

IRC traffic analysis for botnet detection ^{*†}

Claudio Mazzariello

University of Napoli Federico II
via Claudio 21, 80125 Napoli (Italy)

E-mail: claudio.mazzariello@unina.it

Abstract

Networked hosts' vulnerabilities pose some serious threats to the operation of computer networks. Modern attacks are increasingly complex, and exploit many strategies in order to perform their intended malicious tasks. Attackers have developed the ability of controlling large sets of infected hosts, characterized by complex executable command sets, each taking part in cooperative and coordinated attacks. There are many ways to perform control on an army of possibly unaware infected hosts, and an example of such techniques is discussed in this paper. We will address the problem of detecting botnets, by introducing a network traffic analysis architecture, and describing a behavioral model, for a specific class of network users, capable of identifying botnet-related activities.

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 towards a common malicious objective. Furthermore, attackers' motivation has 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* cost are some of the reasons pushing dishonest investors to spend money in this sort of underground competition for customers and resources. Some analysts are starting to refer to network security threats as *crimes* attempting to threaten people's security. The increasing dependence of national governments on mission critical networked infrastructure, therefore poses a serious risk not only for end-user related activities, but also for entire nations [9]. Not only researchers, but also government agencies are becoming worried about the potential effects of a coordinated and collaborative wide-scale threat coming from the existence of very skilled malicious users [17].

The inherent complexity of the network security issue forces the research, design and implementation of up-to-date countermeasures and solutions. In this paper a proposal for a system analyzing network traffic for botnet detection will be presented. Such a system sniffs network traffic, extracts a behavior model of specific classes of traffic, such as IRC traffic, and by means of pattern recognition techniques, aims at detect the expected differences between a normal and a malicious user.

The paper is structured as follows: in Section 2 some background and technical information are given about the botnet phenomenon; in Section 3 some related works about botnet detection are discussed; the framework for implementing the proposed solution, based on an IRC traffic model, is described in Section 4; in Section 5 some experimental results show the detection performance of the proposed traffic classification system; finally, in Section 6 some conclusive remarks are given.

2. Motivation: The Botnet Phenomenon

A Botnet can be regarded as a distributed platform for performing malicious actions. Botnets are made by *zom-*

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project).

†This work has been partially supported by the Ministero dell'Università e della Ricerca (MiUR) in the framework of the PRIN Project "Robust and Efficient traffic Classification in IP networks" (RECIPE).

bie hosts, named *bots* in this context, which are controlled by the attacker, also known as the *botmaster*, by means of a Command & Control (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.

No fully comprehensive botnet taxonomy has been proposed yet, due to the rapid evolution of the phenomenon, and to the constant discovery of novel strategies for their implementation and deployment. Nevertheless, several good references have been established so far [2, 4]. One of the properties which easily separates botnet classes is the type of command and control channel used. Such a choice in fact, dramatically influences botnet effectiveness, and its resiliency to the defenses deployed by network and system administrators. In this paper, the problem of detecting botnets characterized by a centralized command and control channel is addressed [14]. Such a structure for the command and control channel grants a high degree of simplicity in implementing the control function, performed by the botmaster via the only available channel. Yet, it suffers from the drawbacks due to the presence of a single point of failure. In fact, once the command and control channel is identified, the whole botnet is dismantled and torn down. That's the main reason why we address centralized botnet detection at the moment. 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. By analyzing DNS requests it's hopefully possible to intercept and eventually stop a bot before it connects to the rest of the botnet and gets involved in malicious actions. Furthermore, bot-issued DNS queries can allow to detect and tear down the whole command and control channel, and then the whole botnet, at once.

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 [14]. Under this as-

sumption, 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, of course, 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. Of course, in order to be able to do that effectively, 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. If we decide to analyze botnets characterized by a centralized command and control channel based on either IRC or HTTP protocol, for example, we can exclude from our analysis any traffic flow not using either of such protocols.

3. Related Works

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 [15]. Many works try to propose a botnet taxonomy [2, 12], 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 [5] 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 [18] 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 [1] and [3] methods are proposed, which combine both application and network layer analysis. In [4] 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 [10]. 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 [14] 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 [16].

