Anomaly-Based Detection of IRC Botnets by Means of One-Class Support Vector Classifiers*

Claudio Mazzariello and Carlo Sansone

University of Napoli Federico II
Dipartimento di Informatica e Sistemistica
via Claudio 21, 80125 Napoli (Italy)
{claudio.mazzariello,carlo.sansone}@unina.it

Abstract. The complexity of modern cyber attacks urges for the definition of detection and classification techniques more sophisticated than those based on the well known *signature detection* approach. As a matter of fact, attackers try to deploy armies of controlled *bots* by infecting vulnerable hosts. Such bots are characterized by complex executable command sets, and take part in cooperative and coordinated attacks. Therefore, an effective detection technique should rely on a suitable model of both the envisaged networking scenario and the attacks targeting it.

We will address the problem of detecting *botnets*, by describing a behavioral model, for a specific class of network users, and a set of features that can be used in order to identify *botnet*-related activities. Tests performed by using an anomaly-based detection scheme on a set of real network traffic traces confirmed the effectiveness of the proposed approach.

1 Introduction

Computer networks and networked hosts have always been targeted by cyber-attacks. The nature of such attacks has evolved alongside with technology and protocol improvements: modern attacks exploit stealthy techniques for breaking into systems and controlling or disrupting their resources. In fact, the aim of malicious users is often to collect a large number of infected hosts in order to make them cooperate toward a common malicious objective. Furthermore, attackers' motivations have changed over time. While in the past system vulnerabilities were exploited more for the sake of research curiosity and bug exposure, nowadays huge amounts of money are often driving the trends in attack perpetration. The potential damage caused by a well crafted Denial of Service attack on a competitor's activities, the valuable personal information harvested by going out *phishing* ingenuous and unexperienced users, or the significant saving in marketing investment obtained by spamming millions of users at a *ridiculous*

* This work has been partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project).

P. Foggia, C. Sansone, and M. Vento (Eds.): ICIAP 2009, LNCS 5716, pp. 883–892, 2009.

cost are some of the reasons pushing dishonest investors to spend money in this sort of underground competition for customers and resources.

In this framework *botnets* have recently gained increasing interest by the scientific community, the industry, and the media. Like worms, they are able to spread over thousands of host at a very high rate, infecting them using a wide set of known vulnerabilities. Unlike worms, botnets reach a further step in putting network security at danger due to their ability to coordinate themselves, and to cooperate towards a common malicious objective. For that reason, botnets are nowadays one of the preferred means to spread spam all over the Internet [1].

A *Botnet* can be in fact regarded as a distributed platform for performing malicious actions. Botnets are made by *zombie* hosts, named *bots* in this context, which are controlled by the attacker, also known as the *botmaster*, by means of a **C**ommand & **C**ontrol (C&C) channel used to issue commands to, and eventually get responses back from the bots. Bots are usually common hosts, usually not very well protected, infected by means of several techniques.

Many works try to propose a botnet taxonomy [2,3], built by taking into account properties such as the propagation mechanism, the vulnerability exploitation strategy, the type of command and control channel used, or the set of commands available to the botmaster. Dagon et al., in [4] analyze time patterns characterizing botnet activity. The type of channel most often addressed so far is based on the IRC protocol. That is because originally, bots were benign programs used by IRC channel operators to monitor and manage their own channels automatically. In [5] the authors present some metrics based on flow analysis aimed at detecting botnets. After filtering out IRC session from the rest of the traffic, they apply flow based methods in order to discriminate between benign and malicious IRC channels. In [6] and [7] methods are proposed, which combine both application and network layer analysis. In [8] the authors propose to correlate information about IRC channel activity at the application layer, with information coming from the monitoring of network activity. Some authors also address the problem of botnet detection by using machine learning techniques [9]. The intuition behind this idea is that machine learning can provide a good means of characterizing botnets, once a good set of representative features has been selected. In [10] Rajab et al. propose a multifaceted approach, based on distributed monitoring, to detect botnets from the network behavior point of view. Others propose monitoring of services such as DNS for blacklisting botnet related domain [11].

In this paper a proposal for a system analyzing IRC traffic for botnet detection is presented. Differently from other approaches [5], where the analysis is performed at the network layer, our approach addresses the problem of application layer behavior modeling. Our system sniffs network traffic, extracts a behavior model of IRC traffic, and uses pattern recognition techniques for detecting the expected differences between a normal and a malicious user. Similarly to other network security approaches [12], we propose an anomaly-based detection schema, since "clean" IRC channels are easily available for training, while botnet-infected ones could be very difficult to obtain.

