Automated Botnet Traffic Detection via Machine Learning

Fok Kar Wai, Zheng Lilei, Watt Kwong Wai, Su Le, Vrizlynn L. L. Thing
Cyber Security & Intelligence, Institute for Infocomm Research (I^2R)
{fokkw, zhengll, wattkw, lsu, vriz}@i2r.a-star.edu.sg

***Abstract*—Connected machines become more vulnerable to malware infections which potentially cause them to be controlled as part of a botnet for cybercrime activities. Prompt detection of infected machines is required for protecting local networks and infrastructure as well as reducing the impact of botnets. In this paper, we propose the use of machine learning techniques involving multi-layer perceptrons and decision trees on network traffic analysis for the detection of botnet traffic. We enhance components of an existing detection framework with these techniques to automate its processes and improve performance at the same time. Our experiments indicate that the modifications successfully improved the overall performance of botnet traffic detection in both supervised and semi-supervised manners.**

I. Introduction

The connectedness of devices and machines via the network has accelerated the process of malware infections and created the potential for detrimental malicious activities. Infected machine may become part of an existing army of machines known as a botnet, which is controlled by the malware's creator. Botnets are commonly used for cybercrimes such as exfiltration of confidential and valuable data and launching of network-related attacks against target systems. The continued threat of botnets can be observed in Kaspersky Lab's cyberthreat reports. In its latest report for Distributed-Denial-of-Service (DDoS) attacks by botnets for the first quarter of 2018 [1], there was an occurrence of a 297-hour long sustained DDoS attack, one of the longest in recent years.

The detection of anomalous botnet traffic is vital for securing machines in networks. Therefore, vast amounts of effort have been put into researching and developing solutions for network anomaly detection. Statisical-based approaches such as [9], [12] and [10] utilise methods like sequential probability ratio test and likelihood ratio tests to detect anomalies from aggregate traffic. Clustering-based solutions can be found in [6], [8], and [11]. [2] and [5] present surveys that detail several types of techniques for general network anomaly detection as well as botnet detection specifically.

The multitude of different techniques introduced above shows that there is a huge challenge in developing an all-rounded solution that is not only able to detect a wide range of anomalies but also continuously update itself to deal with the anomalies that quickly evolve over time. As most techniques require considerable efforts in manual design and repeated experimentation for validation on specific datasets, they may be rigid in adapting to emerging anomalies. In order to solve this problem, there needs to be more automated solutions.

Therefore, the aim of our work is to explore the effectiveness of automating network anomaly detection techniques that rely on the use of manual processes and human determined thresholds with machine learning. In general, finetuning a manually designed framework on newly emerging anomalies consumes more time as compared to a machine learning framework which can automatically retune itself by learning from more data. Concretely, we build upon the framework in [4] that requires significant manual effort in extracting discriminative features and working out a classification criterion. We adopt popular machine learning techniques such as multi-layer perceptrons and decision trees for automatic feature extraction and classification. The experiments demonstrate that the proposed framework can reduce the manual processes and improve the frameworks ability to detect botnet traffic.

In summary, following are the contributions of our work:

- We present a comparison of manual techniques against automated machine learning based methods for the purpose of detecting anomalous network traffic.
- We propose two methods to automate an existing manual framework that detects botnet traffic with machine learning techniques including decision trees and multi-layer perceptrons. The two methods utilise machine learning in a different manner which allows us to perform deep analysis of the effects of machine learning when compared to the manual processes.
- The proposed methods are evaluated on a dataset containing real-world normal traffic and anomalous traffic belonging to 11 different types of botnets. The experimental results show that introduction of machine learning into the manual framework improved its performance.

In the following section, the some related works are discussed. Next, in Section III and IV, we review the base framework and present our methods of introducing machine learning techniques. Experiments and evaluation performed on our methods are presented and discussed in Section V.

