Detecting Command and Control Channels of a Botnet Using a N-packet-based Approach

Félix Brezo, José Gaviria de la Puerta and Pablo G. Bringas
DeustoTech Computing – University of Deusto
Bilbao, Spain
Email: {felix.brezo, jgaviria, pablo.garcia.bringas}@deusto.es

***Abstract*—The botnet phenomenon is one of the major threats in nowadays cyberspace. The ability of malware writers to code profitable applications with a softened learning curve is forcing public and private organisms to take measures against these infections. In this paper, we propose a method to identify traffic belonging to the Command & Control channels from a botnet. Our method takes into account the attributes of the packets captured from a connection to build vectorial representations of the connection by appending them into sequences of packets. Thus, we provide an empirical study of how these representations can be used to detect such a communicative behaviour by considering the issue as a supervised classification problem and comparing the results obtained by more than 20 machine learning algorithms.**

***Keywords*—botnet detection; n-packets; supervised learning; traffic analysis**

I. THE BOTNET THREAT

The origin of the term *botnet* is commonly set in the fusion of the concepts of *robot networks*. In this way, botnets, as collections of infected machines remotely controlled by cybercriminals, are to naturally evolve into more complex entities that will have to be taken into account by private and public organizations. We can define this phenomenon as the new generation of malware that brings to light the profitable business of obscure economies in the deep web.

They have become a relevant issue to the different organizations internationally dedicated to computer security. Europol and NATO/OTAN are just two revealing examples. The former started to train their professionals to face the threats that these threads bring with them to privacy, anonymity and company's and public administration's security. The latter was forced in 2008 to create an observatory so as to watch for the rights of the allies in the cyberespace after the systematic attack suffered by Estonian cyberfacilities in 2007 [1].

What is certain is the fact that controlling the vast number of computers kidnapped by some of the biggest botnets and their potential computing power have not passed over neither for computer's professionals dedicated to code distributed solutions [2] nor for malware writers specific targets: monetizing massive infections [3]. In this way, botnets, as collections of infected machines remotely controlled by cybercriminals, are to evolve in complexity to constitute a threat even bigger in the near future.

To achieve this goal, the communications between bots have suffered major changes since its origins. Malware designers have found a hot topic on being able to cope with scalability and fault-tolerance. Thus, it is precisely a mechanism capable of maintaining a continuous communication with zombies what would determine the topology of the network, its capacity to avoid detection and disruption and the complexiy of the protocols defined to face this issues [4].

That is how the old-fashioned IRC (Internet Relay Chat) clients were brought in, often created on-the-fly by malware writers so as to open the doors to more complex solutions to boost anonymity and usability. This very last issue is one of the main catalysts of the concept *Hacking as a Service* —also known by some authors as *Crime as a Service* — as a malicious evolution from its benign and profitable branch, *Software as a Service* philosophy. Malware writers, conscious of the profitability of coding malicious applications for third parties, are adapting their tools so as to soften as much as possible the learning curve of the final user to widen their target market. That was the case of *Mariposa* botnet (Butterfly botnet in Spanish), dismantled in 2010 in the framework of of an operation conducted by the Spanish Guardia Civil and coordinated with different European organisms and Panda Software. In them, three people with little technical formation were arrested accused of being the botmasters behind a network of almost four million computers.

Against this background, we have advanced the state of the art with the following contributions:

- We show a method to model traffic connections by using the attributes inherent to the subsequent packets from such connections.
- We provide empirical validation of our method with a study that explores the capability of such representation model to identify Command & Control communications performed by some HTTP botnets.
- We show how the proposed method achieves high detection rates and how these could be used to identify infections. At the same time, we also discuss the shortcomings of the proposed approach and suggest future lines of work that might be explored in the near future.

The remainder of this paper is structured as follows. Section II describes the representation model used in this article. Section III states the methodology applied to evaluate the method as well as the results obtained in the experiments performed. Section IV analyses the implications of the aforementioned results as well as outlines some future lines of work.

 ISBN: 978-1-61208-311-7

Finally, Section V collects the conclusions to be extracted from this article.

II. TRAFFIC MODELLING

Although other approaches have tried to fight botnets by applying transformations to the data fields from a given connection [5], in this paper we are going to perform a more accurate atomization of the characteristics of a connection by grouping representations as sequences of consecutive packets from a connection and analysing their detection capabilities depending on the length of these sequences.