The paper is structured as follows: in Section 2 some background and technical information are given about the botnet phenomenon; the IRC traffic model and the features proposed for distinguishing between normal and botnet-related activities are described in Section 3; in Section 4 some experimental results show the detection performance of the proposed botnet detection system; finally, in Section 5 some conclusive remarks are given.

2 Understanding the Botnet Phenomenon

In the botnet model we decided to address, any infected host scans a defined subset of the whole IP space looking for known vulnerabilities. Once a vulnerable host is found, the vulnerability is exploited, and the host gets eventually infected. Once infected, the new bot contacts a server where it downloads the bot binary. The first examples of bots used to have the botmaster and command and control IP hardcoded, therefore once either of those was discovered, the botnet was made completely useless. For both flexibility and resiliency matters, though, bots started using symbolic names for contacting both the command and control channel and the botmaster. That is why a new bot, once infected, will issue a number of DNS queries to resolve the symbolic names associated to the entities it has to contact. By describing the finite state automaton describing the typical botnet behavior, specific stages of a bot's and botnet's lifecycle are targeted, in order to perform the detection and disruption operations.

Experiments show that known bots are characterized by a propagation profile similar to that of popular Internet worms. That is because many bots inherited their own propagation strategies from some popular worms of the past, which proved very quick and effective [10]. Under this assumption, it can be assumed that the number of infected hosts, and hence of DNS requests issued, varies according to a sigmoidal law. At the beginning, very few hosts are infected. Few hosts, then, perform scans, and the probability of clean hosts to get infected is really low. Once the number of infected hosts increases, they are spread all across the IP space. In this phase, the propagation speed is very high, and increases very quickly. Beyond a certain number of infected hosts, instead, the propagation speed decreases. This happens because the probability of finding a non infected host during scans decreases, and also due to some administrators' reaction. Once the vulnerability and its exploitation are discovered, in fact, some of the bots will be sanitized and removed from the botnet. Hence, it's difficult to identify properties which can allow to detect a botnet during the initial and the final phases of its life. By observing DNS traffic properties during the phase associated to the steepest part of the sigmoid, it is possible to imagine features which can allow to effectively discriminate between legitimate and botnet-related requests. The definition of such features, however, depends on the application context and on the problem we are willing to address.

Once the botnet is established and in a steady state as to its spreading, it becomes more difficult to detect its activity from the network activity point of view. In fact, its related statistics will look steady and hardly distinguishable

from background traffic characteristics. Therefore, it might make sense to think about some detection techniques based on packet inspection, which exploit information at the application layer. In order to be able to do that effectively, however, we have to consider all the well known drawbacks related to packet inspection, which has been criticized lately. In general, application level payload decoding is not feasible in high speed networks, since it's very hard to keep up with network traffic's pace with a detailed analysis. Hence, we need to filter out the largest portion of traffic not of any interest for our botnet hunting purpose.

Starting from these considerations in this paper we decide to analyze botnets characterized by a centralized command and control channel based on IRC protocol, so as to exclude from our analysis any traffic flow not using such protocol.

3 The Proposed Architecture for Detecting IRC Botnets

As stated in the Introduction, IRC has been one of the first protocols exploited to implement a command and control channel for botnets. IRC bots are sets of scripts connecting to IRC channels, and performing automated functions inside such channels. Initially, the bots functions were very simple and related to a small set of automated tasks. Over time, such functions evolved and allowed the bots to perform a wide range of tasks, from channel maintenance to gaming support and activity logging. The introduction of remote control for IRC bots was the preliminary step to the creation of IRC-based bot armies. Malicious bots connect autonomously to IRC channels used by botmasters to issue commands.

In the case of botnet activity detection in an IRC channel, the main steps involved in the detection process can be described as follows (see also Fig. 1). A trace containing IRC traffic is collected from the Internet. An IRC decoder detects the presence of traffic related to the IRC protocol, and decodes it according to the protocol specifications. Since we are willing to reveal the nature of IRC channels, such channels are identified and separated. A feature vector describing, at any time t, the current status of the channel under consideration is extracted and finally a classifier is fed by the feature extractor.

