A Closer look at the HTTP and P2P based Botnets from a Detector's Perspective

Fariba Haddadi and A. Nur Zincir-Heywood

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

Abstract. Botnets are one of the main aggressive threats against cybersecurity. To evade the detection systems, recent botnets use the most common communication protocols on the Internet to hide themselves in the legitimate users traffic. From this perspective, most recent botnets are HTTP based and/or Peer-to-Peer (P2P) systems. In this work, we investigate whether such structural differences have any impact on the performance of the botnet detection systems. To this end, we studied the differences of three machine learning techniques (Decision Tree, Genetic Programming and Bayesian Networks). The investigated approaches have been previously shown effective for HTTP based botnets. We also analyze the detection models in detail to highlight any behavioural differences between these two types of botnets. In our analysis, we employed four HTTP based publicly available botnet data sets (namely Citadel, Zeus, Conficker and Virut) and four P2P based publicly available botnet data sets (namely ISOT, NSIS, ZeroAccess and Kelihos).

Key words: Botnet detection, HTTP, P2P, Machine learning.

1 Introduction

In the past couple of years, parties involved in information technology have experienced enormous growth in cybersecurity. Technology as one side of this domain has changed at an outstanding rate and threats, on the other hand, have evolved significantly. In this scope, botnets are considered as one of the most aggressive threats that are responsible for a large volume of malicious activities.

A botnet is a network of compromised hosts (i.e. bots) that are under the remote control of an offender, called botmaster. Bots are unwillingly and unknowingly utilized by the botmaster to carry out a diverse range of malicious activities from the distributed-denial-of-service (DDoS) attacks to identity thefts and spamming. On the other hand, masters have utilized various protocols (e.g. IRC, HTTP and DNS), topologies (i.e. centralized and de-centralized) and techniques (e.g. encryption and fluxing) since 2003. Such diverse structures have assisted the botmasters to evolve and defeat the detection systems of this field.

Botmaster forms the botnet through five phases called the initial infection, the secondary injection, the connection, the malicious C&C (Command and Control) and the maintenance and update phases. Since a botnet can upgrade its structure (or any algorithm that is used by botnets) in the fifth phase of the lifecycle, automatic pattern discovery could potentially enable security systems to adapt to such changes in the botnet evolution. In this case, many detection approaches have been proposed based on various machine learning (ML) techniques. These approaches can be surveyed from the data perspective (i.e. the type of network data being analyzed and its representation) and the ML algorithms employed. From the data perspective, network traces should be represented to the algorithms through feature sets. Many of the approaches analyzed specific parts/packets of the network trace such as DNS queries, HTTP requests and their corresponding responses. Others utilized the flow[1] definition to aggregate discrete network packets into a collection for the analysis purposes. 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. To be able to analyze communication information such as the domain names, payload information should be available in clear text. However, given that most recent botnets utilize encryption methods to encode the communication information, detection systems should be able decrypt the information. Using such approaches may not be practical given that the decryption process increases the computing complexity significantly. Moreover, the encryption methods/algorithms can be modified/changed on the fly. On the other hand, flow-based detection approaches can be very much useful since the features are only extracted from the packet header.

In our previous work, we have proposed to employ the flow features of the Tranalyzer flow exporter for botnet detection purposes. The effectiveness of this feature set was evaluated on eight HTTP-based botnets using ML techniques. However, botnets with the P2P topology (which may/may not use HTTP as their communication protocol) are also among the most recent aggressive types of botnets. Hence, in this work, we aim to evaluate the proposed approach as well as the suggested feature set against the P2P botnets. We also investigate whether the suggested approach can be as effective on the P2P botnets as it was on HTTP-based botnets. To this end, we use Tranalyzer [1] to extract the flow features from four P2P botnet data sets as well as four HTTP botnet data sets. Three ML algorithms (C4.5 decision tree, Bayesian Networks and the Symbiotic Bid-Based Genetic Programming (SBB)) are employed to build the detection system models. Furthermore, beside evaluating the effectiveness of the proposed approach, we also aim to analyze the trained detection models in order to highlight the behavioural differences of the HTTP and the P2P

[1] Flow is defined as a logical equivalent for a call or a connection in association with a user specified group of elements [14]. 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.

botnets. Last but not the least, a multi-class classification approach is designed to investigate whether each botnet would have enough discrepancy from the classifier's perspective to be distinguished from the others.

