Data Confirmation for Botnet Traffic Analysis

Fariba Haddadi(✉) and A. Nur Zincir-Heywood

Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada
{haddadi,zincir}@cs.dal.ca

Abstract. In this paper, we propose a systematic approach to generate botnet traffic. Given the lack of benchmarking botnet traffic data, we anticipate that such an endeavour will be beneficial to the research community. To this end, we employ the proposed approach to generate the communication phase of the Zeus and Citadel botnet traffic as a case study. We evaluate the characteristics of the generated data against the characteristics of a sandbox Zeus botnet, as well as the Zeus and Citadel botnet captures in the wild provided by NETRESEC and Snort. Our analysis confirms that the generated data is comparable to the data captured in the wild.

Keywords: Botnet traffic analysis · Traffic generation and evaluation

1 Introduction

In the world of fast growing Internet having a secure infrastructure is the primary need to protect users' identity and information. Botnets- among various types of malware- are recognized as one of the main threats against cyber security. A botnet is a set of compromised hosts that are under the remote control of a botmaster. To detect botnets, various detection mechanisms have been proposed by researchers where many rely on network traffic behaviour analysis. Some focus on specific types of botnets while others attempt to build general models. However, in all cases, the first challenge, is to obtain realistic data that represents botnets traffic (behaviour). Due to the malicious nature of such data, there are very few publicly available data sets in the field. Therefore, different approaches have been explored by the researchers to obtain the necessary data for their research purposes. These approaches include (but not limited to): (i) Running botnet binaries (that are publicly available or modifying such binaries) in sandbox environments and capturing the traffic [1–4]; (ii) Obtaining captured Honeynet traffic [5,6]; or (iii) Obtaining traffic from a network operator or a security company [7]. The first approach employs publicly available botnet binaries. This implies that any traffic obtained represents the old behaviours because only older versions of such binaries are publicly available. The second approach employs honeynets. Even though this method can catch newer botnet behaviours, it is also challenging to set up honeynets that can simulate real applications in normal behaviour. So there is no guarantee on the quality of the traffic attracted to the honeynet. The third

F. Cuppens et al. (Eds.): FPS 2014, LNCS 8930, pp. 329–336, 2015.
DOI: 10.1007/978-3-319-17040-4_21

approach is the only case where most up to date behaviours can be analyzed by obtaining data from network companies. However, this type of data has privacy issues so such data might not be made public for further benchmarking and evaluation purposes. We anticipate that if a systematic approach can be found, then potentially publicly available benchmarking data sets can be generated without such problems. Thus, in this work, we propose a systematic approach to generate botnet traffic data representing the connection phase of botnet traffic. In this phase, the infected host intends to locate the C&C server and starts a connection with the server. Once we generate such data using the proposed approach, we demonstrate that such an approach could indeed generate realistic behaviours of the botnet traffic. To this end, we evaluated our approach and confirm that the data generated represent similar behaviour to the botnet traffic that is captured in the wild and the botnet traffic that is generated in a sandbox environment based on publicly available bot binaries. To the best of our knowledge, this is the first work with such a goal. To achieve our goal, we follow two steps to evaluate the data after it is generated: (i) Analysis via flow measurements; and (ii) post-classification analysis via decision tree.

2 Related Work

Gu et al. developed BotMiner based on group behaviour analysis to detect botnets [5]. They collected the botnet data sets by employing publicly available botnet binaries as well as using Honeynet data sets. As for the normal traffic, university campus network traces were employed. Wurzinger et al. proposed an approach to detect botnets by correlating the commands and responses in the C&C channels of the monitored network traces [1]. Running a publicly available bot binary in a controlled environment, they collected the data sets used in this work. Kirubavathi et al. designed specifically an HTTP-based botnet detection system using a multilayer Feed-Forward Neural Network [2]. To evaluate the proposed system, botnet binaries were used in the lab. Strayer et al. developed an IRC botnet detection framework that makes use of classification and clustering techniques [3]. Again, they captured their data in the lab by running bot binaries. Francois et al. proposed a NetFlow monitoring framework that leverages a simple host dependency model to track communication patterns and employed linkage analysis and clustering techniques to identify similar botnet behavioral patterns [7]. They obtained traffic traces from an Internet operator company. Zhao et al. investigated a botnet detection approach based on flow intervals [6]. They also employed botnet binaries in the lab and Honeynet project data as for the normal traces, they used Lawrence Berkeley National Laboratory data sets. Haddadi et al. designed a machine learning based detection system [8]. Network traces representing malicious and normal behaviour were generated in the lab based on botnet and legitimate public data.

