classifiers against each scenario based on their time consumption for training and *MCC* score.

For every data and algorithms, *F*1 scores are higher than *MCC* scores. This is because the *F*1 score does not consider the true negatives. For this reason, *MCC* is preferred for a binary classification. In the following discussion, accuracy refers to *MCC* score and S denotes scenario of the dataset.

Among individual algorithms, for S10 and S11, NN and DT show decent accuracies over 0.91 and 0.98, respectively. However, NN takes much longer time, about 3 - 4 times longer in S4 and S10. For S4, the accuracy score dramatically goes down. The only structural difference between those three datasets is the ratio of botnet traffic. Even though S4 is the largest dataset, it only has one Rbot with 0.15% of botnet traffic ratio, which means the data is highly imbalanced or skewed. On the other hand, S10 has 8.11% of botnet traffic and S11 has 7.6%.

This pattern appears the same on the result of voting. This is on the ground

that voting works by averaging out each outcome from the model. Boosting method does not significantly help either GNB or DT. The nature of boosting is turning weak models, which has slightly better prediction than random, into a strong one. In this regard, it obviously does not make DT strong as it already

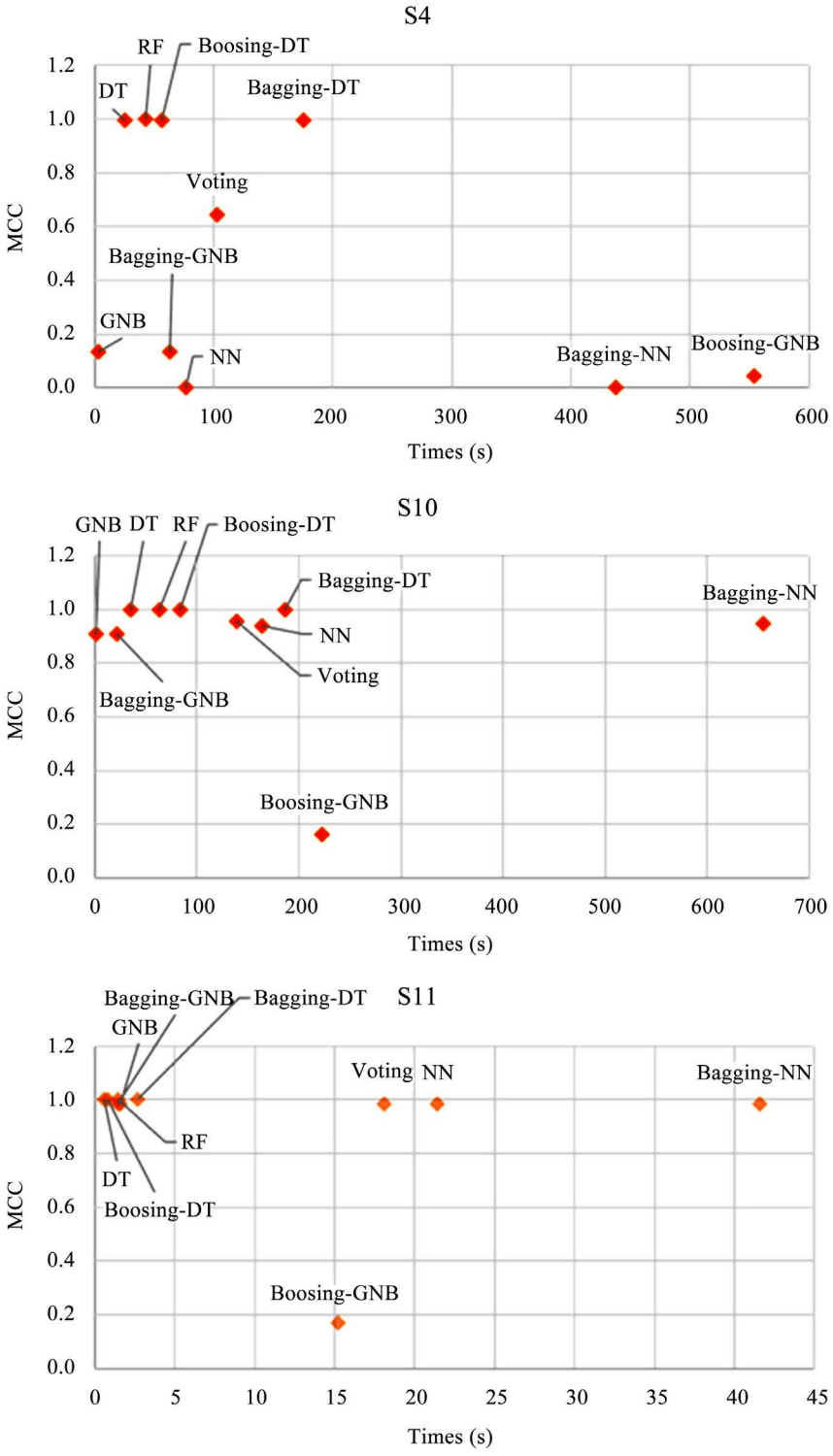

Figure 4. Time and *MCC* evaluation against each dataset.

