On the Effectiveness of Different Botnet Detection Approaches

Fariba Haddadi, Duc Le Cong, Laura Porter, and A. Nur Zincir-Heywood

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

Abstract. Botnets represent one of the most significant threats against cyber security. They employ different techniques, topologies and communication protocols in different stages of their lifecycle. Hence, identifying botnets have become very challenging specifically given that they can upgrade their methodology at any time. In this work, we investigate four different botnet detection approaches based on the technique used and type of data employed. Two of them are public rule based systems (BotHunter and Snort) and the other two are data mining based techniques with different feature extraction methods (packet payload based and traffic flow based). The performance of these systems range from 0% to 100% on the five publicly available botnet data sets employed in this work. We discuss the evaluation results for these different systems, their features and the models learned by the data mining based techniques.

Keywords: Feature extraction, traffic analysis, botnet detection.

1 Introduction

A network of compromised hosts (aka bots) that are remotely controlled by a master (aka botmaster) is called a botnet. These infected bots perform various malicious tasks such as spreading spam, conducting Distributed Denial of Service (DDOS) attacks or identity thefts to name a few. Hence, with the high reported infection rate, the vast range of illegal activities and powerful comebacks, botnets are one of the main threats against the cyber security.

Given that botnets use automatic update mechanisms, automatic pattern discovery could potentially enable security systems to adapt to such changes in the botnet evolution. The clustering and classification techniques that are used for traffic analysis require the network traffic to be represented in a meaningful way to enable pattern recognition. Thus, an important component for such systems is extracting the features (attributes) from the network traffic. These features can be extracted per packet (or in some cases specific packets) or per flow[1] basis.

[1] Flow is defined as a logical equivalent for a call or a connection in association with a user specified group of elements [13]. The most common way to identify a traffic flow is to use a combination of five properties (aka 5-tuple) from the packet header, namely source/destination IP addresses and port numbers as well as the protocol.

J. Lopez and Y. Wu (Eds.): ISPEC 2015, LNCS 9065, pp. 121–135, 2015.
DOI: 10.1007/978-3-319-17533-1_9

Network packets include two main parts: (i) Packet header, which includes the control information of the protocols used on the network, and (ii) Packet payload, which includes the application information used on the network. The per-packet analysis can use any of these two parts while per-flow analysis only utilize network packet headers. Hence, in this work, we evaluate both: Packet payload based and Flow based approaches. Since recent botnets tend to use encryption to hide their information and methodology from the detection systems, clearly the flow-based detection systems have advantage over the packet-based systems given that they can be applied to encrypted traffic (where the payload is opaque). However, we aim to understand how much could be gained (loss) in terms of performance when a system employs payload analysis (flow analysis). To this end, we employ not only data mining techniques but also publicly available intrusion/botnet detection systems to measure performance for both the payload and the traffic flow analysis.

In this case, we evaluate Snort and BotHunter as the rule based detection systems. Snort is a popular intrusion detection and prevention system (IDS/IPS). It is open source and therefore its rule set can be customized easily. BotHunter, which is another publicly available system, utilizes the Snort sensors and customizes Snort rule set to specifically detect botnets. In summary, in this work, we have evaluated Snort, BotHunter, a data mining based packet header/payload analysis system and a data mining based traffic flow analysis system as the four different approaches for botnet detection. In the case of data mining based approaches, we have employed four different machine learning algorithms, namely C4.5 decision trees, Support Vector Machines (SVM), K- Nearest Neighbour (KNN) and Bayesian Networks, for analyzing (i) traffic flows only and (ii) all the packets including the payload. Last but not the least, we have evaluated all of the approaches on five different publicly available botnet data sets coming from different resources to evaluate how well these different approaches could generalize.

The rest of the paper is structured as follows: Related works on botnet traffic analysis are summarized in section 2. The evaluated approaches and the methodology are discussed in Section 3. Evaluation and results are provided in section 4. Finally, conclusions are drawn and the future work is discussed in section 5.

2 Background and Related Work

Unlike first botnets that had a list of exploits to launch on targets where all the commands were set at the time of infection, today a typical advanced bot uses five stages to create and maintain a botnet [10]. The first stage is the initial infection stage. In this stage, attacker infects the victim using several exploitation techniques to find its existing vulnerabilities. In the second stage, secondary injection, the shell-code is executed on the infected victim to fetch the image of the bot binary. Bot binary then installs itself on the victim. At this time, the infected machine is completely converted into a bot. The next stage is the connection stage. In this stage, the bot binary establishes the C&C

channel to be used by the bot master. Once the connection is established then the malicious C&C stage, the fourth stage, starts. This is when the master sends the commands to the botnet, short for bot network. Finally, when the master needs to update the bots for one reason or another, the update and the maintenance stage starts.