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

2 Background and Related Work

A bot program is a self-propagating malware that infects vulnerable hosts known as bots (zombies) and is designed to perform specific malicious tasks after being triggered. Hosts can get infected with malwares in different ways such as visiting an untrusted malicious website or opening a malicious email attachment. Usually bots receive commands from the master through a communication server and carry out malicious tasks such as Distributed Denial of Service (DDoS), spamming, phishing and identity theft attacks [5, 15].

Botnet architecture is categorized in different ways. In one method, centralized and de-centralized are considered as the two main categories of botnets. In the centralized model, command and data exchange between the master and bots is managed at the central C&C server. The C&C server uses different services/protocols to manage the botnet. IRC, HTTP and DNS are the most common protocols in this architecture, which is based on a client-server scheme. Easy implementation of the centralized communication channels and low latency are the two main advantages of this structure. The low latency is caused by the clear connections between the clients and the C&C servers. This feature is very important for the malicious tasks such as DDoS, which require the bots and the servers to be highly synchronized. However, the main disadvantage of this architecture is the single point of failure issue that is caused by the C&C servers. In this case, if the C&C servers are discovered by the botnet detection systems, the whole botnet can then be taken down. Having said this, discovering the botnets with centralized architecture is relatively easier since all the connections are going through specific C&C servers.

On the other hand, in the de-centralized category, botnets either use the P2P architecture or utilize techniques such as fluxing with the C&C based communication servers to compensate the characteristics of the centralized architecture. In the P2P structure, bots can act as both clients and servers. They do not contact any specific server for commands directly but receive the commands from their peers. Botnets with the de-centralized architecture are more resilient than centralized botnets. In other words, discovering and removing a bot from a P2P botnet or a C&C server from the list of all possible servers hardly affect the botnet mission. Moreover, analyzing botnets behaviour with the de-centralized architecture and detecting them is more complicated because of the distributed structure. However, the implementation complexity and latency of botnets in this category is notable comparing to the centralized structure.

The HTTP botnets and P2P botnets (which may or may not use HTTP as their communication protocol) are considered as the two main recent types of botnets. Hence, in this work, we aim to investigate these types of botnets in terms of any discrepancy in designing the detection systems and their performances. Wurzinger et al. proposed an approach to detect botnets based on the correlation of commands and responses in the monitored network traces [17]. 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. Data sets used in this work were collected by running bot binaries in a controlled environment. Traffic features such as the number of non-ASCII bytes in the payload were analyzed to characterize bot behavior. Zhang et al. proposed a botnet detection system to identify P2P botnets based on a two phase analysis of the traffic [18]. In the first phase, P2P hosts are identified regardless of being malicious or legitimate. To do this, Netflow features are extracted and then filtered based on the IP addresses associated with resolved DNS responses. In the second phase, the remaining flows are analyzed to differentiate the P2P botnet traffic from other legitimate P2P traffic. Wang et al. proposed a fuzzy pattern recognition approach (called BBDP) to detect HTTP and IRC botnet 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 botnet malicious domain names and IP addresses. To accelerate the detection process and be able to detect botnets in real-time, traffic reduction and parallel processing were utilized. Their results showed up to 95% detection rate for their system. Kirubavathi et al. designed specifically an HTTP-based botnet detection system using a multilayer Feed-Forward Neural network [12]. Given that HTTP-based botnets do not maintain a connection with the C&C server but periodically make a request to the C&C server (over the HTTP) to download the instructions, they extracted features related to TCP connections in specific time intervals based on the packet headers. To collect data to evaluate their system, botnets were simulated in the lab. Zhao et al. investigated a botnet detection system based on flow intervals [19]. Flow features of traffic packets were utilized with several ML algorithms where the decision tree classifier was finally selected as the preferred classifier to detect botnets. They focused on P2P botnets (such as Waledac) that employ the HTTP protocol and a fast-flux based DNS technique. Their proposed detection approach resulted in up to 99% detection rates with false positive rate around 2%. Beigi et al. investigated the effectiveness of flow-based feature sets employed in previous botnet detection studies and evaluated them using their own feature selection algorithm [3]. Their results indicated that the Byte-based group of features has less effect while the packet-based group has more impact. In their evaluation, IRC, HTTP and P2P botnet data sets were utilized. In our previous work, a machine learning based detection system was designed and evaluated on several HTTP botnets [10]. In addition to different machine learning