3 Methodology

A typical advanced botnet has five phases (i.e. lifecycle): (1) the initial infection phase where exploitation techniques are used to find the vulnerabilities and to infect the target host, (2) the secondary infection phase where the botnet shellcode is executed on the victim machine to fetch the image of the bot binary to install, (3) the connection phase where the bot binary establishes the C&C channel, (4) the malicious C&C phase in which the established C&C channel is utilized by the bot master to send the command and (5) the update and maintenance phase where the botmaster updates the bots when required. It is known that to enhance their functionality and to avoid detection systems, botnets tend to use automatic algorithms in different phases of their lifecycle. Fluxing techniques are one of such methods that botnets use to hide the C&C servers using automatic Domain Generation Algorithms (DGAs). Citadel and Zeus are just two of the recent examples of such botnets. A DGA is designed to periodically generate a large list of domain names. The infected computer (bot) may locate the C&C server by querying (sending a DNS query to retrieve the associated IP address) a list of domain names. However, only one of the domain names from the list is valid at a given time. Once a response is received for the query, the bot establishes a connection to the C&C server to receive updates and commands. The list of domain names provided to the bot is large enough so that blacklisting (blocking) at the firewall level is not straight forward. Therefore, the domain fluxing technique is highly utilized in current botnets. Since there are publically available lists of domain names generated by botnet DGAs, we implemented a systematic approach to generate network traffic using these botnet domain names to represent the botnet communication during the connection phase of the botnet lifecycle.

Once the botnet communication traffic is generated, we analyze and evaluate it against three publicly available botnet traffic data that are captured in the wild and one botnet binary based traffic data captured in the sandbox environment. In doing so, our aim is to confirm that our generated traffic is representative of the publicly available ones and the sandbox ones.

We generate the representative botnet traffic flows using the publicly available (from legitimate resources) C&C domain name lists. As for generating the representative legitimate (normal) traffic, we use a legitimate domain name list. This ensures that both the attack and the normal traffic are generated using the publically available domain name lists to enable re-engineering of the approach for benchmarking purposes. For the malicious domain name lists, the two of the most recent botnets, namely Citadel and Zeus are used. The domain name lists for these two botnets are obtained from ZeusTracker and DNS-BH project websites [9,10]. As for the legitimate name list, some of the most frequently requested domain names from Alexa lists (domain names of the high ranked web sites) are used [11]. Both the malicious and legitimate behaviour data use the HTTP protocol as their communication protocol. So, to generate the traffic, we developed a program to establish HTTP connections with the domain names from the aforementioned lists. First, this program sends out DNS queries for the domain names in the lists. If in return it receives a proper DNS response,

Table 1. Data specification

Data set	No. of Packets	No. of Exported Flows
Alexa (NIMS)	21210	7473
Citadel (NIMS)	79516	5772
Citadel (NETRESEC)	15239	226
Zeus-1 (NIMS)	108947	14884
Zeus (NETRESEC)	7453	361
Zeus (Snort)	6995	145
Zeus-2 (NIMS)	32818	874

Table 2. Softflowd feature set

Softflowd Features							
Duration	Src-AS	Dst-AS	In-If	Out-If	Total-Pkt	F-Pkt	B-Pkt
Total-Byte	F-Byte	B-Byte	Flows	ToS	Src-ToS	Dst-ToS	Src-Msk
Dst-Msk	FWD	Src-Vlan	Dst-Vlan	bps	pps	Bpp	

this indicates that the domain name is registered and is associated with a valid IP address. Then, our program attempts to establish an HTTP connection with that IP address. Hereafter, we will refer to our generated data sets using this approach as Zeus-1 (NIMS) and Citadel (NIMS) botnet data sets.

