CONDENSER: A Graph-Based Approach for Detecting Botnets

Pedro Camelo[12], João Moura[12*], and Ludwig Krippahl[2]

[1] AnubisNetworks R&D, Amadora, Portugal
{pedro.camelo,joao.moura}@anubisnetworks.com
[2] CENTRIA, Universidade NOVA de Lisboa, Portugal
pmcamelo@gmail.com joaomoura@yahoo.com ludi@fct.unl.pt

Abstract. Botnets represent a global problem and are responsible for causing large financial and operational damage to their victims. They are implemented with evasion in mind, and aim at hiding their architecture and authors, making them difficult to detect in general. These kinds of networks are mainly used for identity theft, virtual extortion, spam campaigns and malware dissemination. Botnets have a great potential in warfare and terrorist activities, making it of utmost importance to take action against.

We present CONDENSER, a method for identifying data generated by botnet activity. We start by selecting appropriate the features from several data feeds, namely DNS non-existent domain responses and live communication packages directed to command and control servers that we previously sinkholed. By using machine learning algorithms and a graph based representation of data, then allows one to identify botnet activity, helps identifying anomalous traffic, quickly detect new botnets and improve activities of tracking known botnets.

Our main contributions are threefold: first, the use of a machine learning classifier for classifying domain names as being generated by domain generation algorithms (DGA); second, a clustering algorithm using the set of selected features that groups network communication with similar patterns; third, a graph based knowledge representation framework where we store processed data, allowing us to perform queries.

1 Introduction

Currently there is an large number of infections caused by malware present in all sorts of electronic devices. It is estimated that about 16% to 25% [4] of Internet traffic in the world comes from communication between various types of malware. Moreover, because of the pervasive nature of devices such as smartphones and tablets which are increasingly replacing mobile phones and computers [1], the number of infections is increasing and becoming more complex. More computational power, more tools as well as more opportunities and targets are available to malware develop, as they start exploring broader features within each device.

Botnets are generally deployed for illegal purposes and are used for identity theft, massive spamming, distributed denial of service (DDoS), government and industrial espionage and unauthorized use of computational resources e.g. for bitcoin mining. Statistics estimate that roughly 75% [14] of all global email traffic is coming from spam, and that most of this traffic originates from botnets. Recent develops show us that such networks also have large potential in scenarios of warfare and terrorism, which is another reason to put an effort into stoping the authors that produce malware, and consequently to botnets. There are various governmental authorities, companies and organizations that struggle daily against these cybercriminals.

Identifying botnets can be done by malware analysis in controlled environments (e.g. sandboxing or virtualization), or through analysis of malware communication with the Command and Control (C&C) servers and other infected devices. The two main ways to detect botnets can be broadly divided into: Passive and Active. Passive detection is based on communication analysis of packets between the C&C and infected devices, and has the advantage of providing no warning of the impending discovery of botnet infrastructure. Active sensing involves, in addition, a direct interaction with the botnet through the injection of network packets (e.g. dns query to the domain nameserver). Botnets often implement evasion techniques, making it difficult to take action against. Some examples are, encrypted communication [15], domain generation algorithms [11] (DGA), fast-flux and double fast-flux [7][3].

1.1 Objectives

Monitoring network traffic generates large volumes of data that must be processed in order to obtain useful information. In addition, botnet evasion techniques and the diversity of services that generate network traffic, which is usually encrypted, make it necessary to examine patterns on sets of features in order to detect botnet activity. The problem is, thus, one of inferring useful information from a large flux of complex data. In this paper we present a solution to this problem, using a combination of expert knowledge preprocessing followed by classification and clustering with machine learning algorithms.

This document is divided into four sections. In Section 1, we introduced the motivation, the context and gave an introduction to the topic of botnets. In Section 2 we discuss related work and overview existing research within the topic as well as current solutions to the problem described and present and overview of relevant machine learning techniques. In Section 3 we present and evaluate the machine learning techniques we employed and present also a novel method for representing botnet communications as well as to characterise these botnets in terms of singly connected components of domains and IPs. We finish with conclusions and a discussion of future work in Section 4.

---

[3] Fast-flux is a DNS technique used by botnets in order to hide not only phishing activities but also sites delivering malware behind networks of compromised hosts that change very often and that act as proxies.

2 Related Work

There is growing interest in botnet research, both in the industry and in academia, motivated by the need to counter the criminal activity associated with botnets. In addition to general security considerations, a major issue in botnet research is the development of automated detection methods.