Since botnets have employed different protocols, topologies and techniques to implement the stages of their lifecycle while avoiding detection, naturally an arms race has started between the botnets and the detection systems. Thus, detection methods can be categorized based on the data being analyzed and the approach employed. From the data perspective, while some techniques focus on malware source/binary analysis, others use host and/or network based data. On the other hand, rule (signature) based traffic analysis and anomaly detection are some of the highly employed data analysis approaches. For the rule based as well as anomaly based approaches, the rules or anomalies can be obtained via data analysis performed by a human expert or can be automatically generated via a support system using data mining algorithms to assist the human expert.

Gu et al. [8] developed a system called BotHunter, which correlates Snort IDS alerts to detect botnets. The correlation process is based on the fact that all botnets share a common set of actions as a part of their lifecycle. This technique works the best when an infected bot has passed all the phases of its lifecycle when being monitored by BotHunter. Payload analysis is a part of the detection procedure in this system. Wurzinger et al. proposed an approach to detect botnets based on the correlation of commands and responses in the monitored network traces [18]. To identify traffic responses, they located the corresponding commands in the preceding traffic. Then, using these command and response pairs, the detection model was built focusing on IRC, HTTP and P2P botnets. Traffic features such as the number of non-ASCII bytes in the payload were analyzed to characterize bot behavior. Perdisci et al. proposed a network-level malware clustering system focusing on HTTP-based malwares [12]. The similarity metrics among HTTP traffic traces were defined and used to develop the malware clustering system where the clusters resulted in the signatures. Specifically, to decrease the computational cost and obtain high quality clusters, multi-level clustering was employed. Celik et al. proposed a flow-based botnet C&C activity detection system using only headers of traffic packets [7]. They investigated the effect of calibration of time-based flow features. They employed techniques such as C4.5, Naive-Bayes and logistic regression. Wang et al. proposed a fuzzy pattern recognition approach to detect HTTP and IRC botnets' behavioral patterns [16]. It is known that botnets query several domain names in a given period of time to identify their C&C server, and then form a TCP connection with the C&C server. So, Wang analyzed the features of DNS queries (such as the number of failed DNS responses) and TCP flows to detect malicious domain names and IP addresses. Zhao et al. investigated a botnet detection system based on flow intervals [20]. Flow features of traffic packets were utilized with several ML algorithms focusing on P2P botnets such as Waledac.

In our previous work, we proposed a flow based botnet detection system [10]. We evaluated five different feature sets extracted by open source flow exporters (Maji, YAF, Softflowd, Netmate and Tranalyzer) and investigated the effect of those flow features in botnet detection. Since botnets employ various protocols as their communication carrier, we also investigated the effect of protocol filters. In this work, we will follow on the results of these evaluations.

3 Methodology

As discussed in section 2, network traffic has been analyzed in various ways to detect botnets. However, the differences between these methods are not only based on the analysis method or technique used but also are based on the specific parts of the network traffic traces being analyzed and moreover the features extracted. Some systems/approaches [8,18] require both the payload and the header section of the packets to extract the necessary features while others [7,20,10] only need the header of the packets. Between these two categories, the systems that can detect botnet communication only based on the packet header do have privilege over the other category given that they can be employed on encrypted traffic where the packet payload is opaque. The importance of such systems can be better understood knowing that the most recent aggressive botnets employ encryption to better hide themselves and their information from the detection systems. As discussed earlier, to explore the effectiveness of such systems, we aim to evaluate and analyze the following systems for botnet detection: (i) Packet payload based system; (ii) Flow based system; (iii) Snort intrusion detection system and (iv) BotHunter botnet detection system.

3.1 Systems Employed

Packet Payload Based System: Some of the works in the literature proposed specific packet analysis methods to detect botnet behaviour [9,16]. These systems have focused on specific packets and features from the header and/or payload sections of these packets to identify the type of malware they are interested in. For example, Haddadi et al. [9] extracted the domain name from the DNS packets to detect automatically generated malicious domain names while Mohaisen et al. [11] introduced a set of features focusing on the Zeus botnet. We employ the features introduced by Mohaisen et al. in our evaluations for packet payload based system. Table 1 presents the selected features for this approach.

Since some of the data mining techniques employed in this work can only be applied to numeric features, string to numeric feature conversions are performed and the quartile object sizes are calculated for each of the data sets. Detailed information of the features can be found in [11]. Once the features are extracted from each data set, four classifiers (C4.5, SVM, KNN, Bayesian Networks) are used for botnet detection.

