2014 28th International Conference on Advanced Information Networking and Applications Workshops

Botnet Behaviour Analysis using IP Flows

With HTTP filters using classifiers

Fariba Haddadi, Jillian Morgan, Eduardo Gomes Filho, A. Nur Zincir-Heywood
Computer Science, Dalhousie University
Halifax, NS, Canada
{haddadi, jmorgan, filho, zincir@cs.dal.ca}

***Abstract*— Botnets are one of the most destructive threats against the cyber security. Recently, HTTP protocol is frequently utilized by botnets as the Command and Communication (C&C) protocol. In this work, we aim to detect HTTP based botnet activity based on botnet behaviour analysis via machine learning approach. To achieve this, we employ flow-based network traffic utilizing NetFlow (via Softflowd). The proposed botnet analysis system is implemented by employing two different machine learning algorithms, C4.5 and Naive Bayes. Our results show that C4.5 learning algorithm based classifier obtained very promising performance on detecting HTTP based botnet activity.**

Keywords— botnet detection; traffic IP- flow analysis; machine learning based analysis

I. INTRODUCTION

With the high reported infection rates, the vast range of illegal activities and powerful comebacks, botnets are one of the main threats against the cyber security [1][2]. A botnet is a set of compromised hosts (bots) that are under the remote control of a botmaster. Botnets are mainly used for spreading spams, Distributed Denial of Service (DDoS) attacks, identity thefts or just making use of the victim's computational resources.

Researchers have proposed various detection mechanisms against botnets. However, botmasters have started to employ several protocols with decentralized topologies and fluxing techniques to avoid detection. In this regard, they evolved their communication methodology from utilizing Internet Relay Chat (IRC) to taking advantage of more ubiquitous protocols and decentralized topologies such as Hypertext Transfer Protocol (HTTP) and Peer-to-Peer (P2P). Therefore, it is anticipated that a botnet detection mechanism, which could potentially learn the new patterns by analyzing the traffic, could adapt to the changes in the botnet evolution.

Many existing botnet detection approaches rely on analyzing the network traffic. Some focus on specific types of botnets while others attempt to build a general model for a few types of botnets. Before 2009, most of the proposed systems were specifically designed for IRC botnets (e.g. [3], [4]) but recently researches are more focused on P2P-based and HTTP-based botnets (e.g. [6], [9]). They employ machine learning (ML) techniques (i.e classification and clustering) to automatically generate botnet detection models. In such systems, the first step is to collect the required information from the network traffic (feature extraction) which has always been a challenge. To this end, some only use network packet headers (i.e. [4], [6]), while others use packet payloads or a combination of the two (i.e. [5], [8]). As a response, botnets employ encryption techniques to avoid the detection systems that analyze the communication information embedded in the packet payload. Therefore, in this work, we investigate a ML based botnet detection mechanism that does not use packet payload information, which is opaque when encrypted. Instead, to summarize the network traffic and extract the required features, we employ a flow exporting tool utilizing network packet headers. Exporter tools are used to aggregate packets into flows and then represent them with features. NetFlow, which is introduced by Cisco Systems and is the de-facto standard in IP-flow collection, is employed in this work using an open source flow exporter called Softflowd [19]. We evaluated the performance of our proposed system on two HTTP based botnets both with and without a HTTP traffic filter.

The rest of the paper is structured as follows: Section II summarizes the related work on botnet detection. Our methodology and the proposed system are presented in Section III. Results are discussed in Section IV. Finally, conclusions are drawn and the future work is discussed in Section V.