In addition, we employed the few botnet traffic data sets available at NETRESEC [12] and Snort [13] web sites for Zeus and Citadel botnets. Since many works in the literature employed running publicly botnet binaries in the lab (sandbox), we also run a Zeus botnet in a controlled environment and captured the traces [4]. This toolkit is also analyzed and employed in [14]. However, we could not do the same for the Citadel botnet given that there is no publicly available free kit, to the best of our knowledge. Hereafter, this Zeus data is referred to as Zeus-2 (NIMS).

4 Evaluation

As a case study, we focus on generating Zeus and Citadel traffic data using our proposed system. For analyzing the traffic data generated, features are extracted using IP-flow technology. Specifically, we employ Softflowd [15], which is an open source tool based on the NetFlow standard [16]. Table 1 presents the number of captured packets and the corresponding extracted flows for each of the data sets employed in this work. Table 2 presents the features that are utilized in our approach.

4.1 First Step–Analysis via Flow Measurements

As a first step, we anticipate that it will be beneficial to analyze the data sets on some of the most important features (used by other researchers [1,3,4,6]) of

a flow. These features are: the flow duration (Duration), the number of bytes per flow (Total-Byte) and the number of packets per flow (Total-Pkt).

Given the wide range of the HTTP usage on the Internet, most recent botnets employ HTTP protocol to hide their malicious activities among the normal web traffic. Citadel and Zeus utilize HTTP protocol to communicate with their bots, too. Thus, to analyze and compare the aforementioned data sets, we filter out the non-HTTP flows. This way all the background information is removed from the data and therefore, the data sets can be compared on their fundamental properties.

The analysis show that the majority of flows durations (96 %, 70 %, 91 % in Zeus (NIMS), Zeus (Snort) and Zeus (NETRESEC), respectively) are less than 50 s long (placed in the first bucket). However, 20 % of Zeus (Snort) and 67 % of Zeus-2 (NIMS) flows last longer than the other two Zeus data sets. Zeus-2 (NIMS), Zeus (Snort) and Zeus (NETRESEC) have more flows with higher number of transmitted packets (30 packets or more) and more bytes, respectively. We predict that these differences are due to the connection time, because we do not keep the connection between our system and botnet C&C servers open long. Basically, we close the connection once this phase of communication is finished.

Moreover, the analysis also shows that the pattern for duration of flows for the Citadel (NIMS) botnet, is different from the pattern of the duration of the flows for Citadel (NETRESEC). On the other hand, there is almost no difference (among the two data sets) between the number of packets and the number of bytes transmitted per flow. Therefore, we can conclude that the Citadel C&C connections in NETRESEC data were kept open without any actual client-server command and response communication. Since keeping a connection open for a long time is one of the signs of malicious activities, botnets tend to close the connection when the necessary information is sent/received and open a new connection for the next round when needed. However, a high number of connections also raises flags for malicious behaviour. Therefore, there is a trade-off between these two parameters when the botnets aim to hide their malicious activity in the normal traffic. Thus, in our approach, we decided to keep the communication duration with the C&C servers as short as possible, because it gives us a good balance for the aforementioned trade-offs. In summary, the analysis of the important features of the flows of the generated traffic and the flows of the traffic captured in the wild seem to be similar except the durations of the flows as explained above. More detailed analysis and the visualizations can be found at [17].

4.2 Second Step–Post-classification Analysis via Decision Tree

To further analyze how similar the generated data (by our proposed approach) is to the data captured in the wild and the data generated in the sandbox, we employed a post-classification analysis using a decision tree. To this end, we trained a C4.5 classifier on the proposed data generation approach and tested the trained model on the botnet data from the NETRESEC and Snort web sites as well as on the sandbox data. Table 3 presents the results of this step.