Table 1. Packet-based approach– network features

Feature set	
Port	Source and destination port numbers
Connections	TCP, UDP, RAW
Request type	GET, HEAD, POST
Response type	Response code 200–599
Object Size	Categorised quartiles (1–4)
DNS	MX, NS, A records, PTR, SOA, CNAME

Flow Based System: Among the approaches that use the information of packet headers only, flow based feature extraction methods are highly employed in the recent literature [7][10][16]. In such approaches, communication packets are aggregated into flows and then statistics are calculated. Given that botnets employ encryption techniques to avoid the detection systems that analyze the communication information embedded in the packet payload, flow based approaches can be very effective since they use only network packet headers. Hence, we develop a flow based botnet detection system based on the results obtained from our previous work [10]. The critical phase of such a system is the flow exporting. Our previous work on the effect of such exporters reported Tranalyzer as the best performing flow exporter among the five tools (Maji, YAF, Softflowd, Tranalyzer and Netmate) that we have evaluated under multiple scenarios. Hence, in this work, we employ Tranalyzer to export the flows. After extracting the flow features, the aforementioned classifiers are employed to detect the botnet behaviour as an early warning system. It should be noted here that we employ all of the features exported by Tranalyzer as inputs to the data mining techniques except the IP addresses, port numbers and any non-numeric features. The reasons behind this are two folds: IP addresses can be spoofed whereas port numbers can be assigned dynamically. Thus, employing such features may decrease the generalization abilities of the detection system for unseen behaviours. On the other hand, the presentation of non-numeric features may introduce other biases to the detection system so it is left to the future work to introduce such features.

Tranalyzer: Flow exporters summarize network traffic utilizing the network packet headers only. These tools collect packet information with common characteristics such as IP addresses and port numbers, aggregate them into flows and then calculate some statistics such as the number of packets per flow etc. Tranalyzer is a light weight uni-directional flow exporter that employs an extended version of NetFlow feature set. This tool exports both the binary and the ASCII formats and therefore, does not require any collector. This makes it very easy to use. Tranalyzer supports 93 flow features that can be categorized into Time, Inter-arrival, Packets&Bytes and Flags groups. More detailed information on the tool and its feature set can be found in [4,10].

Snort Intrusion Detection System: Snort is an intrusion detection and prevention system that analyzes packet payload as well as packet header data to detect any evidence of harmful actions that match predefined signatures (rule sets) [1].

Some of these pre-defined rules/signatures take advantage of payload information while the others require only the header features to be analyzed. Snort has been supported by two rule sets: VRT (Vulnerability Research Team), which is the official rule set for Snort, and ET (Emerging Threat), which is published by emergingthreats.com. In our evaluations, we used the VRT rule set. We discuss this in more detail in section 4.3. These two main rule sets have come with many rules that aim to cover all possible network conditions. Hence, users should carefully enable the rules that fit their network conditions and alert priority settings and disable the others. Since 1998, Snort has been known and used in network security area given that it is a cross-platform open source IDS/IPS that can be modified to fit the network security challenges and needs as shown by the BotHunter research.

BotHunter botnet Detection System: Gu et al. introduced BotHunter as a botnet detection system and made it publicly available. This tool uses the combination of Snort and a clustering approach to detect botnet infections. BotHunter is based on the idea of all botnet infection processes are similar and can be illustrated by a lifecycle model explained in section 2. Hence, it uses a modified version of Snort with its plugins to detect specific bot actions of the lifecycle and then correlates the Snort alerts to detect the botnets' behaviour and infected machines. The developers have modified and selected the botnet related Snort rules, developed many additional rules and inserted the IP address checking to Snort rules to make BotHunter's Snort sensors work more efficiently. The initial version of BotHunter used two plugins: *SLADE* and *SCADE*, which are designed for the anomalous traffic pattern and payload detection. Recently, these plugins are replaced by three new plugins: (i) *bhDNS* for malicious DNS analysis, (ii) *bhSD* for scanning detection; and (iii) *Con-P2P* for Conficker-C P2P detection and ethernet tracking. All new plugins use existing information like DNS lists, IP lists, port lists and so on to detect malicious (botnet) behaviour. It should be noted here that to be able to detect new botnets, BotHunter relies on Snort signature updates of new malicious behaviours where the signatures use header and payload information.

3.2 Data Mining Techniques Employed

As discussed in section 3, the first two systems evaluated employ various data mining techniques for botnet detection. These include: C4.5 decision tree, SVM, KNN and Bayesian Networks. These are the four well-known machine learning algorithms that are widely used in the literature for network traffic classification [11,16,20,10].