II. Related Works

A framework for evidence gathering [4] has been proposed recently, involving the detection of anomalous patterns from network traffic. The work utilises regression techniques which are applied on a set of features extracted from the network traffic. For classification, self-defined measures which include the use of thresholds were applied on the patterns to decide whether the associated traffic is anomalous or not. The work

was evaluated on a dataset containing traffic produced by malwares related to eleven different botnets.

Various types of machine learning techniques have been explored for classifying network traffic features. In [16], a comparison of several classification algorithms for detecting anomalous flow data was done. Simple IP address and port features were used for training the classifiers. [7] is another flow based detection approach, but focused primarily on botnet traffic classification. In particular, the authors evaluated different classification algorithms when different types of flow exporters were used. An ensemble method is experimented in [14] where a combination of different classifiers were explored for detecting simple types of anomalies. [3] details a flow based botnet traffic detection approach which only requires learning from the normal traffic data.

Recent works [13] and [15] have experimented with the application of deep learning techniques for network anomaly detection. In [13], flows are represented as a sequence of states based on several extracted features. A special type of recurrent neural network, Large Short Term Memory (LSTM) network, is used to learn the changes in states of the flows over time. Classification of botnet traffic by using convolutional neural networks for representation learning is carried out in [15]. Instead of extracting information by manually analysing the traffic, raw traffic data files are converted into images and directly fed to the neural networks which are able to self-learn features.

While the use of machine learning algorithms has been explored in several of the above related works, there is a lack of fair experimental comparison between manual efforts and the machine learning based solutions. In this work, we carry out such a comparison and discuss the effects of using machine learning in semi-supervised and supervised manners, respectively.

III. Manual Evidence Gathering for Botnet Traffic

The framework of manual evidence gathering [4] has shown its effectiveness in detecting eleven different botnets, with high detection rate and low false positive rate. The framework can be organised into the following three major stages:

1) Traffic Modelling and Feature Extraction: Network traffic flows are grouped together into different sessions. Each session represents a single sample for analysis and evaluation. Four basic features are extracted from each session, which are the Inter-arrival times of flows in the session, Sizes of flows in the session in Packets and in Bytes and the Degree of end-host in the session. Each feature is represented as an array of sequential values.
2) Pattern Detection: Various regression techniques are then applied on the features. Output parameters produced by these techniques are used to determine the strength of anomalous patterns for each feature.
3) Decision Making: A threshold-based meta-decision maker is defined to determine whether a session is anomalous based on the strength of anomalous patterns for all features.

The above framework requires significant manual effort in areas of solution design and classification outcome. For example, in the second stage, the detection of visual patterns indicates that a great amount of effort is required to carry out observations on the patterns that are produced by different regression techniques. The third stage involves a human-defined threshold-based decision maker that is empirically chosen based on the optimum results obtained from evaluation.

IV. Machine Learning for Anomaly Detection

In order to reduce the degree of manual efforts required in the above framework, we propose two methods to modify it with machine learning. Fig. 1 provides an overview of how the modifications are made. We adopt a progressive approach with regards to the level of machine learning utilised in each method. Such a design allows us to fully assess the effects of machine learning compared to human deterministic processes. We summarise the two methods with the following:

- (Method I) Enhancing threshold-based decision maker with machine learning algorithms: The second stage is enhanced via the extraction of output parameters from the regression analysis to produce engineered features. These features are used to train a machine learning model. The trained model is then able to automatically learn from test features and make a classification, removing the need to manually tune thresholds for decision making.
- (Method II) Applying neural networks directly on the basic features: The regression analysis module is further upgraded into a deep learning model. The basic features extracted from the first stage are directly fed into a neural network, thus training it to classify the features and produce scores. These scores are used as feature inputs to the classifiers employed in Method I for final decision making.

A. Method I: Enhancing threshold-based decision maker with machine learning algorithms

A human defined threshold-based decision maker may have some shortcomings. Manual finetuning of the thresholds may be required when detection of previously unseen anomalies is poor. This is not very practical and efficient in the long run. Thus, the aim of this modification is to replace the current decision maker with a machine learning model. The model can be continuously retrained with new data to keep it automatically up to date.