Table 1. Time consumed for model training (sec).

Method	Scenario 4		Scenario 10		Scenario 11	
	*F*1	*MCC*	*F*1	*MCC*	*F*1	*MCC*
GNB	0.986159	0.135260	0.988762	0.910358	0.992302	0.982357
NN	0.998489	0.000000	0.992646	0.939776	0.993299	0.984639
DT	0.999990	0.996779	0.999982	0.999849	0.999971	0.999935
Voting	0.999117	0.644421	0.994763	0.956586	0.993841	0.985878
Boosing-GNB	0.967339	0.043543	0.867776	0.162963	0.378982	0.168781
Boosing-DT	0.999989	0.996285	0.999983	0.999857	0.999963	0.999916
Bagging-GNB	0.986170	0.135319	0.988758	0.910333	0.992253	0.982245
Bagging-NN	0.998489	0.000000	0.993613	0.946912	0.993670	0.985486
Bagging-DT	0.999991	0.996955	0.999981	0.999836	0.999955	0.999897
RF	0.999997	0.998930	0.999988	0.999896	0.999972	0.999935

Table 2. Evaluation result.

Method	Scenario 4	Scenario 10	Scenario 11
GNB	2.68	1.59	1.57
NN	76.24	163.86	21.44
DT	25.48	35.39	0.62
Voting	103.05	139.56	18.06
Boosing-GNB	554.14	222.48	15.2
Boosing-DT	56.77	83.23	0.77
Bagging-GNB	62.90	22.13	1.47
Bagging-NN	437.47	654.84	41.61
Bagging-DT	175.11	186.07	2.65
RF	43.17	63.74	1.44

had a good accuracy. The interesting thing comes with boosting-GNB. For S4, the *MCC* scores are near zero which means the prediction is no better than random. Also, it shows around 0.16 for S10 and S11, which are opposite results of using sole GNB. In the study by Ting and Zheng [23], the similar drop-down appeared in a specific dataset, Tic-Tac-Toe. They explained it is because Naive Bayes is very stable carrying a strong bias, in the boosting process the sub-classifiers may not be diverse enough [23]. But finding the exact reason of the drop-down is put to the future work at this moment.

Bagging each algorithm seems very similar to using a single classifier only for each dataset. While training a bagging model, multiple sub-datasets sampled out from the original dataset make their own classifier and then predictions from those classifiers are voted. This dataset, however, may not take benefit from sampling because the data is too imbalanced.

While the ensemble methods offered by Scikit-learn are not significantly beneficial on each algorithm, random forest appears highly effective in terms of both accuracy and training time. As a combination of decision trees, it performs implicit feature selection taking feature importance into consideration. Also making multiple sub-decision trees with part of features and data rows, it can run extremely faster than other methods and even can be easily parallelized. Considering parallelization is tough to be implemented in boosting and large neural networks, random forest seems like an excellence.

Compared to the previous research [17] where they measured the *F*1 score to several different botnet detection systems based on rule-based approaches and clustering methods, the *F*1 scores from this research is far above for all of the scenarios except S11. According to [16], they used all dataset and separated them into the training and test data in a way that the methods can generalize, detect new behaviors, and avoid the bias. Thus, the evaluation utilizing the three machine, learning algorithms and their ensembles offer better detection accuracy compared to the previous research.

5. Conclusion

In this study, three popular machine learning algorithms—Gaussian Naive Bayes, neural networks, decision tree were tested. Furthermore, the ensemble methods—voting, adaboosting, and bagging were also compared to figure out if ensemble methods would be significantly beneficial for botnet detection. Random forest which is a refined ensemble of decision tree was also tested. To detect botnet traffic out of all network traffic, decision tree without any ensemble method or random forest would be the most reliable approaches. It runs much faster than NN alone, with the better accuracy. Even though GNB runs the fastest, the accuracy varies on the dataset. Unlike the common expectation, adopting ensemble methods on machine learning algorithms for botnet detection in a hope of enhancing the accuracy is not preferable because it does not give remarkably more accurate result while consuming much more time. The question that this evaluation gives is that why the accuracy scores drop down when boosting is applied to GNB. Even though it was explained in [23], the reason why it makes the poor results rather than remain the same could be studied further.

Acknowledgements

This work was supported in part by a grant from Intel Grant #301620.