C4.5 is an extension to ID3 algorithm that aims to find the small decision trees (using pruning) and then convert the trained tree into an if-then rule set. The algorithm employs a normalized information gain criterion to select attributes from a given set of attributes to determine the splitting point of the decision tree. *SVM* is a binary learning algorithm that can easily be extended to K-class classification by constructing k two-class (binary) classifiers. The goal of this

classification algorithm is to build an N- dimensional hyperplane that optimally separates the samples of data into classes with maximal margin. *KNN* stores the training samples and uses Euclidean distance to compute the distance of the test instances from the K nearest neighbour of n-dimensional training feature space. The classifier then assigns a class label to the test instance using a majority voting mechanism. *Bayesian Networks* are graphical representations for probabilistic relationships among the variables given a set of discrete features. The learning process aims to find a Bayesian Network structure that describes the training data in the best possible way. Detailed descriptions of all the algorithms can be found in [5].

3.3 Traffic Employed

All four aforementioned systems require botnet traffic data while the first two systems (Packet payload based and Flow based) also require legitimate traffic given that they employ classification algorithms. In this work, several Zeus botnet traffic captures available at NETRESEC [3] and Snort[2] [2] web sites are employed. Hereafter, we refer to these two data sets as Zeus (Snort) and Zeus (NETRESEC). Moreover, since many researchers in the literature [18,12,16] employed generated botnet traffic in a sandbox environment using the public botnet binaries and toolkits, we also did the same using a public Zeus toolkit version 1.2.7.19. This toolkit is also analyzed and employed in [6]. We set up 12 Zeus bots (infected machines with Zeus botnet) and two C&C servers (one Windows server and one Linux server) in the test bed. Hereafter, we will refer to this data set as Zeus-2 (NIMS)[3]. Since the aforementioned three data sets are purely malicious and the systems based on various data mining techniques require legitimate traffic for training purposes, we employed a data set[3] representing normal behaviour used in [10].

Furthermore, University of Victoria also made a data set publicly available in 2013 [14]. This data set has combined two separate data sets of botnet malicious traffic from the French chapter of honeynet project on Strom and Waledac botnets. This combination represents the malicious side of the data set. On the other hand, their legitimate traffic is represented by two data sets: one data set from the Traffic Lab at Ericsson Research in Hungary and another data set from the Lawrence Berkeley National Laboratory (LBNL) in USA. Hereafter, we will refer to this data set as ISOT-UVic. This data set not only introduces different botnet traffic for our experiments but also introduces different normal traffic than the above.

Finally, CAIDA organization has also captured and made publicly available a Conficker botnet data set [15]. This three day captured traffic is collected by the CAIDA UCSD network telescope when the Conficker botnet was active. The first and the second day covers the Conficker-A botnet infection while during the

[2] "Sample_1" Zeus traffic file is used in this work.

[3] These data sets can be found at
`http://web.cs.dal.ca/~haddadi/data-analysis.htm`

third day Conficker-A and B were active. This data set has been anonymized, the payload has been removed from the packets and the CAIDA network addresses have been masked (destination IP addresses). A more detailed description of the CAIDA Conficker data set can be found in [15]. We included this data set in our evaluations, since it provides different botnet traffic for our evaluations. Hereafter, we will refer to this data set as the Conficker (CAIDA).

4 Evaluations and Results

As discussed earlier, our goal in this work is to evaluate different botnet detection systems where each uses specific parts of the traffic. To achieve this, we chose four different detection systems which are highly employed in academia and industry. These four systems are: packet payload based, flow based, BotHunter and Snort. For evaluation purposes, we have employed five different traffic traces for the botnets: Zeus (Snort), Zeus (NETRESEC), Zeus-2 (NIMS), ISOT-UVic and Conficker (CAIDA); and three different traffic traces for the legitimate traffic: Alexa, Ericsson and LNBL. Last but not the least, we have employed four different data mining techniques, namely C4.5, SVM, KNN and Bayesian Networks, for both the packet payload and the flow based systems.

4.1 Data Sets

Snort web site [2] officially has provided a description of the sample files and also has given information to use as the groundtruth for the data set. On the other hand, we analyzed the employed protocols, domain names and the communication patterns of Zeus (NETRESEC) traffic and compared them against the published characteristics of Zeus botnet to extract the malicious IP addresses. Regarding the Zeus-2 (NIMS) data set, since we set up the data generation environment in the laboratory, we had all the necessary information of the servers and the infected machines in this data set. Moreover, the information about the IP address mapping and the scenarios which were used to combine all four traffic traces from the ISOT-UVic data set can be found at [14]. Last but not the least, since UCSD telescope carries no legitimate traffic and given that there are other malicious background traffic than the Conficker infections in their captures, we used the information provided by CAIDA as the groundtruth for this data set.