II. RELATED WORK

Different detection mechanisms have been proposed focusing on different aspects of botnets. Strayer et al. developed an IRC botnet detection framework that makes use of machine learning techniques [4]. First, they used a classification technique to filter the chat traffic and then a clustering technique was utilized to identify different activities. Finally, a topology analyzer is applied to the clusters to detect the botnets. In this three layer approach, flow-based features such as the number of packets and variance of packet intervals were utilized. Zeidanloo et al. proposed a detection system focusing on P2P and IRC based botnets [5]. Using filtering, classification and clustering methods, they differentiate the group behaviour of botnets from the legitimate traffic. Gu et al. developed BotMiner based on group behaviour analysis for IRC and P2P [7]. This detection system first uses a clustering approach to find similar communication (for the C&C communication traffic) then employs an activity clustering (via Snort) to find the type of activity. Wurzinger et al. proposed to detect botnets based on the correlation of commands and

978-1-4799-2652-7/14 $31.00 © 2014 IEEE
DOI 10.1109/WAINA.2014.19

responses of the monitored network [8]. By first identifying the responses, they aim to locate the corresponding commands in the traffic. In this process, packet payload information was utilized. Then, using these command and response pairs, the detection model (based on behaviour similarity) is built and put up for botnet detection purposes. They evaluated their system on IRC and P2P bot binaries and show that their system gives 88% accuracy on the bots they employed. Kirubavathi et al. designed specifically an HTTP based botnet detection system using a multilayer Feed-Forward Neural network [9]. Based on the fact that web-based botnets do not maintain a connection with the C&C server but periodically make a web request from the C&C web server to download the instructions, they extracted features related to TCP connections in specific time intervals and used them for the detection purposes. Their results on simulated HTTP based botnets showed promising accuracy. Haddadi et al. proposed a co-evolutionary system, called Stateful-SBB, to detect automatically generated malicious (botnet) domain names [10]. Authors showed that, Stateful-SBB could differentiate botnet C&C domain names, which are located in the network packet payload. Francois et al. proposed a NetFlow monitoring framework that leveraged a simple host dependency model to track communication patterns and employed linkage analysis and clustering techniques to identify similar botnet behavioral patterns [11]. Zhao et al. investigated a botnet detection system based on flow intervals [6]. Flow features extracted from packet headers were utilized with Bayesian networks and decision tree classifiers to detect the botnets.

In this work, we employ a NetFlow exporter, called Softflows, to extract the traffic features and two machine learning techniques utilizing the extracted features to detect botnet behaviour. Give that we only use flow based features, our approach can be employed for encrypted botnet traffic classification, too. Moreover, we employ all the possible extracted flow features in order to enable the ML technique in use to select the most appropriate features. This in return enables us to analyze the different botnet behaviours without a priori information.

III. METHODOLOGY

As discussed earlier, in this work, we aim to detect botnets via machine learning classifiers assuming that they employ different types and levels of encryption in their communication. Therefore, we do not have access to the packet payload. Thus, we explore the possibility of detecting botnets by aggregating the network traces into flows using network flow exporters which only utilize the information in the packet headers. Furthermore, we study the potential features chosen by the classifier from a given set. To achieve this, we built the following modules for our proposed system: (a) Traffic Generation module, (b) Feature Extraction module, and (c) Traffic Classification module.

A. Traffic Generation

Since botnets employ fluxing techniques as their strength point for their new versions [12], our main focus in this work is to detect botnet behaviour, which uses domain fluxing techniques. Although these types of botnets are the most seen ones in the field recently, there is no such traffic publically available. Thus, we employed publically available lists of C&C domain names and legitimate web site domain names to generate such representative traffic.

1) Alexa: Alexa Internet Inc. ranks and publishes the list of the most popular websites according to the number of unique site users and page views [13]. In this work, we employed this list to generate representative normal traffic.

2) Zeus: Zeus botnet stole more than 100 million dollars over the years. Serious actions took place to take down Zeus botnets by Microsoft and its collaborators in March 2012. However, there are security reports in 2013 indicating that Zeus botnet is back with a new variant [14]. This makes the Zeus botnet one of the most destructive botnets in the recent history. ZeusTracker initiative actively monitors the Zeus botnet. In this work, we downloaded the C&C domain name list that we used from the ZeusTracker and DNS-BH web pages [15][16].