Table 3. C4.5 decision tree classification results

Training data set	Testing data set	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
Citadel (NIMS)	Citadel (NETRESEC)	100 %	100 %	0 %	0 %	0 %
Zeus-1 (NIMS)	Zeus (Snort)	81 %	81 %	0 %	0 %	19 %
Zeus-1 (NIMS)	Zeus (NETRESEC)	79 %	79 %	0 %	0 %	21 %
Zeus-1 (NIMS)	Zeus-2 (NIMS)	88 %	88 %	0 %	0 %	12 %
Zeus-1 (NIMS)	Zeus-1 (NIMS-unseen)	84 %	84 %	0 %	0 %	16 %

In traffic classification, two metrics are typically used in order to quantify the performance of the classifier: Detection Rate (DR) and False Positive Rate (FPR). In this case, DR reflects the number of botnet flows correctly classified and is calculated using: $DR = TP/(TP+FN)$; whereas FPR reflects the number of legitimate flows incorrectly classified as botnet flows. FPR is calculated using: $FPR = FP/(FP + TN)$. Naturally, a high DR and a low FPR are the most desirable outcomes. On the other hand, False Negative Rate (FNR) indicates that botnet flows are classified as legitimate flows, whereas TNR, True Negative Rate, indicates the correctly classified legitimate flows. As shown in Table 3, the trained model on NIMS Citadel could detect all of the NETRESEC Citadel correctly (100 % TP rate).

In this case, our post-classification analysis show that the main features that the C4.5 classifier employ for the Citadel trained model are: Duration, Total-Pkt, Total-Byte, bps, pps and Bpp (from Table 2). Moreover, in the first step of our analysis, duration feature of a flow presented the most discrepancy between the NIMS botnet data set and the one captured in the wild. However, the classification results indicate that this discrepancy does not seem to have a negative effect on the identification of Citadel botnet behaviour in a given traffic trace. This might be because of the importance of the other features employed. This implies that NIMS Citadel data set is similar to the real-life NETRESEC Citadel data set. Therefore, we can confirm that the proposed approach seems to generate realistic Citadel botnet traffic.

Table 3 also presents the classification results on the Zeus botnet traffic. Again, we trained a C4.5 classifier on Zeus-1 (NIMS) data set, which is generated by the proposed approach. Then, we tested the trained model against the Zeus-2 (NIMS) and Zeus data sets from Snort and NETRESEC. As the results indicate, the DRs of the Zeus-2, Snort and NETRESEC data sets vary between 79 % to 88 %. This is lower than the DR for Citadel, but still a promising performance[1].

To further analyze the generated data by the proposed approach on Zeus botnet, we run another experiment. Since NIMS Zeus generated traffic was bigger than Alexa legitimate data (and therefore had more flows), we only used a

[1] Since these test data sets are one class data sets (only malicious), there is no TNR and FPR.

fraction of the Zeus HTTP traffic to build a balanced Zeus-Alexa training data set for the results presented in Table 3. Hence, we had some unseen Zeus flows that were not included in the training process. Thus, we also tested the trained Zeus model on these unseen Zeus flows. As the result shows, we obtained the DR of 84 %, Table 3. The performance is almost the same as the classification result when testing the trained model on other data sets such as NETRESEC and Snort. This seems to indicate the complicated behaviour of Zeus botnet and confirm that the results are not side effects of the generated data being different (or unrealistic), but rather the classifier performs the same on both the generated data and the data captured in the wild for Zeus botnet. This experiment also indicates that Zeus botnet behaviour is more complicated than Citadel botnet behaviour. Moreover, it seems to hide in the legitimate HTTP behaviour better than Citadel could.