3.1 The Considered Features

The aim of this work is to find features which allow us to discriminate between human and botnet-related activity in an IRC channel, and test their effectiveness. The key observation here is that human users of a normal IRC channel will exhibit a behavior very different from that of automated bots waiting for, and responding to, commands from the botmaster, and the difference in such behaviors can emerge by analyzing them by means of natural language recognition techniques. The intuition is that a bot, due to its limited set of commands, will have a limited dictionary, resulting in a limited number of used terms, and a low variability of each sentence's properties. Since bot commands are structured as common shell commands, we expect to find a very structured set of sentences in a botnet channel. To be more specific, we expect most of the sentences of a

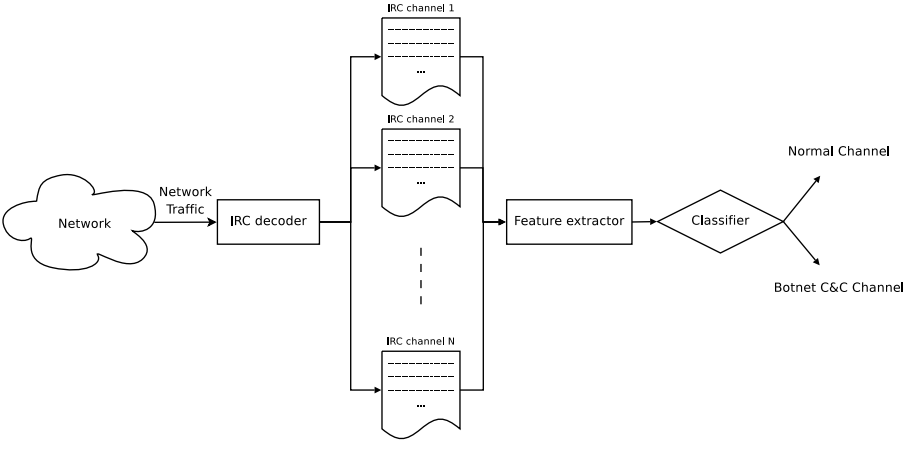

Fig. 1. Scheme of the proposed architecture for detecting IRC botnets

bot-related conversation to look like a command followed by a sequence of arguments or (argument, value) couples. On the other hand, a human conversation should be characterized by a higher variability of sentence properties, a different interaction pattern among chatters, and possibly a broader dictionary.

Based on the above considerations, a numeric model of IRC activity has been defined. Such a model consists of a number of features, which describe the degree of activity in an IRC channel, the variability and complexity of the vocabulary used in sentences typed by users and the degree of activity due to both users and control mechanisms in the channel. A feature vector is associated to each observed channel, and its values are updated at the occurrence of each event within such channel. Significant events may be related to user activity, or channel control activity. At each moment in time, a feature vector can be regarded as describing the channel status, according to the properties described by the features. In particular, the features we used have been defined as follows:

Average words number: average number of unique words in a sentence
Words number variance: variance of the number of unique words in a sentence
Unusual Nickname: percentage of unusual nicknames of channel users, defined with respect to the definitions in [13]. Such nicknames usually contain characters such as "|,%,−,_"
IRC Commands Rate: overall IRC command rate with respect to overall channel activity
PRIVMSG Rate: percentage of PRIVMSG commands with respect to overall channel activity
WHO Rate: percentage of WHO and NAMES commands with respect to overall channel activity
AVG word length: average word length in a PRIVMSG command
Word length variance: word length variance in a PRIVMSG command

Average vowel number: average number of vowels in a message
Average consonant number: average number of consonants in a message
Average punctuation signs number: average number of punctuation
String distance: Levenshtein distance [14] among words in the same message

Additional details regarding the description of the used features can be found in [15].

3.2 The Classifier

As regards the classifier of the architecture shown in Fig. 1, we propose to use an anomaly-based detector. Our choice is justified by the fact the it is quite trivial to obtain normal and "clean" IRC channels, while it could be very difficult to find really infected channels. In computer security, in fact, the main assumption at the base of anomaly detection is the fact that fraudulent activities usually deviates from legitimate activities and then can be detected comparing them to the model of normal behaviors [12].