In previous work [6], we have demonstrated that the analysis of the characteristics of packets on their own could be used as indicators of the kindness of a communication. While we considered the observation of a single packet as the obtention of a snapshot of the state of the connection at a time t_0, we also suggested that the monitoring of the evolution of these characteristics of a connection over time would be pretty more accurate. As a result, we have developed a method to represent such an evolution by employing the attributes of the packets observed over time to create a time-dependant representation model. To achieve this objective, we are going to build representations that take into account the concatenation of the individual characteristics of each packet in a chronologically ordered connection. However, to avoid biases we would perform some previous transformations to the traffic sniffed by hiding information relative to IP addresses, MAC addresses or ports as their inclusion would lead to a problem which is trivially solved once an infected address is identified. In Algorithm 1, we show the pseudo-code relative to the construction of sequences of up to n packets that lead to the obtention of the combined representations.

Given a dataset of connections C compounded by c connections from which we have observed a series of packets, the representation generation process of up to $n = 12$ will go through each and every of the c connections creating representations by appending the attributes of n consecutive packets. Additionally we have added as part of the representation some complementary temporal variables of different measures of central tendency and dispersion. The idea is to use them to represent the temporal intervals that passed between the reception of one packet and the following used in the same representation.

We benefit from having the calculation of such gap as a trivial task, taking into account that any sniffing tool provides the timestamp of when a packet was observed. However, considering this value in absolute terms may lead us to an error. If we only considered the time by itself we could end up introducing a bias in the dataset that can use this parameter in a way that we do not like: identifying connections by the *distance* (understood as *travelling time*) to/from the server. This is not an interesting point because we may be developing a system capable of detecting *distances* instead of using the rest of characteristics inherent to the packets. We want our approach to be more general, avoiding the analysis of the content of the studied connections. The way in which we have addressed this issue is the following: we have considered relative times instead of absolute times. There are two special cases to this rule. For a sequence length of $n = 1$ the spatio-temporal characteristics are not taken into account, whilst for a sequence length of $n = 2$, the values obtained will always be $\{1\}$ as a result of the application of the general formula in Equation 1:
$t_1 = \frac{T_2 - T_1}{T_2 - T_1} = 1$.

Input : A bidimensional array A that contains a list of the attributes of the available packets for a connection c in the dataset C

$muestras \leftarrow []$;
foreach $c \in C$ **do**
  $nPaq \leftarrow len(c)$;
  // The process will be performed as many times as the length of the representations was desired. In this case, up to $n = 12$
  foreach $i \in range(2, n)$ **do**
    // In a connection c we could obtain $nPaq - i + 1$ sequences of length i
    foreach $j \in c$ **do**
      $rep \leftarrow []$;
      $tiempos \leftarrow []$;
      // We select the i packets that will compound the representation
      foreach $k \in i$ **do**
        $rep.append(j + k)$;
        $tiempos.append(j + k)$;
      end
    end
    // We work out the internal times t_i
    $attM \leftarrow calculoTi(tiempos)$;
    $rep.append(attM)$;
    // We add the type of the sample as the last attribute of it
    $rep.append(attTipo)$;
    // We add the whole sample to our list of full representations of the dataset
    $muestras.append(rep)$;
  end
end

Output: A bidimensional array containing a list of the attributes of the available sequences for each connection

Algorithm 1: Obtention of the characteristics of a single packet.

To perform these calculations, we have normalized this spatio-temporal measure: for representations using sequences of n packets, we have considered the time passed between the first and the second packet as a unit, while the rest of spatio-temporal measures observed would be weighted accordingly. Given a sequence of n packets observed in the moments $\{T_1, T_2, \cdots, T_n\}$, we would obtain the $n - 1$ spatio-temporal coefficients $\{t_1, t_2, \cdots, t_{n-1}\}$ by applying the following:

$$t_i = \frac{T_{i+1} - T_i}{T_2 - T_1} \quad (1)$$

 ISBN: 978-1-61208-311-7

obtaining a vector of coefficients as the following:

$$t = \{1, \frac{T_3 - T_2}{T_2 - T_1}, \frac{T_4 - T_3}{T_2 - T_1}, \cdots, \frac{T_n - T_{n-1}}{T_2 - T_1}\} \quad (2)$$

To evaluate the evolution of these parameters, we take into account two different points of view: a central measure and a dispersion one. Central measures are tools used to resume the information of a dataset in a single scalar that tries to represent the central point of a dataset. Taking into account that this information corresponds to relative data, the most appropriate central measure is not the arithmetic mean, but the geometric mean. In spite of being commonly used as the most known central measure, the former is not always a strong statist, as it can be broadly influenced by the appearance of atypical values. At the same time, the most appropriate central measure to be used with relative values is the geometric mean [7]. Mathematically, the geometric mean $\bar{x}$ of a collection of n elements $\{a_1, a_2, \cdots a_n\} \in R$ can be expressed as in equation 3:

$$\bar{x} = \sqrt[n]{\prod_{i=1}^{n} x_i} = \sqrt[n]{x_1 * x_2 * \cdots * x_n} \quad (3)$$