Training a machine learning model that is able to aid in the decision making requires an appropriate feature set. To this end, we create a feature set made up of underlying parameters obtained from the regression analysis in stage two of the original framework. Specifically, they are the output parameters from the linear and quadratic regression models fitted onto the data points of the basic features. These values contribute to the strength of anomalous patterns, hence influencing the decision making in the final stage. By learning from these

features, we expect the trained model to be able to act as the decision maker. Therefore, in this modification, the application of regression techniques on the basic features extracted in the first stage may be treated as a form of feature engineering.

Table I lists the 14 engineered features we use in our feature set. The first ten features in the table are the output parameters of linear regression applied on the basic features. The next two features involve quadratic regression. The remaining two features were derived using other measures indicated in the manual evidence gathering framework.

Next, using this feature set, we train a binary classifier to make a decision on whether an input traffic session is anomalous or not. We investigate the use of three widely adopted techniques which are linear Support Vector Machines (linear-SVM), Decision Tree (DT) and Random Forests (RF).

SVM is originally a linear classifier which attempts to build the most optimum linear hyperplane that is able to separate the samples from the two classes. If a hyperplane exists such that the class boundaries may be approximated well, then we can expect SVM to be a suitable solution for the problem. On the other hand, if the data is not linearly separable, we need to consider non-linear kernels in SVM or other non-linear classifiers.

Therefore, we also investigate the performance of non-linear classifiers such as DT and RF. After learning from the training data, the DT algorithm builds a hierarchical tree model. The model first consists of internal nodes where in each of them, a decision is made based on the value of a particular feature. The outcome leads to the selection of subsequent internal nodes down the tree until a terminal node is reached which indicates the class label. In terms of RF, a set of decision trees are individually trained on a different subset of the input samples and averaging is carried out to benefit from all the sub-classifiers. However, the powerful non-linear models like RF, may possibly overfit on the training data which leads to lower performance especially when classifying unseen test data.

As feature vector normalization is necessary in order to use SVM effectively, we perform Min-Max normalization on the feature vectors before giving them as input to the models. Experimental evaluation of the algorithms will be detailed in Section V.

B. Method II: Applying neural networks directly on the basic features

In the original framework, regression analysis is used to extract patterns from the basic features. Evaluation of the regression techniques and discovery of the potential patterns may be a tedious process. In addition, the patterns are limited to the regression techniques originally implemented and improvements are only possible with further manual effort. Here, we explore the use of Multi-layer Perceptron (MLP) to circumvent the above limitations in the following two-step approach:

1) In the first step, we utilise deep learning capabilities to automatically learn the discerning "patterns" for each feature instead of using regression analysis. Four MLP models are trained, each for one of the four basic features. Each MLP model will be able to produce a probability score indicating the existance of an anomalous pattern for that feature. Thus, the output of this step is a set of four probability scores for each input sample.
2) In addition to the four probability scores from the previous step, we include the score for the illegitimate TCP sequence feature computed in the original framework for a total of five scores. Recall that in the original framework, threshold-based decision is performed on the five scores. Similar to Method I, here we use the five scores as features to train machine learning models to learn to make the decision for us.

For the first step, we build four-layered MLP models for learning to classify the basic features. It uses the ReLU activation function in the two hidden layers. Fig. 2 provides an overview of the MLP model designed for the Flow Size in Packets feature. Before feeding the basic features of sequential values into the MLP layer, we need to align them. This is because for each basic feature, the feature dimension of the samples varies. Recall that each sample refers to a session. Since each session does not contain the same number of network flows, the number of values extracted for each feature