For the data mining based systems, uniform sampling was used to create balanced (in terms of malicious vs non-malicious traffic) data sets for training purposes. For Conficker (CAIDA) data set specifically, we ensured that the training data set included data samples of each day so that behavioural examples of every version of Conficker traffic is represented in the data set. Having said this, there are no training samples (zero) for the Conficker (CAIDA) data set for the packet payload based approach. That is because this approach requires information from the packet payload while such information is not provided by CAIDA in this data set. It should be noted here that the traffic files that are used in the sampling process for the first two systems, are also employed for BotHunter and Snort. This provides consistency for our evaluations.

4.2 Performance Metrics

In traffic classification, two metrics are typically used in order to quantify the performance of the classifiers: Detection Rate (DR) and False Positive Rate (FPR). DR reflects the number of the correctly classified specific botnet samples in a given data set using $DR = \frac{TP}{TP+FN}$ where TP (True Positive) is the number of botnet traffic samples that are classified correctly, and FN (False Negative) is the number of botnet samples that are classified incorrectly (as legitimate samples). On the other hand, FPR shows the number of legitimate samples that are classified incorrectly as the botnet samples using $FPR = \frac{FP}{FP+TN}$ where TN (True Negative) is the number of legitimate traffic samples that are classified correctly.

4.3 Results

Packet Payload Based and Flow Based Systems: We employed the four data mining techniques (classifiers) using an open source tool called Weka [17] for the packet payload based and flow based systems. Our previous study on flow based botnet detection systems showed that C4.5 and Bayesian Networks are the best performed classifiers compared to Artificial Neural Networks, SVM and Naive Bayes [10]. Moreover, in this work, we added KNN to our evaluations since it was reported to give high performance in [11,19]. To evaluate these classifiers on the aforementioned data sets, we run them on each dataset using 10-fold cross-validation to further avoid any dataset biases that might affect the results. Table 2 and Table 3 shows the classification performances of these two systems on the five data sets. Considering the DR and FPR of the classifiers, C4.5 seems to be the best performing classifier that resulted in the detection rate of up to 100% for all five data sets. This classifier's output is in the form of rules that makes it easier to be used by a human expert to understand what this technique models on a given data set. For the packet payload based system, the size of the decision tree output is 5 for ISOT-UVic dataset and 7 for all other datasets. On the other hand, the size of the decision tree is 9, 21, 25, 7 and 525 for the Zeus (Snort), Zeus (NETRESEC), Zeus-2 (NIMS), Conficker (CAIDA) and ISOT-UVic, respectively, for the traffic flow based system. The results indicate that the flow based system solutions are bigger in size, even though the difference is not much in 4 of the 5 data sets. However, for the ISOT-UVic dataset, the complexity difference is significant. This can be caused by the fact that ISOT-UVic dataset is a combination of multiple botnet and legitimate datasets.

BotHunter: This tool provides Snort installation with a customized malware rule set from ET (Emerging Threats rule set and DNS/IP blacklist). Running BotHunter with a traffic pcap file (Batch mode) creates two types of results: BotHunter's Snort alerts (used as input for BotHunter correlator), and bot profiles. Moreover, to run BotHunter on any data set, the user needs to specify the trusted network or the monitored network. Indeed, such a requirement necessitates the users to have information about the data set or the monitored network (if using BotHunter in live mode).

Table 2. Classification results of the packet payload based system

	Data Set	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
C4.5	Zeus (Snort)	100%	100%	0%	100%	0%
	Zeus (NETRESEC)	100%	100%	0%	100%	0%
	Zeus-2 (NIMS)	99.44%	98.9%	0%	100%	1.1%
	Conficker (CAIDA)	-	-	-	-	-
	ISOT-UVic	99.77%	99.7%	0.1%	99.9%	0.3%
KNN	Zeus (Snort)	100%	100%	0%	100%	0%
	Zeus (NETRESEC)	100%	100%	0%	100%	0%
	Zeus-2 (NIMS)	100%	100%	0%	100%	0%
	Conficker (CAIDA)	-	-	-	-	-
	ISOT-UVic	99.72%	99.7%	0.3%	99.7%	0.3%
SVM	Zeus (Snort)	99.52%	99%	0%	100%	1.0%
	Zeus (NETRESEC)	100%	100%	0%	100%	0%
	Zeus-2 (NIMS)	99.72%	99.4%	0%	100%	0.6%
	Conficker (CAIDA)	-	-	-	-	-
	ISOT-UVic	99.79%	99.7%	0.1%	99.9%	0.3%
Bayesian Networks	Zeus (Snort)	100%	100%	0%	100%	0%
	Zeus (NETRESEC)	100%	100%	0%	100%	0%
	Zeus-2 (NIMS)	93.82%	98.9%	11.2%	88.8%	1.1%
	Conficker (CAIDA)	-	-	-	-	-
	ISOT-UVic	99.79%	99.7%	0.1%	99.9%	0.3%