This tool is used to determine the variability of the dataset. This kind of measures determines to what extent the different values in a dataset are distant from the central point of the distribution [8], being higher for very disperse values and smaller for those that vary less. In this paper, we are using the standard deviation. For a collection of n elements $\{a_1, a_2, \cdots a_n\} \in R$, the standard deviation σ can be expressed as equation 4:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2} \quad (4)$$

By including these values to our representations, we claim we are avoiding biases that would lead to the sole identification of the distance of the servers expressed in time-to-destiny.

III. Experimental Validation

To evaluate what can be considered a good reference for botnet traffic detection, we have conducted a series of experiments that apply machine learning algorithms. In the following subsections we are describing the methodology applied, the different algorithms selected and the comparison metrics used to evaluate their performance, as well as the results achieved for each and every representation.

A. General Methodology

The proposal developed in this article considers the identification of HTTP communications from a botnet as a supervised classification problem. This election is not arbitrary. The authors will adapt a philosophy similarly applied to the identification of IRC botnets in the past [9], [10]. The latter performed a supervised approach to detect IRC-controlled botnets identifying the problem as a supervised classification task. In this type of problems, we study phenomenons represented by a d-dimensional vector X in R^d that can be classified in K different ways according to a vector Y of *labels* or *classes*. The application of classification algorithms in supervised learning approaches make use of a previously labelled dataset [11] which, in our case, will correspond to the legitimate or botnet labelled traffic samples.

With such objective, we have defined a training dataset D_n as $D_n = \{(X_i, Y_i)\}_{i=1}^{n}$ where X_i represents the events corresponding to phenomenon X while Y_i is the label that classifies it in the category that the classifier assumes as correct. For instance, for the case of sequence lengths of $n = i(\forall 1 <= i <= 12)$ packets, we can identify a sequence X_i defined by a series of attributes that represent it, being Y_i the category assigned to that sequence in accordance to the estimations of each classifier. As the authors know the concrete characteristics (IP, MAC, etc.) of the connections associated to malicious traffic, we were able to easily label the class of each and every packet as *bot* or *legitimate* to build the training datasets used in Section III.

Below, we go through the different supervised learning algorithms used to face the problem of detecting Command & Control traffic as a supervised classification problem. We have opted to compare the performance of the different classification algorithms given the notable differences in similar experiments depending on the approach used such as the detection of errors in software quality models [12] or the automatic classification of commentaries in social websites [13]. The tool used to perform these experiments is the Waikato Environment for Knowledge Analysis (WEKA)[1]. This software plataform written in Java was conceived to experiment with machine learning and data mining while remaining widely expandable with a variety of official and community-developed plugins.

In the case of this paper, the algorithms employed for this experiments are the ones that follow:

- **Support Vector Machines**. We have used the Sequential Minimal Optimisation (SMO) algorithm [14] employing for the experiments different kernels: a polynomial kernel [15], a normalized polynomial kernel [15], a Pearson VII kernel [16] and a Radial Basis Function (RBF) kernel [15].
- **Decision Trees**. We have selected *Random Forest* [17] and the implementation of the C4.5 [18] performed by the WEKA developers [19], J48.
- **Bayesian Networks**. With Bayesian Networks, we have used different algorithms of structural learning: K2 [20], *Hill Climbing* and *Tree Augmented Naïve* (TAN) [21], as well as testing the effectiveness of the *Naïve Bayes* algorithm [22].
- **Bagging**. We have used the implementation of the Bagging Method with the fast learning algorithm REPTree, a decision tree method used in the past, for example, to successfully evaluate the performance of individuals in online gaming platforms [23].
- **Perceptrons**. We have used different types of perceptrons that try to face the traditional problems of this technique to label classes which are not lineally separable: the time consuming Multi Layer Perceptrons (MLP) [24] and the Voted Perceptron [25].

[1] http://www.cs.waikato.ac.nz/ml/weka/

- **K-Nearest Neighbour (KNN)**. We have conducted experiments in the range from $k = 1$ to $k = 10$ neighbours. The goal is to check if the hypothesis already raised by the authors in the past regarding the inefficiency of the inclusion of more neighbours [26] could also be applied to the n-packet sequences.