In short, our 2-step analysis indicates that the traffic generated using our proposed approach is valid and comparable to the botnet traffic captured in real life (Snort and NETRESEC data) as well as the traffic captured in a sandbox environment (Zeus-2 NIMS). In other words, traffic generated based on the proposed approach can be employed in botnet behaviour analysis for representing real data. We provide our NIMS data sets for further benchmarking purposes at the following URL:

https://web.cs.dal.ca/~haddadi/data-analysis.htm

5 Conclusion

Due to the high reported botnet infection rate and its wide range of illegal activities, botnets are one of the main threats against the cyber security. One problem of this field is the availability of public data sets for meaningful benchmarks and comparisons among the botnet identification systems. In this work, we propose a systematic approach to generate botnet traffic and analyze its properties against that of available botnet traffic captures in the field. To this end, we explore a 2-step analysis. In the first step, we analyzed the data sets using the main features of the botnet traffic reported in the literature [1,3,6]. The results of this step show that the data generated by the proposed approach and the data captured in the wild by NETRESEC and Snort, are not very different in terms of their main measured features. However, the sandbox traffic data has noticeable differences (compared to the data captured in the wild) in terms of most of the measured features. In the second step, we perform a post-classification analysis on the data sets using the C4.5 classifier. In this step, we train the classifier with the data generated by the proposed approach, and test the trained model on the NETRESEC, Snort and the sandbox data sets. The results indicate that the trained model could classify the unseen data captured in the wild and the sandbox data with a high DR and a low FPR. This suggests that the botnet data generated by the proposed approach confirms to the botnet data captured in the wild or in the sandbox. We anticipate that this can further enable benchmarking and evaluation efforts in this research area. Future work will explore employing the proposed traffic generation approach for other botnets.

Acknowledgments. This research is supported by the Natural Science and Engineering Research Council of Canada (NSERC) grant, and is conducted as part of the Dalhousie NIMS Lab at https://projects.cs.dal.ca/projectx/.

References

1. Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., Kirda, E.: Automatically generating models for botnet detection. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 232–249. Springer, Heidelberg (2009)
2. Kirubavathi Venkatesh, G., Anitha Nadarajan, R.: HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network. In: Askoxylakis, I., Pöhls, H.C., Posegga, J. (eds.) WISTP 2012. LNCS, vol. 7322, pp. 38–48. Springer, Heidelberg (2012)
3. Strayer, W.T., Lapsely, D., Walsh, R., Livadas, C.: Botnet detection based on network behavior. Adv. Inf. Secur. **36**, 1–24 (2008)
4. Haddadi, F., Runkel, D., Zincir-Heywood, A.N., Heywood, M.I.: On botnet behaviour analysis using GP and C4.5. In: GECCO Comp, pp. 1253–1260 (2014)
5. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol- and structure- independent botnet detection. In: 17th USNIX Security Symposium, pp. 139–154 (2008)
6. Zhao, D., Traore, I., Ghorbani, A., Sayed, B., Saad, S., Lu, W.: Peer to peer botnet detection based on flow intervals. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) Information Security and Privacy Research. IFIP Advances in Information and Communication Technology, vol. 376, pp. 87–102. Springer, Heidelberg (2012)
7. François, J., Wang, S., State, R., Engel, T.: BotTrack: tracking botnets using netflow and pagerank. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) NETWORKING 2011, Part I. LNCS, vol. 6640, pp. 1–14. Springer, Heidelberg (2011)
8. Haddadi, F., Morgan, J., Filho, E.G., Zincir-Heywood, A.N.: Botnet behaviour analysis using IP flows with HTTP filters using classifiers. In: Seventh International Workshop on Bio and Intelligent Computing, pp. 7–12 (2014)
9. Zeus Tracker. https://zeustracker.abuse.ch/
10. DNS-BH- Malware Domain Blocklist. http://www.malwaredomains.com/
11. Alexa. http://www.alexa.com/topsites
12. Publicly available PCAP files. http://www.netresec.com/?page=PcapFiles
13. Zeus Trojan Analysis. https://labs.snort.org/papers/zeus.html
14. Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the zeus botnet crimeware toolkit. In: Eighth Annual International Conference on Privacy, Security and Trust, pp. 31–38 (2010)
15. Softflowd project. http://www.mindrot.org/projects/softflowd/
16. Cisco IOS NetFlow. http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html
17. Haddadi, F., Zincir-Heywood, A.N.: Data confirmation for botnet traffic analysis. Technical report (2014). https://www.cs.dal.ca/research/techreports/cs-2014-01