Table 4 shows the results of BotHunter on the five data sets. The "# infected hosts" column in the table shows the number of infected machines with the bot program. The "# remote hosts" shows the malicious remote machines that the infected hosts communicate with in the captured data sets. Although finding the infected host in the network is important, it is only one phase of the detection. However, finding the source of the attacks or at least the remote hosts that are utilized by the C&C server is another important phase of detection. Hence, in this work, we also analyzed the remote host information. These remote machines can be the malicious C&C servers or new targets of the botnet that the infected machine aims to infect. BotHunter correlates the Snort alerts (shown in the second column) and finally generates the bot profiles revealing the malicious hosts. In the cells of the Table 4 where two numbers are separated by "/", the first number is the count of IP addresses detected and the second number is the total number of IP addresses in that category. In each cell, the DR for each category is included. However, the overall detection rate of the BotHunter including the infected hosts and the remote hosts is shown in Table 6.

In short, based on the performance of BotHunter presented in Tables 4 and 6, we make the following observations: (1) No alert or bot profile was raised for Conficker (CAIDA) data set. That is because the payload part of the traffic was not provided by CAIDA. Given that BotHunter and its Snort sensors use payload of the traffic (packets) for detecting the botnets, they could not perform

Table 3. Classification results for the traffic flow based system

	Data Set	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
C4.5	Zeus (Snort)	98.9%	97.2%	1.4%	98.6%	2.8%
	Zeus (NETRESEC)	98.25%	98.5%	2%	98%	1.5%
	Zeus-2 (NIMS)	99.67%	99.8%	0.5%	99.5%	0.2%
	Conficker (CAIDA)	99.95%	100%	0.1%	99.9%	0%
	ISOT-UVic	99.9%	99.9%	0.1%	99.9%	0.1%
KNN	Zeus (Snort)	98.6%	99.3%	2.1%	97.9%	0.7%
	Zeus (NETRESEC)	96.75%	97.0%	3.5%	96.5%	3.0%
	Zeus-2 (NIMS)	99.74%	99.8%	0.3%	99.7%	0.2%
	Conficker (CAIDA)	99.95%	99.9%	0%	100%	0.1%
	ISOT-UVic	99.91%	99.8%	0%	100%	0.2%
SVM	Zeus (Snort)	99.3%	100%	1.4%	98.6%	0%
	Zeus (NETRESEC)	91.01%	89.3%	7.3%	92.8%	10.7%
	Zeus-2 (NIMS)	99.90%	99.6%	1.8%	98.2%	0.4%
	Conficker (CAIDA)	99.89%	99.9%	0.1%	99.9%	0.1%
	ISOT-UVic	99.82%	99.8%	0.2%	99.8%	0.2%
Bayesian Networks	Zeus (Snort)	97.21%	100%	5.6%	94.4%	0%
	Zeus (NETRESEC)	83.89%	76.8%	9.0%	91.0%	23.2%
	Zeus-2 (NIMS)	99.61%	99.6%	0.4%	99.6%	0.4%
	Conficker (CAIDA)	98.47%	99.9%	3.0%	97.0%	0.1%
	ISOT-UVic	99.86%	99.9%	0.2%	99.8%	0.1%

well on this data set. (2) BotHunter could successfully detect all the infected machines and remote hosts of the Zeus-2 (NIMS) data set. That is because payload is provided and all the phases of the botnet lifecycle are present in this data set. (3) Although Snort did create serious alarms on Zeus (Snort) (such as "E4[rb] TROJAN Zeus POST Request to CnC"), BotHunter did not report any bot profile. This could be because a combination of different types of alerts (representing different phases of lifecycle) is required by BotHunter to form a bot profile. (4) For ISOT-UVic data set, BotHunter could successfully detect four infected hosts out of five known ones. However, from almost 15000 remote hosts with which these infected machines communicate, only 40 of them were identified in the generated bot profiles. Again, we think that this was because BotHunter: (i) requires all the phases of the botnet lifecycle for all the machines; and (ii) depends on Snort sensor rule set.