4. The Proposed Framework

The proposed technique for detecting botnet activity relies on a model of IRC user behavior. The reference framework detects and decodes IRC activity within raw network traffic and, by analyzing a set of descriptive parameters, allows a classifier to separate normal activity instances from botnet-related ones. The main building blocks are depicted in figure 1.

In the following a description of the proposed framework is sketched; each building block's will be analyzed from a functional point of view.

The traffic sniffer sets a network interface card in promiscuous mode, and intercepts packets in order to analyze them. Any sniffer, such as Snort™, could be used. Common and well known sniffers, indeed, perform many functions which are unwanted in this case. Therefore, a simple and lightweight sniffing library has been developed.

As a counterpart to the network sniffer simplicity, a more complex elaboration of sniffed traffic has to be performed at the *Protocol Detection* module. Such a component is entitled to perform complex analysis in order to detect the presence of a protocol of interest for the performed analysis. The task of the protocol detector can be performed in sev-

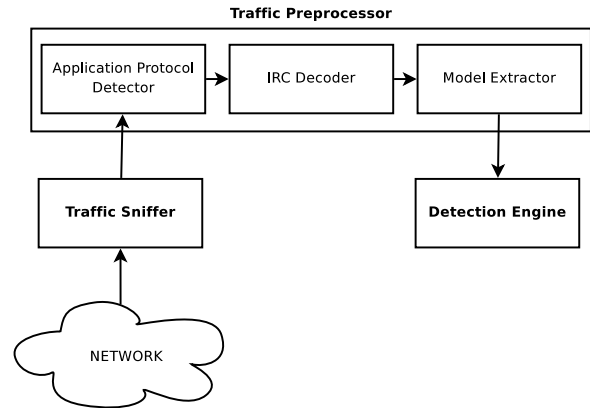


Figure 1. Traffic Analysis Framework

eral ways, from very simple to quite complex. In the case of protocols used by well known services, port based protocol identification is the most straightforward method, but it is completely ineffective when port numbers other than the standard ones are used. Many techniques have been proposed and implemented to cope with blind protocol identification [7], and some of those are also embedded in common, production level Intrusion Detection Systems, such as the PIA module for Bro [6]. Such techniques are based on the analysis of statistical properties of header or payload fields, or on the presence of some specific signatures within the payload. Some techniques rely on the direction of packets, their size and timing.

Though application level payload inspection can be unfeasible in most cases, once flows transporting the correct application level protocol are selected and isolated from the rest of the traffic, the majority of sniffed packets will be filtered out, and only those packets belonging to application protocols of interest will be relayed to further analysis modules for deeper inspection; under certain constraints and circumstances, the analysis of such a smaller fraction of packets might even be affordable on high speed links as well.

In the framework of the work described in this paper, the case of IRC-based botnets will be addressed. In such botnets, after the infection and the bot binary download phases, bots subscribe to certain IRC channels, and commands are usually issued by the botmaster either via the channel topic, or broadcast and private messages [8]. In this case, the proposed solution and the underlying model and classification technique address the problem of botnet detection during the operating phase. In fact, the model aims at describing properties of a Command and Control channel based on the expected behavior of both the bots and the botmaster, and on the interaction between them. Therefore, the next module present in the architecture is an IRC protocol detection mod-

ule. Such a module has been developed in order to comply with the standard IRC protocol definition in RFC 1459 [11]. In general, it can be transparently substituted, together with the following module, by the modules corresponding to the selected application protocol which has to be examined.

IRC channels are typically used by humans to intercommunicate. The IRC protocol was introduced in order to provide its users with a scalable and distributed environment for chatting among themselves [11]. Due to its simplicity and to its diffusion, IRC has been one of the first protocols exploited to implement a command and control channel for botnets. Bots, intended as members of a malicious botnet, are named after IRC bots. 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. The aim of this work is to find some properties which allow to discriminate between human and botnet-related activity in an IRC channel. 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 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. We aim at defining some features, also borrowed from natural language analysis theory, which allow us to detect botnet-related channels. By analyzing such features by means of pattern recognition techniques, we can implement anomaly detection easily, since "clean" IRC channels are easily available for training.