This research focused both on active and passive detection. For active detection we used DNS records analysis. For passive detection we identified three sub-categories: domain names analysis, flow analysis and packet analysis. For the purpose of this work we did not consider flow analysis since we were positioned in the middle of devices communication and NAT and IP pooling makes it harder to detect relevant IP changes.

The literature reports a diversity of approaches to traffic analysis, and thus, illicit activity detection. For example, Amini, et al. [2], used Neural Networks to detect network intrusions; Ruehrup, et al. [10] used common destinations as a way to compute connection sub-graphs to identify malicious P2P networks; Stalmans, et al. [13] showed that geographic information of DNS records helped identify suspicious traffic using Morgan Indexes and the Geary Coefficient. Antonakakis, et al. [3] studied the behavior of some malware that use Domain Generation Algorithms to reach their C&Cs, and Davuth, et al. [6] showed that support vector machines could identify generated domain names with better performance than other classifiers, such as *Naïve Bayes* and *C5.0* classification trees. Schiavoni, et al. [11] concluded that domain name analysis alone is not sufficient to detect botnets, so they included IP and DNS analysis as a similarity feature to group same DGA domains. This diversity of methods, some using expert knowledge and others machine learning, inspired our combined approach.

We present next an overview of the machine learning techniques we use in our work.

2.1 Classification: Support Vector Machines

Machine learning is a "field of study that gives computers the ability to learn without being explicitly programmed", as defined by Arthur Samuel in 1959. In the field of cyber security and threat intelligence we highlight applications in the areas of malware detection, malicious traffic detection and spam detection.

In machine learning classification, support vector machines (SVM) are models created with supervised learning, associated with learning algorithms that analyze data and recognize patterns. They are mostly used for classification and regression analysis. In the case of binary SVMs, given a set of training examples each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the categories are divided as best as possible. Examples for classification are then mapped into that same space and predicted to belong to a category based on which side of the division they fall on.

In general, an SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensionality space. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called

functional margin). It is thus the case that, the larger the margin the lower the generalization error of the classifier.

2.2 Clustering: Self-Organizing Maps

A self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning, with the purpose of reducing the dimensionality of features into a low-dimensional representation of the input space. These self-organizing maps differ from traditional artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.

2.3 Cluster Interpretation and Validation: Indexes

The Davies-Bouldin [5] index – represented as DB_k – identifies *clusters* that are compact and distant from each other, in Equation 1 the diameter for *cluster* c_i is obtained, where n_i represents the number of points belonging to *cluster* c_i. Symbol z_i corresponds to the centroid of *cluster* c_i and x is a point belonging to *cluster* c_i.

$$DB_k = \frac{1}{k} \sum_{i=1}^{k} Max_{j=1,...,k,i \neq j} \left\{ \frac{diam(c_i) + diam(c_j)}{eucl(c_i - c_j)} \right\} \tag{1}$$

In DB_k, k corresponds to total number of existing *clusters*, c_i and c_j correspond to to the centroid of *clusters* i and j respectively.

$$diss(c_i, c_j) = min_{x \in c_i, y \in c_j} ||x - y|| \tag{2}$$

Equation 2 corresponds to the dissimilarity index between clusters c_i and c_j, Equation 3 is the diameter of the cluster C.

$$diam(C) = max_{x,y \in C} ||x - y|| \tag{3}$$

The Silhouette [8] index – represented as SI_k – identifies the average membership of each point to all k clusters, where n is the total number of existing samples in the dataset, a_i is the average distance between point i and all points of its cluster; b_i is the minimum average dissamilarity between point i and all the formed clusters. In this metric, the partition with bigger SI value is considered optimal.

$$SI_k = \frac{1}{n} \sum_{i=1}^{n} \frac{(b_i - a_i)}{max(a_i, b_i)} \tag{4}$$

3 Contributions

Currently implemented modules include one Support Vector Machine (SVM) classifier for detecting suspicious domain names (generated by DGA); a Self-Organizing Map for data clustering; one from two IP reputation services; one from Maltracker[4], a malware

---

[4] https://maltracker.net/

analysis platform that extracts information and network behavior of malware samples ran in sandboxed environments; one from an AnubisNetworks proprietary API informing whether an IP address had contacted one of AnubisNetworks' sinkholes; and finally we developed a module that crawls over DNS servers gathering historic information about domain names and IPs, with the goal of detecting common name servers and IP addresses used as C&Cs, botnet domain rotation and botnets that operate using fast-flux and double fast-flux schemes.