In order to build a good model of normal channels, we propose to use a one-class classifier, the ν-SVC presented in [16] which is inspired by the Support Vector Machine classifier proposed by Vapnik in [17]. The one-class classification problem is formulated so as to find a hyperplane that separates a desired fraction, say ν, of the training patterns from the origin of the feature space. Such a hyperplane cannot always be found in the original feature space, thus a mapping function from it to a kernel space must be used. In particular, it can be proved that when the Gaussian kernel given by:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\gamma}\right), \tag{1}$$

is used, it is always possible to find a hyperplane that solves the separation problem [12]. The parameter γ, together with ν, has to be determined during the training phase of the ν-SVC.

4 Experimental Results

The results presented here were obtained by analyzing IRC logs. The normal channel logs were collected by sniffing traffic produced by activity in several real IRC channels; the botnet-related logs, instead, were collected at the Georgia Institute of Technology, in the Georgia Tech Information Security Center (GTISC) labs. The complete dataset consists of about $1,250,000$ samples, respectively belonging to eight normal rooms and nine infected rooms, as reported in Table 1.

In order to evaluate the best values for the ν and γ parameters presented in Section 3.2, we performed a grid search, by fixing different values for ν, and allowing γ's value to change in the range $(2^{-1}, 2^{-11})$. So, the best values for ν

Table 1. The considered dataset

	# of channels	# of patterns
Normal channels	8	55,000
Botnet channels	9	1,192,376

and γ have been fixed to 2^{-3} and 2^{-5}, respectively, since those values gave the best accuracy on the training data.

The tests we performed can be roughly divided into three categories. First of all, we performed a per-pattern evaluation of our approach. Then, we tested the detection ability of the proposed system, when trained with a certain number of normal channels. Finally, we investigate how the introduction of a suitable threshold can affect the performance of the system in detecting normal and infected channels.

Independent patterns. For the first set of tests, we consider the whole dataset as merely composed of patterns, without taking the nature of the problem into account. Classification is performed on a per-pattern basis. We evaluate how classification accuracy changes by increasing the percentage of patterns selected from the whole dataset for training. In Table 2 the first column indicates the percentage of patterns randomly selected from the dataset for training. As it is evident, when training the system with 50% of the normal data (about 28,000 patterns) we have very good results in terms of detection of botnet-related patterns. An higher number of normal patterns for training do not increase the overall performance.

Indivisible channels. In the second set of tests, we consider the problem's nature for the first time, both during the training and the test phase. Such tests are performed by dividing the dataset into subsets, each composed of patterns associated with the same IRC channel. In this case, training and test set are not selected randomly. In each of the tests we use all the patterns referring to a defined number of normal IRC channels for training, and the remaining channels (both normal and infected) for tests. The experimental evaluation of the system is again performed on a per-pattern estimation of classification accuracy.

Table 2. Classification accuracy on a per-pattern basis, as the number of training patterns varies

Percentage of the dataset selected for training	***Normal* channels**	***Botnet* channels**
50%	87.22%	99.86%
60%	87.18%	99.85%
70%	87.22%	99.86%
80%	87.20%	99.86%
90%	87.15%	99.85%

Table 3. Classification accuracy on a per-pattern basis, as the number of training channels varies

Training channels	***Normal* channels**	***Botnet* channels**
1	72.36%	99.70%
3	77.90%	100.00%
5	85.96%	99.45%
7	90.23%	99.98%

Therefore, in the second and third column of Table 3 we report the percentage of patterns relative to test channels which were correctly classified individually. In order to have a more robust estimate of the system performance, results on each row are the average values obtained by considering all the training sets that can be built with a specific number of normal channels. In general, it's worth pointing out that normal channel classification accuracy is lower than botnet channel classification accuracy. This may depend on the fact that botnet channels exhibit a regularity in connected users behavior patterns which is not peculiar to normal channels. The latter, in fact, may have different characterizations depending on the channel topic, the users type and the inherent variability of human conversations.