The method employed to compare these algorithms is making use of a cross-validated series of experiments and balanced training datasets as detailed below.

1) Cross Validation: One of the ways of validating the behaviour of the classifiers consists of using cross validation techniques. These techniques divide the data collected into k training datasets compounded by the $[100 - (k-1) \cdot \frac{100}{k}]\%$ of the samples. Thus, after the training, the model would be validated with the rest of the $(\frac{100}{k})\%$ samples to evaluate its efficiency. The system error E can be defined as follows:

$$E = \frac{1}{k} \cdot \sum_{i=1}^{k} E_i \tag{5}$$

being E_i the error of each and every iteration. The k value used in WEKA for this research is $k = 10$, so that each dataset used is divided into ten training and testing datasets: the former would be compounded by the 90% of the representations whilst the latter, the ones used to analyse the capabilities of the model, by the remaining 10%.

2) Resampling and balance of the training data: As previously suggested, the number of packets of each session may vary noticeably depending on the data analysed, a point which has its importance if we take into account that it is much easier for us to generate legitimate traffic than to monitor botnet connections. However, the usage of unbalanced data may introduce skews in the classification producing over-fitting in certain cases [27]. Because of that, we have decided to use resampling techniques that readjust the number of packets of each instance as a previous step to the appliance of the different classification algorithms. In this case, the algorithm will opt to reduce the dataset to a number of samples per class equal to the number of the class with less samples in the initial dataset: that is to say, in the case of using sessions where the number of legitimate samples was greater than the number of malicious samples ($b >= m$), the algorithm will proceed to select only m benign samples to train the models. WEKA implements this technique with the Spread Subsample algorithm.

B. Comparison Metrics

The evaluation of each and every method is going to be performed according to different parameters commonly used in the evaluation of the performance of the classification methods [28]. As a first step, we have to take into account the significance of the confusion matrix shown in Table I. Thus, we can define *True Positives* TP as the number of botnet packets correctly identified, *False Positives* FP as the number of legitimate packets incorrectly classified as relative to a botnet, *True Negatives* TN as the number of legitimate packets correctly identified as legitimate and *False Negatives* FN as those packet generated by botnets that the system was unable to correctly identify. In this way, by the combination of the aforementioned data, we have defined the following

TABLE I. GENERIC CONFUSION MATRIX.

		prediction outcome		
		p	**n**	**total**
actual value	**p′**	True positive	False negative	P′
	n′	False positive	True negative	N′
	total	P	N	

comparison metrics so as to evaluate the performance of each classifier:

- ***Accuracy*** **(Acc.)**. The $Acc.$ —see equation 6— is worked out by dividing the total number of correct labels by the total number of instances that compose the full dataset.

$$Acc. = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

- ***Possitive Predictive Value*** **(**PPV**)**. The PPV —also known as *precision*— is a value that presents the possibility of finding a positive result representing the tested condition, *id est*, a sample labelled as a positive which effectively is a TP. It is mathematically defined in equation 7 as follows:

$$PPV = \frac{TP}{TP + FP} \tag{7}$$

- ***Area Under ROC Curve*** **(**$AURC$ **o** AUC**)**. The geometric meaning of the ROC curve derives from the establishment of a relationship amongst the false negatives and the false positives [12]. This value is obtained representing, for each possible election of cut values, the TPR in the y-axis and the FPR for the x-asis. Originally used by the American army in researches after the Pearl Harbor attack in 1941 so as to detect radar signatures of Japanese aircraft [29], this measure has been used sin the last part of the XX Century as a comparison metric to evaluate the performance of classification algorithms [30]. It is broadly used to generate statistics that represent the performance of a classifier in a binary system as a tool to select those probably optimal models from the suboptimal ones. Although this measure does not provide information about the good behaviour of a model, the $AURC$ helps to determine the validity of the data distribution given a series of predictory conditions [31].

C. Results

The benign dataset used in this experiment corresponds to traffic obtained from different sessions run by the students of the *Máster Universitario de Seguridad de la Información*

—in English, Master of Information Security— held by the University of Deusto. Thus, the authors could proceed to the monitoring of the connections during the following browsing hours by using a sniffing tool to monitor the traffic. As an additional remark, and because of the legal implications that would imply the fact of storing such kind of traffic data, it was necessary to obtain the students agreement to participate in the experiment being held. This restriction has some implications experimentally speaking: it may introduce a sort of skew in the data analysed regarding the final content of the communications as a user having being advised of being monitored for an experiment is substantially less inclined to visit, for instance, banking sites or webpages with pornographic content. However, the authors consider this naturally introduced error as acceptable since the students were told to perform standard browsing sessions following the patterns stated by a recent study which tried to provide data on the average usage of the Internet [32].