For the DGA classifier and as benign dataset, we used the Top 10,000 domains from Alexa, and as anomalous dataset we used a dataset of approximately 10,000 DGA domains provided by AnubisNetworks. We also automated the generation of such classifier modules by implementing a system for feature selection, training and evaluation of classifiers, making it simple not only to adapt the current DGA classifier to changes in the data stream, but also to include other classifiers in the future. The Self-Organizing Map groups data with similar patterns discarding the source and destination of the data communication since we want to find similar traffic that can have no intersection in its communications at all (e.g. same botnet family, different campaign and operators).

3.1 Graph Based Detection

To improve botnet detection, we created a modular application that extracts and aggregates packet data, by combining expert knowledge and machine learning classification and clustering. This system is composed by a set of modules that obtain information from external services and incoming packet data. The modular implementation makes it easier to adapt the system to different data streams, and the combination of expert knowledge rules and machine learning takes advantage, of both, known indicators of malicious activity and the capacity of machine learning algorithms to identify patterns in complex data. The output is a connected graph of the observed communications, including the data obtained from the modules, where nodes are IPs, domains, SOM Clusters representing communication with similar patterns and objects such as links to executable files and/or their signatures that were seen communicating with common destinations/IPs. With this system, we correlate information gathered from many sources, to detect suspicious sub-graph topologies and relations that allows us to characterise botnets in terms of singly connected components of domains and IPs.

3.2 Discussion

The presented work aims to identify all types of known botnet topologies (star, distributed, hybrid), by correlating information from various sources. We based our work in the existing related work, and we are able to identify common evasion techniques such as, domain names generated by domain generation algorithms (DGA), with the corresponding machine learning classifier. The precision of this classifier indicate a result of 77,9% when ran against a set of domains generated by malware without a pre-defined dictionary in its implementation (e.g. english names).

With our DNS crawler we gather information about normal and suspect domains, normally associated with low TTLs and high record changing (associated with fast-flux and double fast-flux). However there are some legitimate uses that can be miss-

interpreted as these evasion methods. For example, Content Delivery Networks (CDNs) rely on low TTL records in their domains. This behavior can raise false positives if we only take the TTL value in consideration. For this reason we must take other information when we decide that we are over a new botnet family. Since our work relies in network communication, and taking in consideration malware that use the network as a way to communicate with the C&C[5], we can gather information about all the malware lifecycle.

The clustering algorithm provides a way to detect communication with similar patterns of known clusters even if the packet source and destination do not intersect, this characteristic provide a way to detect same malware families. When we consider the full system we have a way to detect not only the typical communication cluster, with enriched data, but also communication that matches already known communication patterns. With the information obtained from our sinkholes we are able to group communication that matches known botnet families.

Traffic that resulted in non-existent domain (NX-Domain) replies can not be compared with destinations that were "alive" when we take DNS and Geographic IP information in consideration, for this reason we didn't include features that match these information in our machine learning algorithms.

3.3 Evaluation

For the domain name classifier, we obtained $77,9\%$ of precision, using a set of 11 features. For the clustering algorithm we obtained a rating of 9.32 with the Davies-Bouldin Index and of aproximately 0.27 for the Silhouette Index with a total of 81 features.

During our research we were not able to compare our results with work available in the literature because existing work only takes in consideration traffic inside universities campus, or inside small networks sometimes even with artificially generated traffic which is also referred in [12], and since we are positioned between internet operators, we are unable to extract some of the features proposed in related work.

4 Conclusions and Future Work

We developed a system that aims to correlate information from various sources, including an automatic classifier taking domain names as inputs and deciding whether they are part of a DGA or not, information about live samples and sinkholed domains, and a clustering tool using Self-Organizing Maps to group network traffic having the same patterns. For checking if a domain name was generated by a DGA, our classifier has $77,9\%$ of precision, for the clustering algorithm used we obtained a rating of approximately 9.32 using the Davies-Bouldin Index. As an auxiliary mechanism we developed a system that crawls over DNS servers, gathering historic DNS information which allows us to discover both fast-flux and double fast-flux domains as well as rotation of

---

[5] Contrary to this, computer worms sometimes do not rely in any type of communication from a C&C, or can became dormant for a long period of time relying in a physical way of spreading, like an usb stick. For these cases, we are not able to get data.

C&C servers. We store all this data in a graph that allows us to single out botnets as singly connected components of malicious servers (domains) and infected machines (IPs).

As future work, we will proceed with our research by using continuous training algorithms for the clustering algorithms, such as implementing a growing and hierarchical extension of self-organizing maps, termed Growing Hierarchical Self-Organizing Maps [9] (GHSOM).