3) Citadel: Citadel botnet entered the Internet realm after the Zeus botnet leaked in 2011. Since then, this botnet has stolen more than 500 million dollars and infected more than 5 million PCs in different countries [1][17]. In this work, we obtained the lists of Citadel botnet C&C domain names from the Citadel botnet section of the ZeusTracker and DNS-BH [15][16].

Both the Citadel and Zeus botnets use the HTTP protocol for their C&C communications. Alexa domain names are the high ranked web pages that can be accessed through the HTTP protocol, too. So, we coded a script to initiate HTTP connections with the domain names from the aforementioned lists to generate the representative traffic. In the process, all the generated traffic was captured and no sampling was applied.

B. Flow Generation

Flow generation tools summarize traffic utilizing the network packet headers. These tools collect packets information with common characteristics such as IP addresses and port numbers, aggregate them into flows and then calculate some statistics such as the number of packets per flow etc. Cisco NetFlow V.9, i.e. RFC 3954, defines the flow as "a unidirectional sequence of packets with some common properties that pass through a network device" [18][19]. The most common way to identify an IP flow is using a combination of five properties, i.e. 5-tuple: Source / Destination IP addresses, Source / Destination Port Numbers, and Protocol.

Cisco Systems introduced NetFlow to collect and aggregate IP traffic information. Given that Cisco is the leader of IP flow technology, soon NetFlow became an industry standard. To collect and analyze NetFlow traffic data, three components should work together: (i) NetFlow exporter, which generates the NetFlow data, (ii) NetFlow collector, which collects the NetFlow data from the exporter, and (iii) NetFlow analyzer which analyzes the collected data, Fig. 1.

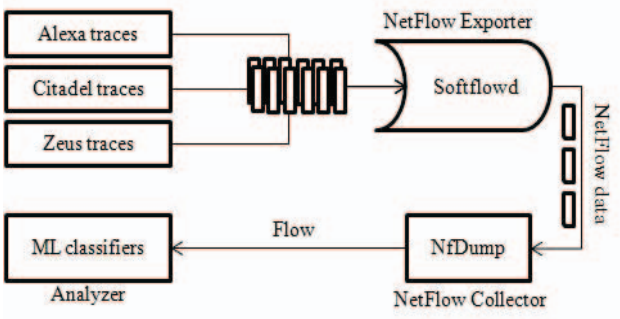

Fig. 1. NetFlow traffic analysis

In this work, we employ Softflowd [20], which is an open source tool, as the Netflow exporter. Softflowd can export uni-directional NetFlow data using the traffic on a simple device interface or from a pre-captured traffic trace. The exported data by SoftFlowd is then collected and analyzed. We chose NfDump [21] as the collector because it is an open source, easy to use tool which supports different versions of NetFlow. Finally, for the analyzer, we employed our proposed machine learning based classifiers and visual analysis to study the traffic.

C. Classifiers

In this work, two well-known machine learning based classifiers are used: C4.5 and Naïve Bayes.

a) C4.5: C4.5 is a decision tree algorithm, which is a non-parametric supervised learning method, that is an extention to ID3 decision tree algorithm. In this case, the decision tree aims to find the small trees by pruning, then converts the trained tree into an if-then rule set.

C4.5 constructs decision trees based on a training data set, where each exemplar is a set of already classified features by applying the Information Entropy concept. The algorithm employs a normalized information gain criterion to select features from a given set to determine the splitting point. In other words, the feature with the highest information gain value is selected as the splitting point, Eq. 1.