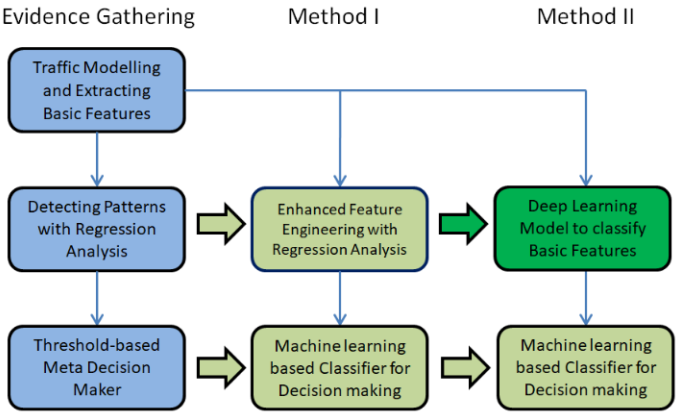

Fig. 1: Methods proposed to automate the detection of botnet traffic.

Basic features	**Engineered features**
Flow Size in Packets	Sum of Squared Errors Slope Coefficient of Determination
Flow Size in Bytes	Sum of Squared Errors Slope Coefficient of Determination
Degree of end-hosts	Sum of Squared Errors Slope Hypothesis Testing Result on Slope Coefficient of Determination, Linear Coefficient of Determination, Quadratic Difference between the Coefficients
Flow Inter-arrival times	Activity value
Illegitimate TCP flows	Count of flows

TABLE I: List of features.

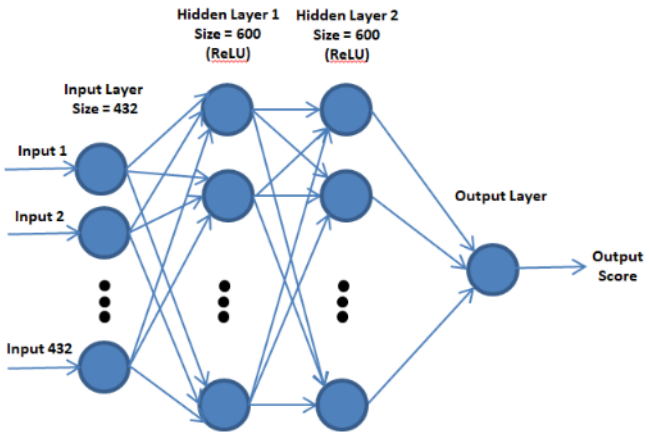

Fig. 2: Four-layered MLP neural network for the Flow Size in Packets feature.

is also different for each session. Therefore for each feature, we first compute the average dimension length of all the samples and select a value that is close to the average as the fixed dimension length. For example, for the Flow Size in Packets feature, the fixed dimension length chosen is 432. Hence in Fig. 2, the size of the input layer is 432. Next, for samples with feature dimensions smaller than fixed dimension length for that feature, we apply zero-padding to the head of the feature values; for cases where the feature dimensions are larger than the fixed dimension length, we perform the spline interpolation to down-sample the feature values. Finally, the trained MLP models are able to produce a score within the range [0, 1] that represents the probability of the aligned feature being anomalous or not. In the second step, we once again experiment with the use of the DT classifier.

V. Evaluation

A. Dataset

For the purpose of evaluating the performance of our machine learning approaches directly against the original framework in [4], we use the same dataset. The data consists of both normal and anomalous traffic, representing the two different classes. The normal traffic is combined from three different sources. The anomalous traffic is made up of malware traffic belonging to 11 different botnets which were obtained from the Stratosphere IPS Project. There are in total 1397 sessions for the anomalous dataset and 1006 sessions for the normal dataset.

B. Experimental Set-up

In this subsection, we illustrate the methods we use to divide the dataset into separate sets for training and testing of the machine learning models.

1) Leave-one-out Cross Validation: In this set-up, the dataset is split into different portions based on a specific criteria. The criteria used is samples belonging to the same botnet, 11 rounds of model training and testing are carried out. In each round, all the anomalous samples belonging to one botnet (out of the 11) are left out for testing while the samples from the remaining ten botnets are used in the training set. For the normal traffic samples, one-eleventh of them are randomly selected for testing in each round.