References

1. Smart insights. mobile marketing statistics. `http://www.smartinsights.com/mobile-marketing/mobile-marketinganalytics/mobile-marketing-statistics/`, 2013.
2. Morteza Amini, Rasool Jalili, and Hamid Reza Shahriari. RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, 25(6):459–468, September 2006.
3. Manos Antonakakis, Jeremy Demar, Christopher Elisan, and John Jerrim. DGAs and Cyber-Criminals: A Case Study. pages 1–9, 2012.
4. Basil AsSadhan, José M. F. Moura, David Lapsley, Christine Jones, and W. Timothy Strayer. Detecting botnets using command and control traffic. In *Proceedings of the 2009 Eighth IEEE International Symposium on Network Computing and Applications*, pages 156–162, Washington, DC, USA, 2009. IEEE Computer Society.
5. David L. Davies and Donald W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979.
6. Nhauo Davuth and SR Kim. Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method. *sersc.org*, 7(1):51–58, 2013.
7. G Hogben, D Plohmann, E Gerhards-Padilla, and F Leder. Botnets: Detection, measurement, disinfection and defence. 2011.
8. P. J. Rousseeuw L. Kaufman. *Finding Groups in Data: An Introduction to Cluster Analysis.* 1990.
9. A Rauber, D Merkl, and M Dittenbach. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 13(6):1331–41, January 2002.
10. Stefan Ruehrup and Pierfrancesco Urbano. Botnet detection revisited: theory and practice of finding malicious P2P networks via Internet connection graphs. *IEEE Conference on*, (Tma):435–440, 2013.
11. Stefano Schiavoni, F Maggi, L Cavallaro, and Stefano Zanero. Tracking and Characterizing Botnets Using Automatically Generated Domains. 2013.
12. Robin Sommer and Vern Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316. IEEE, 2010.
13. Etienne Stalmans and Barry Irwin. Spatial Statistics as a Metric for Detecting Botnet C2 Servers. *botconf.eu*.
14. Trustwave. 2013 Global Security Report. Technical report.
15. David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali Ghorbani, and Dan Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39:2–16, November 2013.

A Considered Features

Table 1. Domain Name Features

Feature	Type	Feature Description
consonantRatio	Numeric	Consonant Ratio
consonantVowelRatio	Numeric	Consonant Vowel Ratio
domainLength	Numeric	Domain Length normalized as RFC 3986 (255 characters)
othersRatio	Numeric	Other characters ratio
vocalRatio	Numeric	Vowel Ratio
digitRatio	Numeric	Digit Ratio
numRepeatsByUniGram	Numeric	Number of string repetitions by Uni-gram analysis
numRepeatsByBiGram	Numeric	Number of string repetitions by Bi-gram analysis
numRepeatsByTriGram	Numeric	Number of string repetitions by Tri-gram analysis
numRepeatsByTetraGram	Numeric	Number of string repetitions by Tri-gram analysis
lowFrequenceOccurrence	Categoric	Domains starting and ending with a digit

Table 2. URI Features

Feature	Type	Feature Description
queryLength	Numeric	URI Query Length
queryArgumentSize	Numeric	Number of URI Query Arguments
uriPathLength	Numeric	URI Path Length
uriPathLevelLength	Numeric	URI Path Level Length
uriPathPlusLength	Numeric	URI Path + Following Components Length
uriExistence	Categoric	URI Path + Following Components Existence
exe, bat, cmd, msi, com, drv, js, css, dat, ppt, doc, docx, txt, rtf, php, cgi, asp, aspx, html, xhtml, jsf, dll, png, jpg, bmp, bin, dll, zip, rar, swf, scr, wpad, pac, ini	Categoric	URI Extension
unknownExtension	Categoric	Unknown Extension (not in the feature list above)
unavailableExtension	Categoric	Extension not available
consonantRatio	Numeric	URI Base Consonant Ratio
vocalRatio	Numeric	URI Base Vowel Ratio
consonantVowelRatio	Numeric	URI Base Consonant Vowel Ratio
extensionLength	Numeric	URI Extension Length

Table 3. Meta Features

Feature	Type	Feature Description
packetSize	Numeric	TCP/IP Packet Size
packetSizeInexistence	Categoric	Unknown TCP/IP Packet Size
200, 301, 400, 404, 413	Categoric	Código de Resposta HTTP
unknownReplyCode	Categoric	Unknown HTTP Reply Code (not present in the above feature list)
inexistentHttpRCode	Categoric	HTTP Reply Code not available
HTTP/1.0, HTTP/1.1	Categoric	HTTP Version
unknownHttpVersion	Categoric	Unknown HTTP Version
inexistentHttpVersion	Categoric	HTTP Version not available