Meanwhile, malware traffic samples were obtained after infecting different machines with samples of Flu, Prablinha and Warbot. All of them are controlled from a web-based Command & Control panel from which the botmaster can execute the different attacks. For the purpouse of this research, all along the infected sessions we performed different attacks requesting the infected node to execute different tasks: performing DDoS attacks, downloading and executing a file, transferring files, storing the keystrokes from the user, etc. The number of Command & Control packets (the only ones analysed in this paper) stored ranged from 1,000 to 5,000 in each session.

With this information, we built 12 datasets for each piece of malware (a total of 36 different datasets) generating representation samples of $n = \{1, 2, 3, \cdots, 12\}$ packets. Each dataset was trained with each and every of the 23 previously defined classification algorithms to complete the 828 experiments conducted whose results are explained below. Note that the results of each individual family have been grouped into a single graph by calculating the arithmetic mean of the values separately obtained.

In Figure 1, we can see the evolution of the percentage of correct classifications performed by every algorithm for each proposed representation. Generally, we can see an important improvement of the classifications performed for those representations taking into account longer sequences. This improvement, however, is not linear. For all the algorithms, even for those with worse behaviour such as SMO RBF, Naïve Bayes and Voted Perceptron, we can observe a very important jump from sequences of single packets to those using $n = 2$ and $n = 3$ packets, after what the performance of the classifier starts to stabilise its improvements. As occurred in similar research performed in the past, the authors have noticed that taking into account more neighbours in the KNN classifier does not improve the classification. On the contrary, we have observed that the more neighbours are considered, the worse the results obtained. We can affirm that the best results have been obtained by Bayes Net K2 classifier for a sequence length of $n = 12$ (being the best example of a bett), while we can consider the second best algorithm Random Forest for sequence lengths of $n = 6$ and $n = 7$. Random Forest shows signs of stagnation for sequences longer than $n = 8$. As a negative remark, we have to pinpoint the little efficiency of some classifiers in comparison to the rest of the algorithms tested. *Voted Perceptron* and *Naïve Bayes* achieved the worst results for all lengths.

The use of the PPV puts in context the results shown in Figure 1. The PPV graphs (see Figure 2) show the relation between the true positives and the false positives encountered. This measure is important taking into account that a high detection rate would be useless if the False Positive Ratio is also high. If this was the case and a system was implemented in a real environment, users could feel frustrated and the detection efforts would have been in vain.

In this research, we have found out that 12 out of the 23 classifiers have obtained values over 0.95 at least once. In this case, the best overall results were obtained by SMO using the Pearson VII kernel, achieving almost the maximum score for lengths of $n = 10$ and $n = 11$. We have to highlight here the Bayesian classifiers (excepting Naïve Bayes), Bagging and Random Forest, although this last one shows again symptoms of lightly losing efficiency for the longest lengths of packets. This indicator confirms what we have exposed before about the consideration of more neighbours on KNN, while Multi Layer Perceptron does not obtain significant results taking into account the extra time needed to perform the experiments for such classifier. Once again, the worst classifiers are Naïve Bayes, SMO with RBF kernel (though they could cut the gap towards the rest of the classifiers, specially for the longest sequences) and Voted Perceptron.

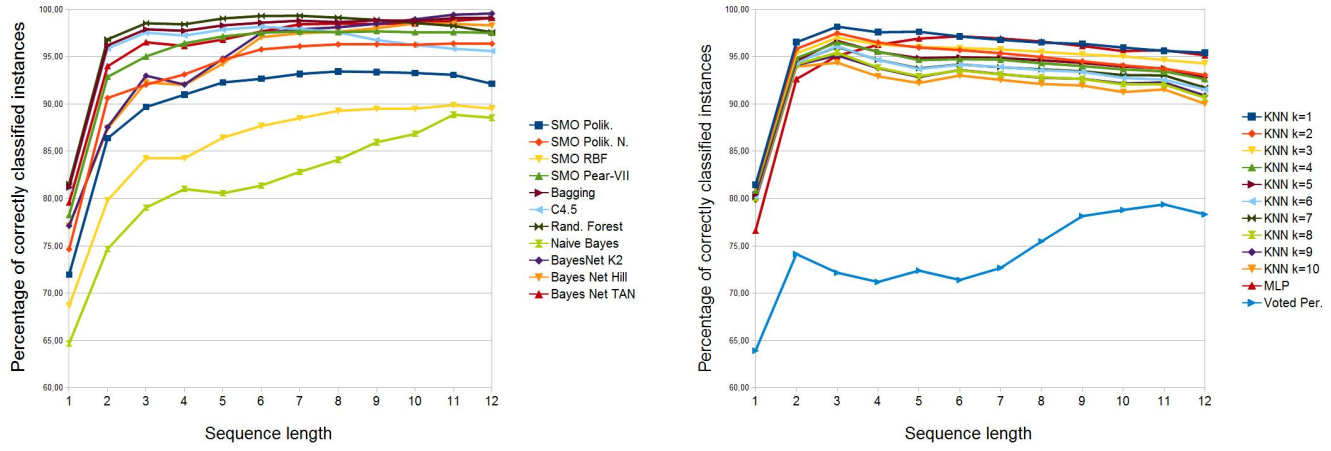
Finally, in Figure 3 we show the evolution of the values under the Roc Curve ARC for the detection of Command & Control traffic depending on the number of packets included in the representation. These values represent the kindness of the dataset used while we increase the number of packets included per sequence.