Using this set-up, we are able to better evaluate the generalisation capabilities of the models, i.e., ability of the models to detect samples from botnets which did not appear during training. This is extremely important in the real world where newer botnets and anomalies appear constantly. Systems need to be able to detect them without having prior information about the particular anomaly. Also the results produced by this method can be directly compared with that of the original framework.

2) Partial Leave-one-out Cross Validation: Similiar to the previous set-up, the anomalous data is divided up into train and test sets based on the botnet category resulting in another 11 rounds of experiments. However, there is a slight difference where in each round, instead of using all samples of the selected botnet in the test set, we use most of them. Thus, the train set consists of the remaining small number of samples from the selected botnet and the samples from the other ten botnets. The purpose of this is to further evaluate our models when they are able to learn from minimal amounts of data from previously unseen botnet. This is applicable to the situation where new botnets have just appeared and we are able to obtain some information which can be fed to the models for retraining.

C. Results

Before presenting the results, we explain the two statistics that we use to evaluate the classification results of our methods. They are the recall and the false positive rate (FPR) as defined in the following equations. The recall or the detection rate is the ratio of the number of detected botnet samples to the total number of anomalous samples in the same botnet category; the FPR is the probability of falsely classifying a normal sample as anomalous.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (1)$$

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} \quad (2)$$

For the 11 rounds of experiments on the different botnets, we also report the overall and average statistics. For example, the overall recall is calculated by finding the ratio of the total number of detected botnet samples to the total number of botnet samples in the dataset. The average recall is simply the mean of the 11 recall scores for each botnet group.

1) Results for Method I: In Table II, we present the Recall and FPR results for Method I in our work together with the existing manual evidence gathering work for comparison. There are three sets of results for Method I, each for a different machine learning algorithm employed in the method. These experiments were performed using the Leave-one-out Cross Validation set-up. In the table, both the overall and average

Botnet	# sessions	Manual evidence		Method I - SVM		Method I - DT		Method I - RF	
		Recall(%)	FPR(%)	Recall(%)	FPR(%)	Recall(%)	FPR(%)	Recall(%)	FPR(%)
Andromeda	148	89.2	-	17.4	10.4	81.6	0.9	51.1	0.5
Barys	16	100.0	-	100.0	10.7	100.0	1.1	99.6	0.8
Emotet	95	100.0	-	100.0	10.9	90.0	1.1	100.0	0.7
Geodo	63	69.8	-	80.7	11.1	85.7	1.0	81.1	0.8
Htbot	287	59.6	-	45.6	7.9	76.6	1.5	55.7	0.7
Miuref	82	53.7	-	64.6	10.7	63.7	1.2	66.9	0.7
Necurse	19	100.0	-	100.0	11.1	100.0	1.3	100.0	1.0
Sality	440	98.9	-	99.9	11.0	99.9	0.9	99.8	1.0
Vawtrak	40	100	-	100.0	11.7	100.0	1.1	100.0	0.9
Yakes	39	64.1	-	33.3	11.2	96.9	1.0	97.6	0.8
Zeus	168	79.2	-	63.1	9.8	91.5	0.7	81.6	0.6
Overall	-	82.6	7.9	70.8	10.6	**88.6**	**1.1**	80.6	0.8
Average	-	83.1	7.9	73.2	10.6	**89.6**	**1.1**	84.9	0.8

TABLE II: Results for manual evidence gathering framework and Method I using Leave-one-out Cross Validation.

results are provided as well as the individual results of each botnet category on its own.

The final results as presented in the evidence gathering work is an overall recall of 82.6% and a FPR of 7.9%. Thus, we also compute the average statistic for their results. By comparing the results in Table II, it can be observed that the introduction of the DT classifier in our method is able to improve the performance of the original work. While the use of SVM led to worse results, DT is able to increase the Recall by approximately 6% and simultaneously bring down the FPR from 7.9% to only 1.1%. When looking at DT results specifically, the individual Recall for six of the botnets are improved substantially. On the other hand, there is a performance decrease on only two botnet categories, Andromeda and Emotet. This indicates that in most cases, the model is able to learn discerning characteristics from the engineered features and at the same time have good generalisation capabilities to be able to detect unseen data. At certain times, the simple threshold based decision making performs better when the relatability between the training data and unseen data is not as high.