Channel's anomaly score thresholding. In this experiment, we define a per-channel anomaly score, and a thresholding mechanism for assigning a class label to each channel. The anomaly score of a channel is represented by the ratio of the number of patterns, associated to the same channel, identified as botnet-related, with respect to the number of patterns describing the same channel which are identified as describing a normal user behavior. The channel is assigned a class label, depending on the current value of the anomaly score, compared to the fixed thresholding value. In Table 4 we report the overall per-channel classification accuracy; once again values have been averaged on all the possible training sets that can be built with a specific number of normal channels. As it can be expected, experimental results show that the classification accuracy increases with the number of channels used for training. However, the accuracy decreases when the anomaly score threshold increases. In fact, a higher value for the anomaly score requires more confidence in assigning the class label. So, it is not always possible to correctly classify a channel with a too high threshold, since training

Table 4. Accuracy in channel classification by thresholding the anomaly score, as the number of training channels varies

Training channels	**Threshold**				
	6%	12%	25%	50%	75%
1	89.84%	89.55%	85.93%	74.22%	69.53%
3	97.27%	95.79%	94.64%	92.35%	83.93%
5	99.70%	97.17%	96.87%	96.28%	92.41%
7	100.00%	98.75%	98.75%	98.75%	97.50%

data may not be perfectly representative of all possible normal channels. On the other hand, it is very interesting to note that just one channel seems to be sufficient for training a system that is able to correctly classify about the 90% of the test channels, both normal and infected ones.

5 Conclusions and Future Work

In this paper, we presented the model of IRC user behavior in a channel in order to distinguish between normal and botnet-related activity. On this model we built a classification system based on a one-class Support Vector Classifier. The proposed approach demonstrated its effectiveness on a dataset of about 1,250,000 patterns extracted from 17 different IRC channels. The next step will be the deployment of the system in order to work online on a real network, sniffing traffic from the wire, and producing live and hopefully timely alerts in real-time.

References

1. Ramachandran, A., Feamster, N.: Understanding the network-level behavior of spammers. SIGCOMM Comput. Commun. Rev. 36(4), 291–302 (2006)
2. Barford, P., Yegneswaran, V.: An inside look at botnets. In: Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C. (eds.) Special Workshop on Malware Detection. Advances in Information Security, vol. 27. Springer, Heidelberg (2007)
3. Puri, R.: Bots and botnets: An overview. Technical report, SANS institute (2003)
4. Dagon, D., Zou, C., Lee, W.: Modeling botnet propagation using time zones. In: NDSS, The Internet Society (2006)
5. Strayer, W.T., Walsh, R., Livadas, C., Lapsley, D.: Detecting botnets with tight command and control. In: Proceedings 2006 31st IEEE Conference on Local Computer Networks, November 2006, pp. 195–202 (2006)
6. Akiyama, M., Kawamoto, T., Shimamura, M., Yokoyama, T., Kadobayashi, Y., Yamaguchi, S.: A proposal of metrics for botnet detection based on its cooperative behavior. In: SAINT-W 2007: Proceedings of the 2007 International Symposium on Applications and the Internet Workshops, Washington, DC, USA, p. 82. IEEE Computer Society, Los Alamitos (2007)
7. Binkley, J.R., Singh, S.: An algorithm for anomaly-based botnet detection. In: SRUTI 2006: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, Berkeley, CA, USA, p. 7. USENIX Association (2006)
8. Cooke, E., Jahanian, F., Mcpherson, D.: The zombie roundup: Understanding, detecting, and disrupting botnets, June 2005, pp. 39–44 (2005)
9. Livadas, C., Walsh, R., Lapsley, D., Strayer, W.: Using machine learning technliques to identify botnet traffic. In:31st IEEE Conference on Local Computer Networks, pp. 967–974 (November 2006)
10. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: Almeida, J.M., Almeida, V.A.F., Barford, P. (eds.) Internet Measurement Conference, pp. 41–52. ACM, New York (2006)
11. Ramachandran, A., Feamster, N., Dagon, D.: Revealing botnet membership using dnsbl counter-intelligence. In: SRUTI 2006: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, Berkeley, CA, USA, p. 8. USENIX Association (2006)

12. Giacinto, G., Perdisci, R., Del Rio, M., Roli, F.: Intrusion detection in computer networks by a modular ensemble of one-class classifiers. Information Fusion 9(1), 69–82 (2008)
13. Goebel, J., Holz, T.: Rishi: identify bot contaminated hosts by irc nickname evaluation. In: HotBots 2007: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, Berkeley, CA, USA, p. 8. USENIX Association (2007)
14. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
15. Mazzariello, C.: Irc traffic analysis for botnet detection. In: Fourth International Conference on Information Assurance and Security, IAS 2008, September 2008, pp. 318–323 (2008)
16. Schlkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471 (2001)
17. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)