The results are especially positive in the case of Bayes Net (with K2 and TAN search methods), Random Forests and Bagging. At the same time, most KNN versions of the algorithm obtain very high values. With regard to this group of algorithms, we find interesting to pinpoint an inversion of the trends observed in the aforementioned indicators showing that KNN with $k = 1$ is the worst classifier amongst them. The worst overall results are obtained again by Voted Perceptron, Naïve Bayes and SMO RBF, which are unable to cope with the problem in an efficient way.

IV. Discussion and future work

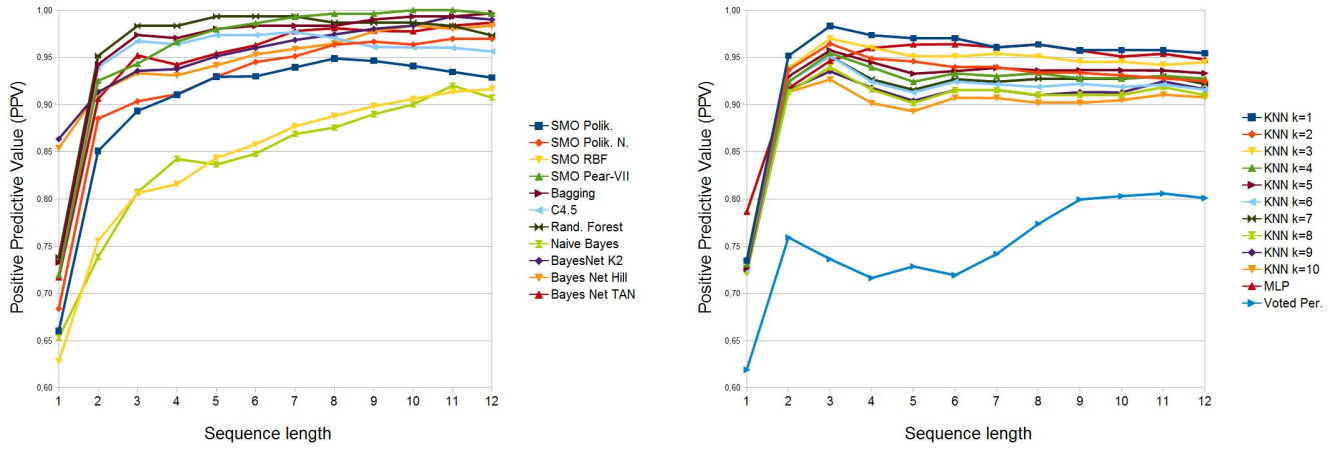
One of the main problems supervised learning brings into the field of traffic classification is the immense amount of data that has to be processed. At the same time, researchers will have to deal with some important legal difficulties associated to the extraction of traffic samples which could serve to avoid any kind of biases in the training datasets. We are applying different algorithms assuming the additional efforts required in the usage of supervised approaches [33] as a first step towards the application of other philosophies. In this line, the legal —and logical— need of having the consent of the users that will donate their user sessions, forces us to assume certain risks associated to the inexistence of browsing habits that in real environments would be considered as legitimate ones —such