algorithms, five flow feature sets were benchmarked and investigated in detail. The result of those analysis showed that the C4.5 classifier can detect HTTP based botnets using Tranalyzer feature set with up to 100% detection rate. The proposed approach was also compared against the Snort intrusion detection system, a machine learning packet based system and BotHunter botnet detection system [8]. The comparison results showed that the approach outperforms the other three detection systems considering several performance criteria such as complexity and detection rate. Moreover, the performance of the proposed approach was evaluated over a period of time. The results indicated that not only the approach performs well facing the new versions of the botnet it is trained for but also, it can identify the change of topology by observing the changes in the performance [11].

In summary, some systems/approaches [16, 17] require both the payload and the header section of the packets to extract the necessary features while others [19, 10] only need the header of the packets (e.g. flow based systems). The importance of the approaches in the second group can be better understood knowing that the most recent aggressive botnets employ encryption to better hide themselves and their information from the detection systems. Our proposed approach stands in this group. Moreover, there are several studies on flow based botnet detection systems where each proposed their own set of features [18, 19]. Some studies have analyzed the feature selection algorithms to extract the most effective feature sets [3]. Such feature selection processes can cause the models to be focused on specific type(s) of botnet(s) which may not be very effective for other types. Hence, using a ML algorithm that has the ability to perform attribute selection as an implicit property of constructing the classifier may be a better way to approach feature selection while utilizing all the possible extracted flow features. This is the main idea behind our previously proposed approach [10].

3 Methodology

As discussed in section 1, flow based botnet detection systems are beneficial since most of the recent botnet traffic communications (placed in the packet payloads) are encrypted and such systems do not require payload information at all. On the other hand, using ML based techniques to build the detection system potentially provides the system with the ability to cope with the botnet upgrades using minimum apriori knowledge. In our previous work, we benchmarked five flow feature sets and five ML techniques in order to find the best performing combination [10]. Although the combination of Tranalyzer feature set with the C4.5 classifier was proposed as the final and best combination, the evaluation was only done on eight HTTP based botnets. Given that in addition to HTTP botnets, P2P botnets are also among the most recent destructive active botnets, we aim to examine the proposed system on the P2P architectures as well. To this end, four HTTP botnets and four P2P botnets are collected and three best-performed classifiers from our previous works [9, 10] (i.e. C4.5 decision tree, Bayesian Networks and SBB) are selected and evaluated in this

research. Moreover, for the feature extraction purpose, Tranalyzer flow exporter is utilized. Finally, additional experiments and model analysis are performed to reveal the behavioural differences between these two categories of botnets.

3.1 Learning algorithms

The candidate ML algorithms in this work are C4.5 decision tree, Bayesian Networks and SBB.

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. In other words, the attribute with the highest information gain value is chosen as the splitting point. A more detailed explanation of the algorithm can be found in [2].

Bayesian Networks are graphical representations for probabilistic relationships among the variables given a set of discrete features. The graph nodes that are associated with the attributes, are connected through the links that correspond to the direct influence from one feature to the other. Given the Bayesian networks' structure, the conditional probability distribution of the graph is then computed. The learning process aims to find a Bayesian Network structure that describes the training data in the best possible way. Detailed explanation on Bayesian Networks can be found in [2].

SBB is a form of linear genetic programming with a co-evolutionary architecture [13]. Three populations are co-evolved in this algorithm: A point population, a team population and a learner population. The learner population represents a set of learners, which associate a GP-bidding behaviour with an action. The team population comprises a set of learners and finally the point population denotes a subset of training data exemplars. Evaluating a team on the points, all of the team's learner programs are executed while only the learner with the highest bid suggests its action as the team's action. The bidding procedure employs linear GP in addition to a sigmoid function to standardize the bid values between zero and one.

3.2 Traffic employed

In this paper, eight publically available botnet traffic traces are employed for evaluation while four of them represent HTTP botnets behaviour and the rest represents P2P botnets behaviour. To the best of our knowledge, the P2P botnet data sets employed in this research are the only ones that are publicly available at this time. Furthermore, the HTTP botnet data sets are not only publicly

available but also are different from the data sets used in our previous work [10]. By choosing different data sets in this work, we intend to explore and evaluate how much the performance of the proposed approach depends on the data sets and how well it generalizes.