Snort: To run Snort, the first thing required is to set the rule set that will be used. Two main rule sets for Snort are VRT and ET. In our evaluations, we used the VRT rule set because it is the official rule set of Snort that gets updated frequently. Furthermore,the other rule set, ET, is the one used by BotHunter in our evaluations. In our analysis of Snort, we found that many rules of the VRT rule set are disabled and specifically all pre-processor and shared-object rules are disabled by default. Based on the information Snort has provided on botnet detection, we enabled all of the rules such as (*sid= 16460 and 11192*) that are

Table 4. BotHunter detailed detection results

Data Set	Snort alerts	# infected hosts	# remote hosts
Zeus (Snort)	26	0/1 (0%)	0/14 (0%))
Zeus (NETRESEC)	23	1/1 (100%)	7/11 (63.6%)
Zeus-2 (NIMS)	486	12/12 (100%)	2/2 (100%)
Conficker (CAIDA)	0	0 (0%)	0 (0%)
ISOT-UVic	831	4/5 (80%)	40/15000 (0.3%)

Table 5. Snort detailed detection results

Data Set	Snort alerts	# infected hosts	# remote hosts
Zeus (Snort)	11	1/1 (100%)	7/14 (50%)
Zeus (NETRESEC)	58	1/1 (100%)	7/11 (63.6%)
Zeus-2 (NIMS)	401	12/12 (100%)	2/2(100%)
Conficker (CAIDA)	7244	6457/360191 (1.8%)	430/80380 (0.5%)
ISOT-UVic	102755	2/5 (40%)	2326/15000 (15.5%)

Table 6. BotHunter and Snort overall performances

	Data Set	DR	TPR	FNR
BotHunter	Zeus (Snort)	0%	0%	0%
	Zeus (NETRESEC)	66.6%	66.6%	0%
	Zeus-2 (NIMS)	100%	100%	0%
	Conficker (CAIDA)	0%	0%	0%
	ISOT-UVic	2.9%	2.9%	0%
Snort	Zeus (Snort)	53.3%	53.3%	0%
	Zeus (NETRESEC)	66.6%	66.6%	0%
	Zeus-2 (NIMS)	100%	100%	0%
	Conficker (CAIDA)	1.6%	1.6%	0%
	ISOT-UVic	15.5%	15.5%	0%(6/23140)

related to Zeus botnet. Tables 5 and 6 show the performance of Snort on our five data sets. For big data sets that contain considerable number of malicious traffic, Snort raises a lot of alerts. This makes it complicated to process the results (such as for Conficker CAIDA and ISOT-UVic data sets shown in Table 5). Thus, in this work, any alert with high priority that was raised on botnet IP addresses (such as alerts indicating "Win.Trojan.Zeus") is considered as TP, and any such alert triggered on legitimate IP addresses is considered as FP.

Table 7 shows detailed information on the type of rules that Snort and BotHunter employed for all of the five data sets. The "Header+payload based" column shows the rules that require both the header and the payload of the packet traces whereas the "Header based" rules are the ones that use only the packet header information. The first number in each category shows the number of rules triggered and the second number in parenthesis shows the number of alerts

Table 7. The number and the type of Snort and BotHunter Rules

Data Set	Snort		BotHunter	
	Header based	Header+payload based	Header	Header+payload based
Zeus (Snort)	2 (9)	1 (2)	0 (0)	2 (26)
Zeus (NETRESEC)	3 (38)	8 (20)	0 (0)	4 (23)
Zeus-2 (NIMS)	2 (202)	1 (199)	0 (0)	4 (486)
Conficker (CAIDA)	1 (7244)	0 (0)	0 (0)	0 (0)
ISOT-UVic	3 (102755)	0 (0)	0 (0)	9 (831)

generated in that category. As indicated in the table, BotHunter only used the Snort rules (ET) that are based on both the header and the payload information while Snort utilized more of the header based rules from the VRT rule set. In the "Header+payload" category the "MALWARE-CNC" type rules and in the "Header" category the "CONTENT-LENGTH" type rules are frequently used by Snort. On the other hand, BotHunter employed the "E4[rb] " and "E4[rb]" type rules more frequently. The results also shows that when almost all phases of communication are available in the data set (as for the Zeus-2 (NIMS)), the number of alerts being triggered is increased as well as the DR.

Finally, it should be noted that all the tools and the data sets are publically available and the experiments can be repeated to be compared against other approaches.

4.4 Discussion and Highlights

The main advantage of the first two systems based on data mining (packet payload based and flow based) is the ability to automatically discover patterns in big traffic data sets. This also provides the capability of detecting malicious communications at any stage of the botnet lifecycle without focusing on one side of the network (as BotHunter does). The performances of these two systems indicate that both feature extraction techniques can be used to build botnet detection models with high performances. However, given that the packet payload based approach requires the payload information of the packets for analysis and this information may not be available due to encryption or simply not captured, we believe that the flow based detection system is the winner among the two. As demonstrated in our evaluations, the performance of the flow based system is higher or similar to the results reported in the literature (with detection rate of up to 100%, up to 79%, 88% and 99% in [8], [12], [18] and [20], respectively.