 ISBN: 978-1-61208-311-7



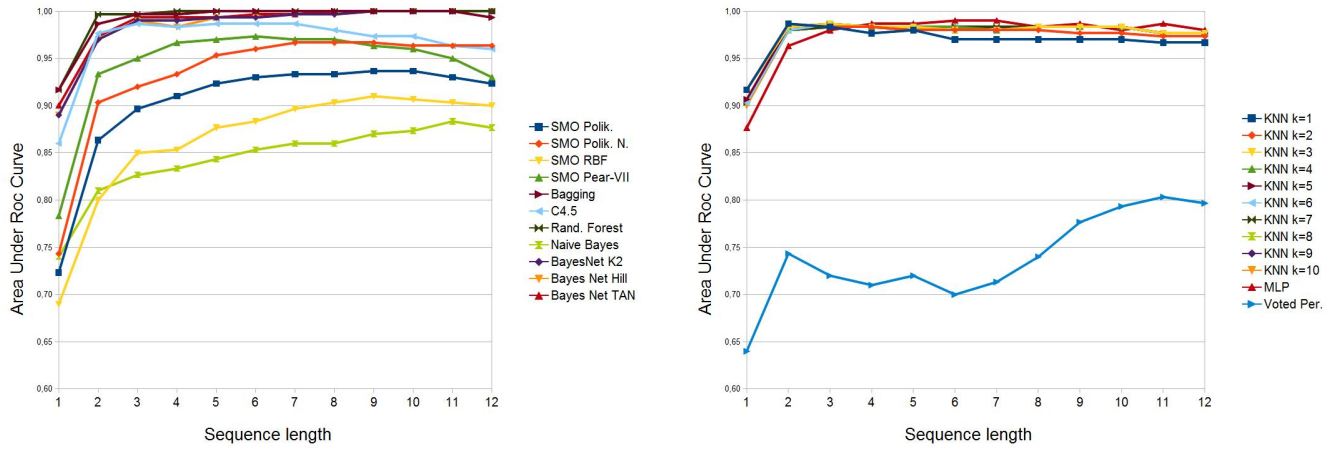
(a) (b)

Fig. 1. Evolution of the $Acc.$ while using different sequence lengths as the representations used by the classification algorithms to detect C&C traffic.



(a) (b)

Fig. 2. Evolution of the PPV while using different sequence lengths as the representations used by the classification algorithms to detect C&C traffic.



(a) (b)

Fig. 3. Evolution of the ARC while using different sequence lengths as the representations used by the classification algorithms to detect C&C traffic.

 ISBN: 978-1-61208-311-7

as the visit to websites with adult content—. However, the appearance in testing environments is little or none because of the fact that the user knows that he is surfing through a network that is being monitored.

The virtue of the method defined in this paper resides in the fact of classifying traffic independently of the content that flows through the network, using as detection metrics the values relative to the length of the headers, the connection frequency or the periodicity of the polls between the infected machine and the machine from where the botmaster issues the orders.

One of the most important issues is the fact that the malicious charge of a packet (or even of a short sequence of packets) is relatively low. Accordingly, the appearance of True Negatives is less harmful than in other fields such as malware detection, where being unable to detect a malicious instance could have dramatic consequences. As the number of packets generated by a single connection is, normally, pretty big, we can think of a detection method which could employ the historical records of the representations connected to a single point. By doing this, we would be able to assign a value that, at any time, could let the system know the reliability of the current connection taking into account how the packets with certain source or destiny have been classified. The fact of being able to tolerate certain True Negatives would provide resilience to a system which would be able to inform the user the establishment of suspicious connections even without having a traditional malware detection solution on its computer.

V. CONCLUSIONS

Botnets are going to remain as one of the most important cyberthreats in the near future. In this regard, we have proposed in this article a methodology to validate a new representation model that could be used to identify the Command & Control communications performed by a botnet. We have proved that the attributes obtained from the observation of the packets in a connection could be used to fit the initial objectives. We have defined a series of transformations to perform to the observed packets to build vectorial representations sensible to both, atomic attributes of the packets and spatio-temporal variables. To face this issue, we have considered the problem as a binary classification problem in which the algorithms employed would have to be able to differentiate benign samples from malicious ones.

The comparison metrics employed have guided us to compare the performance of the different representations employed and the different algorithms. We can state that the inclusion of more packets in our representations improve the results obtained by a given algorithm for shorter representations. At the same time, we have proved that some algorithms, such as Bayes Net and Random Forest obtained very high detection ratios. Anyway, the characteristics of the packet classification problem has some advantages that could be exploited in the favour of the security analyst: the payload of a single packet is pretty less relevant to the payload of a malicious binary, so a greater False Negative ratios would be tolerated with little or no harm for the final user, letting the system take less borderline decisions by waiting for more pieces information.

The increasing amount of data to be analysed in current corporate networks will undoubtedly force us to face congestion problems. Thus, future work should lead us to the management of fewer amounts of data, making the inclusion of an unsupervised approach which would need less labelled instances and the consideration of more botnet families should be studied to give more robustness to a methodology whose implementation could complement the traditional honeypot solutions and content-based detection techniques.