References

[1] Silva, S.S., Silva, R.M., Pinto, R.C. and Salles, R.M. (2013) Botnets: A Survey. *Computer Networks*, **57**, 378-403. https://doi.org/10.1016/j.comnet.2012.07.021

[2] Garcıa, S., Zunino, A. and Campo, M. (2014) Survey on Network-Based Botnet Detection Methods. *Security and Communication Networks*, **7**, 878-903. https://doi.org/10.1002/sec.800

[3] Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B. and Dagon, D. (2007) Peer-to-Peer Botnets: Overview and Case Study. *HotBot*, **7**, 1-1.

[4] Chandrasekar, K., Cleary, G., Cox, O., Lau, H., Nahorney, B., *et al.* (2017) Internet Security Threat Report. *Technical Report*, **22**, 1-77.

[5] van der Meulen, R.(2017) Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, up 31 Percent from 2016. www.gartner.com/newsroom/id/3598917

[6] Spamhaus (2018) Botnet Threat Report 2017. https://www.spamhaus.org/news/article/772/spamhaus-botnet-threat-report-2017

[7] Barford, P. and Yegneswaran, V. (2007) An inside Look at Botnets. In: Christodorescu, M., Jha, S., Maughan, D., Song, D. and Wang, C., Eds., *Malware Detection*, *Advances in Information Security*, Springer, Boston, 171-191.

[8] Livadas, C., Walsh, R., Lapsley, D. and Strayer, W.T. (2006) Using Machine Learning Techniques to Identify Botnet Traffic. *Proceedings* 2006 31*st IEEE Conference on Local Computer Networks*, Tampa, 14-16 November 2006, 967-974.

[9] Strayer, W.T., Lapsely, D., Walsh, R. and Livadas, C. (2008) Botnet Detection Based on Network Behavior. In: Lee, W., Wang, C. and Dagon, D., Eds., *Botnet Detection*: *Countering the Largest Security Threat*, Springer, Berlin, 1-24.

[10] Sangkatsanee, P., Wattanapongsakorn, N. and Charnsripinyo, C. (2011) Practical Real-Time Intrusion Detection Using Machine Learning Approaches. *Computer Communications*, **34**, 2227-2235. https://doi.org/10.1016/j.comcom.2011.07.001

[11] Lu, W., Rammidi, G. and Ghorbani, A.A. (2011) Clustering Botnet Communication Traffic Based on *N*-Gram Feature Selection. *Computer Communications*, **34**, 502-514. https://doi.org/10.1016/j.comcom.2010.04.007

[12] McCallum, A., Nigam, K., *et al.* (1998) A Comparison of Event Models for Naive Bayes Text Classification. *Learning for Text Categorization*: *Papers from the* 1998 *AAAI Workshop*, Madison, 26-27 July 1998, 41-48.

[13] Metsis, V., Androutsopoulos, I. and Paliouras, G. (2006) Spam Filtering with Naive Bayes—which Naive Bayes? *Third Conference on Email and Anti-Spam* (*CEAS*), Mountain View, July 27-28 2006, 28-69.

[14] Nielsen, M.A. Neural Networks and Deep Learning. Determination Press. http://neuralnetworksanddeeplearning.com

[15] Dietterich, T.G., *et al.* (2000) Ensemble Methods in Machine Learning. In: Josef, K. and Fabio, R., Eds., *Multiple Classifier Systems*. *MCS* 2000. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 1-15.

[16] Breiman, L. (2001) Random Forests. *Machine Learning*, **45**, 5-32. https://doi.org/10.1023/A:1010933404324

[17] Garcia, S., Grill, M., Stiborek, J. and Zunino, A. (2014) An Empirical Comparison of Botnet Detection Methods. *Computers & Security*, **45**, 100-123.

[18] Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A. (2012) Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. *Computers & Security*, **31**, 357-374.

[19] Dainotti, A., King, A., Papale, F., Pescape, A., *et al.* (2012) Analysis of a "/0" Stealth Scan from a Botnet. *Proceedings of the* 2012 *Internet Measurement Conferenc*e, Boston, 14-16 November 2012, 1-14. https://doi.org/10.1145/2398776.2398778

[20] Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W., Felix, J. and Hakimian, P. (2011) Detecting P2P Botnets through Network Behavior Analysis and Machine Learning. 2011 *Ninth Annual International Conference on Privacy*, *Security and Trust* (*PST*), Montreal, 19-21 July 2011, 174-180.

[21] Sony, C. and Cho, K. (2000) Traffic Data Repository at the Wide Project. *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, San Diego, 18-23 June 2000, 263-270.

[22] Boughorbel, S., Fethi, J. and Mohammed, E. (2017) Optimal Classifier for Imbalanced Data Using Matthews Correlation Coefficient Metric. *PLoS ONE*, **12**, e0177678. https://doi.org/10.1371/journal.pone.0177678

[23] Ting, K.M. and Zheng, Z.J. (2003) A Study of AdaBoost with Naive Bayesian Classifiers: Weakness and Improvement. *Computational Intelligence*, **19**, 186-200. https://doi.org/10.1111/1467-8640.00219