On the other hand, BotHunter focuses on finding the Snort alerts corresponding to the botnet lifecycle and correlating them to create a bot profile. Our evaluations show that it cannot detect the botnet related malicious communications on the network if it cannot find the necessary phases of the botnet lifecycle in the traffic. However, BotHunter seems to be successful when a specific network is under constant monitoring and the goal is to detect the infected machines of

a trusted network. Constant monitoring helps the Snort sensors to detect all the phases of a botnet lifecycle. Having said this, it uses the pre-defined customized Snort rule set so it cannot correlate different behaviours if they are not detected by Snort.

Unlike BotHunter, Snort does not just focus on detecting the infected hosts based on tracing the existence of the botnet lifecycle. Instead, it monitors all the network communications (the default HOME_NET is"any") and flags any suspicious communication that matches its pre-defined rules if using the VRT rule set. The performance of Snort depends on the quality of the rules. Since Snort is a well-known IDS/IPS, its rule set gets updated frequently. This makes it one of the more popular detection systems available today. However, as we observed in our evaluations, it generates a lot of alerts.

5 Conclusions

In this work, four botnet detection systems are investigated. Each one of these uses specific features from network traffic with different levels of human involvement. The first system is a packet payload based system, which employs classifiers using the features extracted from the header and the payload of a packet. The second system is a traffic flow based system where features are extracted on a per flow basis instead of packets. Since the features used by this system are extracted from only the header section of the packets, this approach can be applied to encrypted traffic as well. In addition to these two systems, Snort and BotHunter are also evaluated as publicly available botnet detection systems. These two systems represent rule based detection systems where both the packet headers and the packet payloads are analyzed. The evaluation of all four systems on five public data sets show that the first two systems performed better than the last two systems with detection rates approaching up to 100% on some of the data sets. Comparing the payload based and flow based systems, neither of them significantly out-performed the other and they both achieve very similar detection performances (highest when C4.5 was used in both cases). Having said this, the packet payload based system results in much lower false positive rates. This makes it very desirable on all data sets except the ones that do not have any payload. In those cases, this system cannot perform at all. However, the flow based system can still perform. This gives the flow based method an important advantage.

Acknowledgments. This research is supported by the Canadian Safety and Security Program(CSSP) E-Security grant. The CSSP is led by the Defense Research and Development Canada, Centre for Security Science (CSS) on behalf of the Government of Canada and its partners across all levels of government, response and emergency management organizations, nongovernmental agencies, industry and academia.

References

1. https://www.snort.org/
2. https://labs.snort.org/papers/zeus.html
3. NETRESEC repository: publicly available pcap files, http://www.netresec.com/?page=PcapFiles.
4. Tranalyzer, http://tranalyzer.com/
5. Alpaydin, E.: Introduction to Machine Learning. MIT Press (2004)
6. Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the zeus botnet crimeware toolkit. In: PST (2010)
7. Celik, Z.B., Raghuram, J., Kesidis, G., Miller, D.J.: Salting public traces with attack traffic to test flow classifiers. In: CSET (2011)
8. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: Bothunter: detecting malware infection through ids-driven dialog correlation. In: 16th USENIX Security Symposium (2007)
9. Haddadi, F., Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I.: Malicious automatically generated domain name detection using stateful-SBB. In: Esparcia-Alcázar, A.I. (ed.) EvoApplications 2013. LNCS, vol. 7835, pp. 529–539. Springer, Heidelberg (2013)
10. Haddadi, F., Zincir-Heywood, A.N.: Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. IEEE Systems Journal, 1–12 (2014)
11. Mohaisen, A., Alrawi, O.: Unveiling Zeus. In: IW3C2 (2013)
12. Perdisci, R., Corona, I., Dagon, D., Lee, W.: Detecting malicious flux service networks through passive analysis of recursive DNS traces. In: ACSAC (2009)
13. RFC 2722 (October 1999), http://tools.ietf.org/html/rfc2722
14. Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W., Fleix, J., Hakimian, P.: Detecting P2P botnets through network behavior analysis and machine learning. In: PST (2011)
15. The CAIDA USCD Network Telescope- 'Three Days of Conficker', http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml
16. Wang, K., Huang, C., Lin, S., Lin, Y.: A fuzzy pattern-based filtering algorithm for botnet detection. Computer Networks 55, 3275–3286 (2011)
17. weka, http://www.cs.waikato.ac.nz/ml/weka/
18. 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)
19. Zhang, J., Chen, C., Xiang, Y., Zhou, W., Vasilakos, A.: An effective network classification method with unknown flow detection. IEEE Transactions on Network and Service Management 10 (2013)
20. Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., Garant, D.: Botnet detection based on traffic behavior analysis and flow intervals. Computers and Security 39 (2013)