Let P_i be the probability of an arbitrary sample in data set D belonging to class C_i:

$$p_i = \frac{|C_{i,D}|}{D} \quad (1)$$

Then, the amount of information that is required to classify an instance in D, where m is the number of unique instances of the data set, is given by Eq. 2:

$$Entropy(D) = -\sum_{i=1}^{m} p_i log_2(p_i) \quad (2)$$

Expected information needed to classify the objects of the data set D in all v sub-trees after using the feature A to split D into v partitions is given by Eq. 3:

$$Entropy_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{D} \times Entropy(D_j) \quad (3)$$

Finally, the information that is gained by branching on the feature A is given by Eq. 4:

$$Gain(A) = Entropy(D) - Entropy_A(D) \quad (4)$$

After finding the splitting node with the highest information gain, a decision node is generated based on this selected point. The training process then recursively continues on the corresponding sub-lists that are obtained until all of the data samples associated to the leaf nodes are of the same class or the classifier runs out of training samples. More detailed information on C4.5 learning algorithm can be found in [22].

b) Naïve Bayes: A Naïve-Bayes classifier is a simple probabilistic classifier based on the Bayes theorem, which assumes that the presence of a feature in a given class is independent of other features. The classifier uses the method of maximum likelihood (probability) for parameter estimation. Given a training set *(X, Y)* where for each sample *(x, y), x* is an n-dimensional vector and *y* is the class label out of *k* number of classes, $C_1, C_2 \ldots C_k$, the classifier predicts that the sample belongs to the class C_i having the highest posteriori, conditioned on *x*:

$$P(C_i|x) > P(C_j|x) \text{ for } 1 \leq j \leq k, j \neq i \quad (5)$$

where:

$$P(C_i|x) = \frac{P(C_i)\,P(x \mid C_i)}{P(x)} \quad (6)$$

which equals to:

$$Posterior = \frac{Priori \times Likelihood}{Evidence} \quad (7)$$

All classifier parameters (i.e. class priori) can be calculated using different assumptions (i.e. priories = *1/k* where *k* is the number of classes). A more detailed explanation of the algorithm can be found in [22].

IV. Evaluations

As discussed earlier, the proposed system is evaluated on two botnets (Citadel and Zeus) employing two machine learning classifiers (C4.5 and Naïve Bayes) using traffic flows generated by Softflowd. Moreover, the performance of the proposed system is also evaluated both with and without a HTTP filter.

A. Data sets

Given that even Alexa lists might have malicious domain names [23], we manually extracted 500 benign domain names from Alexa lists for the data sets employed in this work[1]. For Zeus botnet, we employed the list from ZeusTracker and DNS-BH blocklist [15][16]. For Citadel botnet, we employed the ZeusTracker and DNS-BH Citadel list [15][16]. Table I presents the number of domain names with which we communicated and captured the traffic.

Once, we communicated with the C&C domain names over the HTTP and captured the traffic, Softflowd flow exporter tool is employed on the captured traffic. Table II presents the number of extracted flows by Softflowd on all traffic and Table III presents the number of flows extracted by Softflowd once the HTTP filter is in effect.

It should be noted here that Softflowd provides 41 features

[1] http://web.cs.dal.ca/~haddadi/alexa-list.pdf

TABLE I. NUMBER OF DOMAIN NAMES AND NUMBER OF GENERATED PACKETS

Data Sets	# of Domain Names	# of Packets
Alexa	500	21210
Zeus	684	108947
Citadel	42	79516

TABLE II. NUMBER OF ALL FLOWS EXPORTED BY SOFTFLOWD

Data Sets	# of All Flows
Alexa	7473
Zeus	14884
Citadel	5772

TABLE III. NUMBER OF HTTP FLOWS EXPORTED BY SOFTFLOWD

Data Sets	# of HTTP Flows
Alexa	2899
Zeus	5237
Citadel	1921

TABLE IV. SOFTFLOWD FEATURES

Feature set	
Flow duration	Type of Service (ToS)
Source AS number	Source ToS
Destination AS number	Destination ToS
Input Interface	Source mask
Output Interface	Destination mask
Total number of packets	Forwarding status
Forward number of packets	Source Vlan label
Backward number of packets	Destination Vlan label
Total number of bytes	Bits per second
Forward number of bytes	Packets per second
Backward number of bytes	Bytes per packet
Number of aggregated flows	