Next, based on the results of the different classifiers used in our method, we can conclude that the linear-SVM is not compatible with the nature of our data, i.e. underfitting of the training data. This is indicated by the substantial difference in performance between SVM and the non-linear classifiers. The close to zero FPR of DT and RF show that machine learning is extremely useful when a suitable type of classifier for the data is employed and the training data and validation data share similar characteristics. While RF produces the lowest FPR, DT is able to output a much higher Recall while recording only a slightly higher FPR. Thus, we can conclude that during training, the DT model suffers less from overfitting to the training data than the RF model. In summary, experiments indicate that DT is the overall best classifier on this data and is able to produce better performance than the original framework.

2) Results for Method II: Here we present the results for our second method, in which MLP was utilised to classify the basic features. Table III shows the results for both the leave-one-out and partial leave-one-out cross validation experiments. Recall that this method involves a two-step process. In the first step, MLP models are trained on basic features to produce scores for each of them. These scores are used in the following step as input features to a DT classifier. We have also tried other classifiers such as SVM and RF, but almost identical results are produced when different classifiers are used in the second step. Based on this we can conclude that the MLP models in the first step play a defining role in the entire process by producing discerning probability scores. This leaves a simple task for the classifiers in the next step. Thus for simplicity, we show only a single result of the DT classifier for each of the experiments in the table.

Botnet	Leave-one-out		Partial Leave-one-out	
	Recall(%)	FPR (%)	Recall(%)	FPR (%)
Andromeda	100.0	0.0	100.0	0.0
Barys	100.0	0.0	100.0	0.0
Emotet	100.0	0.0	100.0	0.0
Geodo	88.9	0.0	90.4	0.0
Htbot	28.6	0.0	62.8	0.0
Miuref	68.3	0.0	93.6	0.0
Necurse	89.5	0.0	93.3	0.0
Sality	78.0	0.0	84.4	0.0
Vawtrak	72.5	0.0	74.5	0.0
Yakes	100.0	0.0	100.0	0.0
Zeus	100.0	1.1	100.0	1.1
Overall	75.1	0.1	85.6	0.1
Average	84.2	0.1	90.8	0.1

TABLE III: Results for Method II using the two experimental set-ups.

Let us first observe the results for the leave-one-out cross validation experiment. A very distinct observation that can be made is that there are almost no false positives. In fact, there is only one occurrence in the whole experiment where false positives exist. This can be attributed to the similarity of the normal data in the train and validation sets. On the other hand, we do not obtain a significant improvement for the Recall

result. The overall Recall is lower as compared to the original framework, but is better when the average statistic is used. This method also has mixed results in terms of being able to improve the detection for certain botnets and at the same time worsen it for the others. This is highlighted by the result for Htbot where the Recall is as low as 28.6%. This could imply that it may be difficult for the neural network to learn the differentiating characteristics of the two classes using only the very basic features. However, we do note that the size of the dataset is only in the range of thousands of samples. This may be too small to bring out the full potential of neural networks for generalisation.

For further evaluation, we also carry out the second type of experiment (partial leave-one-out) with the results as shown in Table III. For each round in this experiment, we randomly choose five samples from the botnet category selected for testing. Thus, the test set comprises of all samples belonging to the selected botnet except for the previously chosen five. These five samples are mixed with the samples from the other botnets to form the training set. To produce more accurate results, we repeat the same experiment for a hundred times such that a different set of five samples are randomly selected each time. We then compute the average.