Conficker (CAIDA), Zeus (NIMS), Citadel (NIMS), Virut (CVUT) are the sample data sets of the HTTP category. Conficker (CAIDA) is the Conficker botnet data set collected and published by the CAIDA organization. The data set is a three-day capture of Conficker version A and B which is anonymized. In other words, the payload information is removed and the CAIDA network addresses are masked. A more detailed description of the CAIDA Conficker data set can be found at [4]. Moreover, Zeus (NIMS) and Citadel (NIMS) are the Zeus and Citadel botnet data sets that are generated in the NIMS[2] lab sandbox. To generate these botnet traffic traces, Zeus botnet toolkit version 2.1.0.1 and Citadel botnet toolkit version 1.3.5.1 are utilized. In the sandbox testbed, 12 windows bots and one C&C server are configured and set up. Finally, Malware capture facility Project at the Czech Technical University in Prague (CVUT) have collected several malware traffic logs [6]. Virut (CVUT) is one of the data sets from this collection. Detailed information of this capture can be found at [7].

On the other hand, NSIS (CVUT), ZeroAccess (CVUT), Kelihos (CVUT) and ISOT (Uvic) are representative of P2P botnets used in this work. NSIS (CVUT), ZeroAccess (CVUT) and Kelihos (CVUT) are from the CVUT malware capture facility project [6] and ISOT (Uvic) is made publically available by the University of Victoria [19]. ISOT (Uvic) 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 of two botnets traffic represents the malicious side of the ISOT (Uvic) data sets.

In order to differentiate botnet behaviour from legitimate behaviour, a data set must include legitimate data samples, representing legitimate behaviours. In this case, ISOT (UVIC) data set includes traffic traces from two legitimate resources, the traffic Lab at Ericsson Research in Hungary and the Lawrence Berkeley National Laboratory (LBNL) in USA, to be representative of legitimate behaviours. However, all other seven data sets either do not have any legitimate traffic included or include some background traffic which may or may not represent legitimate behaviours. Given that other researchers in the literature have also used the LBNL traffic traces to represent the normal behaviour, we utilized and combined these traffic traces with the other seven data sets (all except the ISOT (Uvic)). Therefore, all of the data sets employed in this work share the same type of legitimate behaviour. Since we are not using the source/destination IP addresses and source/destination port numbers, different ranges of these four features will not affect the results.

ML approaches require the training and testing data sets to be presented by feature sets. However, network traces are formed by network packets and there-

[2] Network Information Management and Security: https://projects.cs.dal.ca/projectx/

fore, need to be processed in order to be represented by features. Although fields of the network packets can be utilized as features (in packet based detection approaches), aggregating the network packets into flows and extracting the flow features has been shown effective in the recent literature. Hence, in this research, Tranalyzer flow exporter is employed in the feature extraction phase. In general, 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 lightweight uni-directional flow exporter that employs an extended version of NetFlow feature set to support 93 flow features. More detailed information on the tool and its feature set can be found in [1, 10].

4 Evaluation and results

As discussed earlier, in this paper, we aim to investigate any differences in terms of detection performances and solutions between HTTP and P2P botnets as well as the generalization of the approach from one data set to another. To this end, eight botnet data sets (four in each category) are evaluated using three ML algorithms, namely C4.5, Bayesian networks and SBB.

To prepare the data sets, Conficker (CAIDA), Kelihos (CVUT), NSIS (CVUT), Virut (CVUT), ZeroAccess (CVUT), Citadel (NIMS) and Zeus (NIMS) network traces are combined with LNBL legitimate network traces similar to the approaches used in the literature [19]. Tranalyzer flow exporter is then employed to extract the features and finally, uniform sampling was used to create balanced (in terms of malicious vs non-malicious samples) data sets for training purposes. We employed all of the numeric features provided by the Tranalyzer as inputs to the ML classifiers except the IP addresses and port numbers. The reasons behind this are: IP addresses can be anonymized whereas port numbers can be assigned dynamically. Thus, employing such features may decrease the generalization abilities of the detection systems for unseen behaviors. Table 1 shows the number of flow samples of all the eight data sets in addition to the original size of the data sets (traffic trace files before adding the LBNL traces).