in total. Detailed definition of these features can be found in NfDump and Softflowd project web sites [20][21]. We employed 23 of these features as inputs to our proposed system, Table IV. The ones that we omit are: IP addresses, port numbers and any non-numeric features. The reasons behind this are: IP addresses can be spoofed whereas port numbers can be assigned dynamically. Thus, employing such features may decrease the generalization abilities of a classifier for unseen behaviours. On the other hand, the presentation of non-numeric features may introduce other biases to the detection system so it is left to future work to include the non-numeric features.

B. A High Level Look at the Flow Data sets

As discussed in Section III-B, traffic is analyzed by a flow analyzer (3rd component) after being exported (1st component) and collected (2nd component). Although analyzers employ complex analysis techniques and different views to illustrate the flows in a more understandable way for the analyst (aka security expert or network administrator/manager), we thought it might be more beneficial to analyze the data set on some of the most important features of the flow (used by other researchers [4][6][8]) to understand whether there is any kind of data set bias or anomaly.

Because of the page limitation, it is not possible to illustrate all of our observations in this context. So, we just present the most illustrative ones on only the HTTP flows of Softflowd. The analysis of these flows shows that:

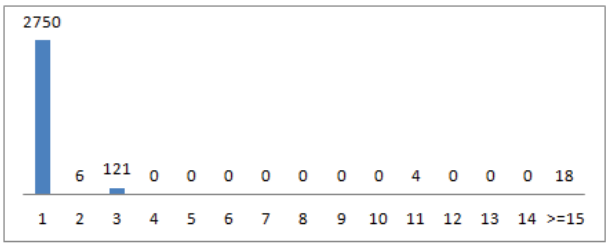

Fig. 2. Alexa- Frequency of flow Duration (Buckets of 50 sec)

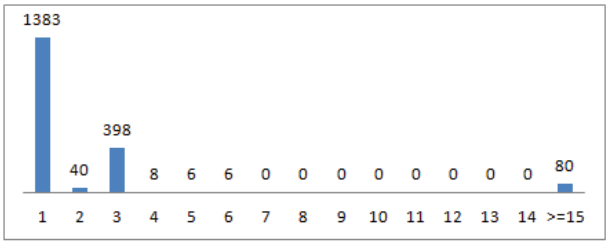

Fig. 3. Citadel- Frequency of flow Duration (Buckets of 50 sec)

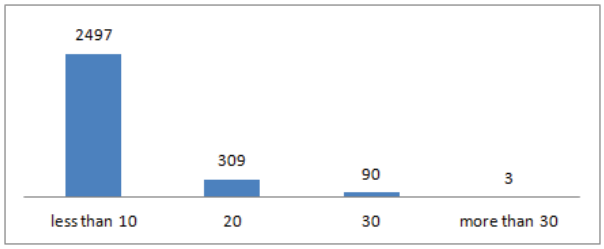

Fig. 4. Alexa- Frequency of number of packets per flow (Buckets of 10 sec)

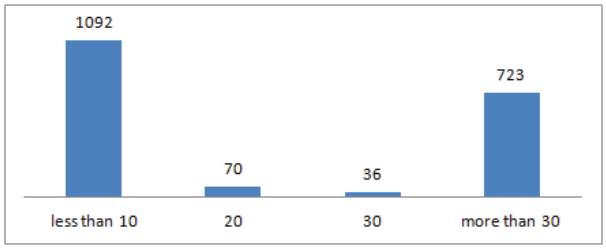

Fig. 5. Citadel- Frequency of number of packets per flow (Buckets of 10 sec)

- 95% of the flows' durations are between 0 to 50 seconds in Alexa-HTTP. These represent the normal behaviour, Fig 2. In Zeus-HTTP, 95% of flows' durations are also in this range. However, in Citadel- HTTP only 72% of the flows are in this range. This indicates that 23% of Citadel-HTTP connections are longer than Alexa-HTTP (normal behaviour), Fig 3.
- In Alexa-HTTP, almost all of the flows include less than 30 packets, Fig. 4. Zeus-HTTP flows also include about 30 packets or less, Fig. 5. On the other hand, in Citadel-HTTP, 62% of the flows are in the same range as Alexa-HTTP flows.