The results reflect that given the opportunity to learn from a small amount (only five samples) of data from the unseen botnet category, the models are able to substantially improve performance. Most importantly, we note that the Recall for Htbot increased from 28.6% in the previous experiment to 62.8% here. Furthermore, this improvement was achieved with only five Htbot samples in the training set that has a malicious sample size of 1115. There are still 282 Htbot samples remaining in the test set to provide basis for reliable validation. With this we may observe that although the use of neural networks in this method does not produce the best results, it has the potential to perform much better than the previous semi-supervised manner when it is used in supervised manner (i.e., newer data available for learning). In addition, this method removes the need for further engineering of features, making it a simpler task.

VI. Conclusion

In this work, we explored machine learning approaches for network anomaly detection, in particular the detection of anomalous botnet traffic. Specifically, we refer to a manual evidence gathering framework and replaced certain components with machine learning models to improve its performance and efficiency. Our experiments show that by using DT classifier in place of the existing threshold-based decision maker, the recall of the framework is substantially increased while lowering the FPR to almost zero. We also experimented further by using a combination of MLP models and traditional classifiers. These replaced the core of the framework and only required the use of its basic features. While this method did not produce even better results, we were able to show the merits of supervised machine learning learning in the experiments. For future work, we aim to curate larger and different datasets for more thorough evaluation of our deep learning methods.

Acknowledgment

This material is based on research work supported by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034

References

[1] Ddos attacks in q1 2018. https://securelist.com/ddos-report-in-q1-2018/85373/. Accessed: 2018-04-30.

[2] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.*, 60(C):19–31, January 2016.

[3] W. Chen, X. Luo, and A. N. Zincir-Heywood. Exploring a service-based normal behaviour profiling system for botnet detection. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 947–952, May 2017.

[4] Dinil Mon Divakaran, Kar Wai Fok, Ido Nevat, and Vrizlynn L.L. Thing. Evidence gathering for network security and forensics. *Digit. Investig.*, 20(S):S56–S65, March 2017.

[5] Sebastián García, Alejandro Zunino, and Marcelo Campo. Survey on network-based botnet detection methods. *Sec. and Commun. Netw.*, 7(5):878–903, May 2014.

[6] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 139–154, Berkeley, CA, USA, 2008. USENIX Association.

[7] F. Haddadi and A. N. Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems Journal*, 10(4):1390–1401, Dec 2016.

[8] Amuthan Prabakar Muniyandi, R. Rajeswari, and R. Rajaram. Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm. *Procedia Engineering*, 30:174 – 182, 2012. International Conference on Communication Technology and System Design 2011.

[9] I. Nevat, D. M. Divakaran, S. G. Nagarajan, P. Zhang, L. Su, L. L. Ko, and V. L. L. Thing. Anomaly detection and attribution in networks with temporally correlated traffic. *IEEE/ACM Transactions on Networking*, 26(1):131–144, Feb 2018.

[10] Federico Simmross-Wattenberg, Juan Ignacio Asensio-Perez, Pablo Casaseca-de-la Higuera, Marcos Martin-Fernandez, Ioannis A. Dimitriadis, and Carlos Alberola-Lopez. Anomaly detection in network traffic based on statistical inference and alpha-stable modeling. *IEEE Trans. Dependable Secur. Comput.*, 8(4):494–509, July 2011.

[11] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. Unsupervised clustering approach for network anomaly detection. In Rachid Benlamri, editor, *Networked Digital Technologies*, pages 135–145, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[12] Gautam Thatte, Urbashi Mitra, and John Heidemann. Parametric methods for anomaly detection in aggregate traffic. *IEEE/ACM Trans. Netw.*, 19(2):512–525, April 2011.

[13] P. Torres, C. Catania, S. Garcia, and C. G. Garino. An analysis of recurrent neural networks for botnet detection behavior. In *2016 IEEE Biennial Congress of Argentina (ARGENCON)*, pages 1–6, June 2016.

[14] Juan Vanerio and Pedro Casas. Ensemble-learning approaches for network security and anomaly detection. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, Big-DAMA '17, pages 1–6, New York, NY, USA, 2017. ACM.

[15] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*, pages 712–717, Jan 2017.

[16] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi. Real-time network anomaly detection system using machine learning. In *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 267–270, March 2015.