5. Experimental Evaluation

Based on the considerations in section 4, a numeric model of IRC activity has been defined. Such a model con-

sists 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.

Since this work is still in a preliminary phase, we are now testing the model properties and performance. What we are willing to understand is whether the proposed IRC behavior model is suitable for the botnet classification task.

The language used by human speakers in a chat is expected to be different from the language used by bots. The commands used by popular bot families are characterized by a limited cardinality of the used dictionary, much smaller than a chat room populated by human users. A direct consequence is a low value for both the mean and variance of words properties in a sentence. From the grammar point of view, in bot commands many punctuation characters are used, even though it's worth pointing out that such characters are often used by human users in order to express feelings or moods, as in the case of smiley, or ASCII art. Bots commonly use terms related to English language, thus inducing a high number of false alarms, or missed detections, in case of imprecise training. Languages other than English, on the other hand, are subject to different grammars, therefore generating different values for the descriptive features. Such a consideration is a constraint which can't be disregarded when defining the features to use for classification. A good model for the problem at hand must take into account the lexicon used in the channel, the analysis of users nickname, topic modeling and other more discriminant parameters. Another possible aspect to model

	normal	botnet-related
normal	25000	0
botnet-related	0	25000
Correct Classification	50000	
Wrong Classifications	0	
Accuracy	100%	
Error	0%	

Table 1. Classification performance over 50000 samples using SVM

consists of the statistics related to IRC commands and channel control activities. Commands such as PRIVMSG, JOIN, QUIT, WHO, and so on can be monitored in order to identify a difference between normal and anomalous channels. The results presented here were obtained by analyzing IRC logs. The normal channel logs were collected across several repositories of common chat room logs; the botnet-related logs, instead, were collected at the Georgia Institute of Technology, in the Georgia Tech Information Security Center (GTISC) labs. Both the training and the validation

	normal	botnet-related
normal	75010	0
botnet-related	0	74989
Correct Classification	149999	
Wrong Classifications	0	
Accuracy	100%	
Error	0%	

Table 2. Classification performance over 149999 samples using SVM

set are computed starting from such logs. The complete dataset consists of 165000 t-uples.

We will show how the model performs when classifiers based on Support Vector Machine [19] (SVM) and J48 [13] decision trees are used. The features we used are defined as follows:

Users Number: total number of users in the channel

Average words number: average number of unique words in a sentence

Average/Variance of Channel Dictionary Cardinality: mean and variance of the vocabulary's cardinality

Unusual Nickname: defined with respect to the definitions in [8]

Equal Answers: number of sentences with a common ordered subset of words

Control Commands Number: count of control commands issued

Join Number: JOIN rate in the channel

SetMode Number: SetMode rate in the channel

Nickname Changes: count of nickname changes in a channel

Ping Number: PING rate in the channel

IRC Commands Number: overall IRC command rate

Active Users Number: number of users active in the channel

	normal	botnet-related
normal	150000	0
botnet-related	0	15000
Correct Classification	165000	
Wrong Classifications	0	
Accuracy	100%	
Error	0%	

Table 3. Classification performance over 165000 samples using SVM

	normal	botnet-related
normal	24997	3
botnet-related	0	25000
Correct Classification	49997	
Wrong Classifications	3	
Accuracy	99.994%	
Error	0.006%	

Table 4. Classification performance over 50000 samples using J48

	normal	botnet-related
normal	75010	0
botnet-related	0	74989
Correct Classification	149999	
Wrong Classifications	0	
Accuracy	100%	
Error	0%	

Table 5. Classification performance over 149999 samples using J48

We will present the results in terms of confusion matrices calculated for different sizes of the used datasets, and for each used algorithm.