TABLE V. CLASSIFICATION RESULTS-- WITHOUT THE HTTP FILTER

	Data set	DR	Botnet		Legitimate		Time Complexity (sec)
			TPR	FPR	TPR	FPR	
C4.5	Citadel	88%	88%	12%	88%	12%	0.61
	Zeus	86%	87%	16%	84%	13%	0.74
Naïve Bayes	Citadel	78%	7%	1%	99%	93%	0.06
	Zeus	54%	10%	1%	99%	90%	0.09

TABLE VI. CLASSIFICATION RESULTS-- WITH THE HTTP FILTER

	Data set	DR	Botnet		Legitimate		Solution Complexity	Time Complexity (sec)	Feature Complexity
			TPR	FPR	TPR	FPR			
C4.5	Citadel	97%	97%	3%	97%	3%	83	0.11	6
	Zeus	86%	86%	15%	85%	14%	223	0.16	8

Our high level look to the data sets indicates that differentiating Zeus-HTTP botnet and Citadel-HTTP botnet flows from Alexa-HTTP flows will not be as easy as one might expect. As can be seen in Fig. 2 -5, specially Zeus botnet seems to be very similar to Alexa normal behaviour in terms of high-level statistics. This may result in high false alarm rates or high solution complexities for the classifiers when they aim to differentiate Zeus botnet behaviour from normal behaviour.

C. Performance Metric for the Proposed System

In this work, we used the following metric in our evaluations:

1) Detection Rate (DR): DR is the fraction of all the correctly labeled instances.

2) False Positive (FP) and True Positive (TP) Rates: In general, positive means "identified" and negative means "rejected". Therefore, FP means incorrectly identified and TP means correctly identified. Thus, FP Rate (FPR) means the ratio of incorrectly identified samples and TP Rate (TPR) means the ratio of correctly classified samples of each class.

3) Complexity: The definition of complexity often depends on the concept of the "system". Speaking of classifiers, complexity can be measured on different criteria such as memory consumption, time or solution. In this work, three complexity criteria are utilized. Firstly, computation time, which is a typical scale for learning algorithms during training procedure denoted as training time. After a classifier is trained, the trained model is presented as the solution to be used for testing purposes on unseen data. Given that presenting a better solution to a problem is important, we define the solution complexity as our second criteria. This is the tree size for the C4.5 based classifier. Given that solutions which employ less number of features might be beneficiary in the case of summarizing the data set, we finally define our last complexity parameter, called feature complexity, as the number of distinct features that are used as part of a solution.

D. Results

We implemented C4.5 and Naïve Bayes learning algorithms via an open source tool called Weka [24]. We compare these two learning classifiers because they represent two well-known categories of ML. While C4.5 represents a decision tree that generates a solution in the form of rules, which are more understandable by the human expert, Naïve Bayes presents more of a black box solution. Moreover, while C4.5 has the ability of choosing the most appropriate features from all the features given to it, Naïve Bayes does not have this ability. Such an ability of C4.5 enables any analysis that can be done post classification. To evaluate these classifiers on our data sets, first an equal number of flows were randomly selected (using the uniform random selection algorithm in Weka) from the non-malicious data set (Alexa) as well as from each of the malicious data sets (Zeus and Citadel). Classifiers were then run on these balanced data sets using 10-fold cross-validation to further avoid any data set biases that might affect the results.