REFERENCES

[1] S. Blank, "Web war i: Is europe's first information war a new kind of war?" *Comparative Strategy (2008)*, vol. 27, no. 3, p. 227, 2008. [Online]. Available: http://www.informaworld.com/10.1080/01495930802185312

[2] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@ home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[3] Z. Li, Q. Liao, and A. Striegel, "Botnet economics: uncertainty matters," in *Managing Information Risk and the Economics of Security*. Springer, 2009, pp. 245–267.

[4] N. Cornhill and M. Morris, "Communication structures of botnets with case studies," 2012.

[5] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 129–138.

[6] F. Brezo, J. Gaviria de la Puerta, X. Ugarte-Pedrero, I. Santos, P. G. Bringas, and D. Barroso, "Supervised classification of packets coming from a http botnet," in *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*. IEEE, 2012, pp. 1–8.

[7] B. Carlson, "Algorithms involving arithmetic and geometric means," *The American Mathematical Monthly*, vol. 78, no. 5, pp. 496–505, 1971.

[8] J. M. Bland and D. G. Altman, "Statistics notes: measurement error," *Bmj*, vol. 312, no. 7047, p. 1654, 1996.

[9] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*. Citeseer, 2008.

[10] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *In 2nd IEEE LCN Workshop on Network Security (WoNS2006*, 2006, pp. 967–974.

[11] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," in *Proceeding of the Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, 2007, pp. 3–24.

[12] Y. Singh, A. Kaur, and R. Malhotra, "Comparative analysis of regression and machine learning methods for predicting fault proneness models," *International Journal of Computer Applications in Technology*, vol. 35, no. 2, pp. 183–193, 2009.

[13] I. Santos, J. de-la Peña-Sordo, I. Pastor-López, P. Galán-García, and P. G. Bringas, "Automatic categorisation of comments in social news websites," *Expert Systems with Applications*, 2012.

[14] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," *Advances in Kernel Methods-Support Vector Learning*, vol. 208, 1999.

[15] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[16] B. Üstün, W. Melssen, and L. Buydens, "Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel," *Chemometrics and Intelligent Laboratory Systems*, vol. 81, no. 1, pp. 29–40, 2006.

[17] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] J. R. Quinlan, *C4. 5 programs for machine learning*. Morgan Kaufmann Publishers, 1993.

 ISBN: 978-1-61208-311-7

[19] S. R. Garner, "Weka: The Waikato environment for knowledge analysis," in *Proceedings of the New Zealand Computer Science Research Students Conference*, 1995, pp. 57–64.

[20] G. F. Cooper and E. Herskovits, "A bayesian method for constructing bayesian belief networks from databases," in *Proceedings of the* 7^{th} *conference on Uncertainty in artificial intelligence*, 1991.

[21] D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, "Bayesian network classifiers," in *Machine Learning*, 1997, pp. 131–163.

[22] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," *Lecture Notes in Computer Science*, vol. 1398, pp. 4–18, 1998.

[23] K. J. Shim, K.-W. Hsu, and J. Srivastava, "Modeling player performance in massively multiplayer online role-playing games: The effects of diversity in mentoring network," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 2011, pp. 438–442.

[24] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)–a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[25] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[26] F. Brezo, J. Gaviria de la Puerta, X. Ugarte-Pedrero, I. Santos, P. G. Bringas, and D. Barroso, "Supervised classification of packets coming from a http botnet," in *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*. IEEE, 2012, pp. 1–8.

[27] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," 2012.

[28] D. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation," *School of Informatics and Engineering, Flinders University, Adelaide, Australia, Tech. Rep. SIE-07-001*, 2007.

[29] D. Green, J. Swets *et al.*, *Signal detection theory and psychophysics*. Wiley New York, 1966, vol. 1974.

[30] K. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proceedings of the sixth international workshop on Machine learning*. Morgan Kaufmann Publishers Inc., 1989, pp. 160–163.

[31] J. Lobo, A. Jiménez-Valverde, and R. Real, "Auc: a misleading measure of the performance of predictive distribution models," *Global Ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2007.

[32] F. Lera-Lpez, M. Billon, and M. Gil, "Determinants of internet use in spain," *Economics of Innovation and New Technology*, vol. 20, no. 2, pp. 127–152, 2011. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10438590903378017

[33] A. Teichman and S. Thrun, "Tracking-based semi-supervised learning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 804–818, 2012.

 ISBN: 978-1-61208-311-7