Three datasets of increasing size have been analyzed (50000,149999 and 165000 samples). The classification results are presented respectively in Tables 1 and 4, Tables 2 and 5, and Tables 3 and 6. Features vectors can be regarded as a representation of the channel status; they are updated synchronously with the occurrence of any event within the channel. At any time, the most recent feature vector represents the current status of the channel it refers to. In the first two experiments, the dataset consists of 50% samples for each of the two classes (normal and botnet-related), whereas in the latter there are approximately 90% samples belonging to the *normal* class and the remaining 10% belonging to the *botnet-related* class. In the case of SVM, 10-fold cross validation is performed. The detection perfor-

	normal	botnet-related
normal	150000	0
botnet-related	0	15000
Correct Classification	165000	
Wrong Classifications	0	
Accuracy	100%	
Error	0%	

Table 6. Classification performance over 165000 samples using J48

mance of the model seem to be very good. Indeed, there could be several reasons for such a result. Further investigation is needed in order to understand whether the results are biased with respect to the used dataset or to the classifiers employed. To such a purpose, further experiments are needed. Despite that, the very encouraging results make the model very promising for employment in a more complex real network scenario.

6. Conclusions and Future Work

A framework for botnet detection has been introduced. In this paper, we presented the model of IRC user behavior in a channel in order to distinguish between normal and botnet-related activity. From the classification point of view, further investigation is needed in order to understand whether the obtained results are biased due to the employed classifiers or the analyzed dataset. Furthermore, once a thorough supervised analysis has been performed, next step will be the testing of pure unsupervised anomaly detection techniques. After an effective anomaly detection scheme has been defined, the remaining module from the architecture can be deployed, in order to allow the system to work on a real network, sniffing traffic from the wire, and producing live and hopefully timely alerts in real-time. Besides the architecture-related evolution of the system, the behavior model needs to be refined and made suitable for a wide range of scenarios. Last, but not least, the same concepts can be applied to different C&C channels, once the protocol detector, the protocol decoder and the behavior model have been suitably adapted.

References

- [1] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi. A proposal of metrics for botnet detection based on its cooperative behavior. In *SAINT-W '07: Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*, page 82, Washington, DC, USA, 2007. IEEE Computer Society.
- [2] P. Barford and V. Yegneswaran. An inside look at botnets. In M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, editors, *Special Workshop on Malware Detection*, volume 27 of *Advances in Information Security*. Springer Verlag, 2007.
- [3] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 7–7, Berkeley, CA, USA, 2006. USENIX Association.
- [4] E. Cooke, F. Jahanian, and D. Mcpherson. The zombie roundup: Understanding, detecting, and disrupting botnets. pages 39–44, June 2005.
- [5] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *NDSS*. The Internet Society, 2006.
- [6] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 18–18, Berkeley, CA, USA, 2006. USENIX Association.
- [7] M. Gebiski, A. Penev, and R. K. Wong. Protocol identification of encrypted network traffic. In *Web Intelligence*, pages 957–960. IEEE Computer Society, 2006.
- [8] J. Goebel and T. Holz. Rishi: identify bot contaminated hosts by irc nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.
- [9] M. Lesk. The new front line: Estonia under cyberassault. *Security and Privacy, IEEE*, 5(4):76–79, July-Aug. 2007.
- [10] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer. Using machine learning techniques to identify botnet traffic. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 967–974, Nov. 2006.
- [11] J. Oikarinen and D. Reed. Internet relay chat protocol. RFC, May 1993.
- [12] R. Puri. Bots and botnets: An overview. Technical report, SANS institute, 2003.
- [13] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [14] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In J. M. Almeida, V. A. F. Almeida, and P. Barford, editors, *Internet Measurement Conference*, pages 41–52. ACM, 2006.
- [15] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
- [16] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using dnsbl counter-intelligence. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 8–8, Berkeley, CA, USA, 2006. USENIX Association.
- [17] G. P. Schaffer. Worms and viruses and botnets, oh my!: Rational responses to emerging internet threats. *IEEE Security and Privacy*, 4(3):52–58, 2006.
- [18] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting botnets with tight command and control. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 195–202, Nov. 2006.
- [19] V. Vapnik, S. E. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation and signal processing. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *NIPS*, pages 281–287. MIT Press, 1996.