Table V shows the classification results on these traffic flows without using a HTTP filter. These results clearly demonstrate that C4.5 is a better choice for our classification purposes even though it takes longer to run. Although C4.5 DRs in these evaluations are good, however the FPRs are for higher than desired. Thus, as the next step, we employed a HTTP filter on the data sets to analyze whether it would have any effect on the DR and/or FPR.

1) HTTP Filtering: Given the wide range of the HTTP usage on the Internet, most recent botnets employ HTTP protocol to hide their malicious activities among the normal web traffic, Fig. 2 - 5. Citadel and Zeus fall under this category, too. Their C&C channels utilize HTTP protocol to communicate with their bots. Therefore, to investigate the effect of protocol filtering on botnet detection, specifically on false alarm rates, we employed a HTTP filter to select only HTTP related traffic. Then we repeat our previous approach to train our detection model and evaluate it again.

Table VI shows the classification results for Citadel, and Zeus botnets when the HTTP filter is employed on the traffic. The results indicate that: (i) Filtering the HTTP traffic, in other words classifying the traffic first based on the application protocol suspected that a botnet uses seems to be effective, specifically in terms of FPR and time complexity metrics. (ii) Higher than desired FPR in Zeus classification shows that differentiating Zeus traffic form Alexa normal traffic is a challenging task. This was also seen in the analysis we discussed in Section IV-B. Probably a different way of traffic representation is necessary for Zeus botnet, given that Softflowd traffic representation with traffic filtering can achieve up to 97% detection rate and 3% false alarm rate for the Citadel botnet but much less for the Zeus botnet. This

means that there is a suitable feature set in Softflowd that can very well represent Citadel botnet behaviour and help to differentiate it from Alexa normal behaviour, but the same set of features are not effective for Zeus botnet. (iii) Based on our post-classification analysis, only six features of Softflowd were common in C4.5 solutions. These six features are: Flow duration, Total number of packets, Total number of bytes, Number of bits per seconds, Number of packets per seconds and finally Number of bytes per packet. This may indicate that these six features represent the similar behaviours between the two botnets but additional features are used to represent each one's unique behaviours.

In summary, we think that these are very promising results. Our proposed system detects not only the botnet behaviour without using IP addresses, port numbers and payload information but it can also provide a platform to determine the most appropriate features indicating the botnet behaviour that is under analysis. Although, our proposed system is only evaluated on Zeus and Citadel botnets (recent aggressive botnets), since we employ no a priori information, it can be applied easily to other types of botnets with or without encrypted traffic.

V. CONCLUSION

Due to the high reported botnet infection rate and its wide range of illegal activities, botnets are one of the main threats against the cyber security. In this scope, Citadel and Zeus are the two most powerful botnets that have affected the legitimate Internet realm the most in the past few years. In this work, two well-known machine learning techniques are investigated on these two well-known botnets for the purpose of botnet detection in traffic flows without using IP addresses, port numbers or any payload information. Since ML classifiers cannot be applied on network traffic directly, Softflowd, which is an open source tool to generate NetFlow based traffic, is employed on the captured packets to convert them into network flows and extract their features. Then, two approaches are employed: (i) Employing all the flows (without any HTTP filter); and (ii) Employing just the HTTP flows (with a HTTP filter).

Given that Citadel and Zeus botnets are HTTP-based botnets and Alexa traffic is also a representation of HTTP normal traffic, the effect of a HTTP filter was investigated. Indeed, the use of a HTTP filter keeps the core of the botnet/Alexa communication and discards the rest. Therefore, the results of the second approach resulted in better performance for detecting both of the botnets and thereby show the importance and the effect of the HTTP filtering. In short, our results indicate how a machine learning based system and the choice of features can affect the results of botnet identification in traffic flows. Moreover, these results also indicate the importance of traffic classification/filtering for determining malicious behaviour.

Future work will follow studying different flow exporter and feature extraction techniques as well as a more in depth study of protocol (application) filtering.

ACKNOWLEDGMENT

This research is supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) Engage Grant, and Solana Networks Inc. The research is conducted as part of the Dalhousie NIMS Lab at https://projects.cs.dal.ca/projectx/.

REFERENCES

[1] Citadel's defences breached, (2013) [Online]. http://www.symantec.com/connect/blogs/citadel-s-defenses-breached

[2] "The Role of DNS in Botnet Command & Control," Open DNS Inc., Whitepaper 2012.

[3] C. Livadas, R. Walsh, D. Lapsley, and W.T. Strayer, "Using machine learning techniques to identify botnet traffic," in 2nd IEEE LCN workshop on network security, pp. 967-974, 2006.

[4] W.T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," *Advances in Information Security*, vol. 36, pp. 1-24, 2008.

[5] H.R. Zeidanloo, A. Bt Manaf, P. Vahdani, F. Tabatabaei, and M. Zamani, "Botnet detection based on traffic monitoring," in Networking and Information Technology (ICNIT) , pp. 97-101, 2010.

[6] D. Zhao, I. Traore, A. Ghorbani, B. Sayed, S. Saad, and W. lu., "Peer-to-Peer botnet detection based on flow intervals," in IFIP international information security and privacy, pp. 87-102, 2012.

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: clustering analysis of network traffic for protocol- and structure- independent botnet detection," in 17th USNIX Security symposium, pp. 139-154, 2008.

[8] P. Wurzinger, L. Bilge, Th. Holz, J. Goebel, Ch. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in 14th European conference on research in computer security (ESORICS) , pp. 232-249, 2009.

[9] V. Kirubavathi and R.A. Nadarajan, "HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network," in Information Security Theory and Practice: security, privacy and trust in computing systems and ambient intelligent ecosystems, pp. 38-48, 2012.

[10] F. Haddadi and A.N. Zincir-Heywood, "Analyzing string format-based classifiers for botnet detection: GP and SVM," in IEEE Congress on Evolutionary Computation (CEC) , pp. 2626-2633, 2013.

[11] J. Francois, Sh. Wang, R. State, and Th. Engel, "BotTrack: tracking botnets using Netflow and PageRank," *Networking*, vol. 6640, pp. 1-14, 2011.

[12] PushDo evolves again: enhances evasion with Domain Generation Algorithm (DGA), (2013). [Online]. https://www.damballa.com/downloads/r_pubs/WP_PushDo_Evolves_Again.pdf

[13] Alexa. [Online]. http://www.alexa.com/topsites

[14] Zeus/ZBot Malware Shapes up in 2013. (2013) [Online]. http://blog.trendmicro.com/trendlabs-security-intelligence/zeuszbot-malware-shapes-up-in-2013/

[15] Abuse: Zeus Tracker. [Online]. https://zeustracker.abuse.ch/

[16] DNS-BH- Malware Domain Blocklist. [Online]. Available: http://www.malwaredomains.com/

[17] Microsoft, financial services and others join forces to combat massive cybercrime ring, (2013). [Online]. http://www.microsoft.com/en-us/news/press/2013/jun13/06-05dcupr.aspx

[18] Cisco IOS NetFlow. [Online]. http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

[19] RFC 3954, (2004). [Online]. http://www.ietf.org/rfc/rfc3954.txt

[20] Softflowd. [Online]. http://www.mindrot.org/projects/softflowd/

[21] NfDump. [Online]. http://nfdump.sourceforge.net/

[22] E. Alpaydin, Introduction to Machine Learning.: MIT Press, 2004.

[23] Paul Royal. Maliciousness in Top-ranked Alexa Domains. [Online]. https://www.barracudanetworks.com/blogs/labsblog?bid=2438

[24] WEKA. [Online]. http://www.cs.waikato.ac.nz/ml/weka/