4.1 Performance metrics

Performance 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.

Table 1: Specification of the data sets employed

Data set	Size	Sample count (# of flows)
Conficker (CAIDA)	183GB	4135673
Kelihos (CVUT)	409MB	1098448
NSIS.ay (CVUT)	281MB	26294
ZeroAccess (CVUT)	59.2MB	214442
Virut (CVUT)	109MB	305664
Citadel (NIMS)	40.4MB	12662
Zeus (NIMS)	18.7MB	21356
ISOT (Uvic)	10.6GB	197462

Complexity Classifier complexity can be measured by different criteria such as memory consumption, time or the learned model by the learning algorithms. In this work, two complexity criteria are utilized: **1) training (computation) time** is employed where this is estimated on a common computing platform. **2) solution complexity**, is measured using the tree size for C4.5 and the program size of the solution team for SBB. It should be noted here that a direct comparison between solutions from different representations is impossible since the underlying units of measurement are different in different ML algorithms.

4.2 Results

Table 2 shows the results of the three classifiers on all four HTTP data sets. Comparing the results of the three classifiers on the HTTP botnets, C4.5 and SBB performed the same. To this end, we expand our evaluations over time and solution complexity. Table 3 shows that there is no consistent pattern in the time complexity as C4.5 had the highest and the lowest training time of the table on Conficker (CAIDA) and Citadel (NIMS) data sets, respectively. However, in terms of solution complexity, SBB consistently provides solutions with lower complexity. This confirms the results of our previous work on HTTP botnets [9].

On the other hand, Table 4 shows the evaluation results of the P2P data sets. In this case, C4.5 outperformed the other two classifiers. Unlike our experiments on HTTP botnets, SBB could not keep up with C4.5 and showed the lowest DR and the highest FPR. However, complexity analysis (Table 5) resulted in the same pattern as our HTTP botnets, indicating that SBB offers solutions with lower complexity.

Although SBB finds solutions with lower complexity in both types of botnets (HTTP and P2P), a high performance is still the primary goal in malware detection systems. Hence, SBB's lower performance on P2P botnets makes C4.5 the desirable classifier in this work. Comparing the C4.5 classification results of the HTTP and the P2P botnets, the classifier performed equally well on both types of botnet. Hence, in terms of detection capability and performance, we can conclude that our ML detection approach (combination of Tranalyzer flow

Table 2: Classification results of the HTTP botnets

	Data Set	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
C4.5	Zeus (NIMS)	99.93%	99.9%	0.1%	100%	0.1%
	Citadel (NIMS)	99.90%	99.9%	0.1%	99.9%	0.1%
	Conficker (CAIDA)	99.95%	100%	0.1%	99.9%	0%
	Virut (CVUT)	99.88%	99.9%	0.1%	99.9%	0.1%
Bayesian Networks	Zeus (NIMS)	98.45%	96.9%	0%	100%	3.1%
	Citadel (NIMS)	98.76%	97.5%	0%	100%	2.5%
	Conficker (CAIDA)	98.62%	99.1%	1.9%	98.1%	0.9%
	Virut (CVUT)	94.64%	94.5%	5.2%	94.8%	5.5%
SBB	Zeus (NIMS)	99.97%	99.94%	0%	100%	0.06%
	Citadel (NIMS)	100%	100%	0%	100%	0%
	Conficker (CAIDA)	99.06%	99.01%	0.9%	99.01%	0.9%
	Virut (CVUT)	98.29%	98.25%	1.77%	98.33%	1.75%

Table 3: Complexity analysis of the classifiers on the HTTP botnets

	Data Set	Time Complexity (sec)	Solution Complexity
C4.5	Zeus (NIMS)	2.11	41
	Citadel (NIMS)	1.06	31
	Conficker (CAIDA)	7454.18	1317
	Virut (CVUT)	215	481
SBB	Zeus (NIMS)	279.6	38
	Citadel (NIMS)	295.55	30
	Conficker (CAIDA)	235.439	58
	Virut (CVUT)	214.76	17

exporter and C4.5 classifier), which was previously suggested for HTTP botnets, can be a valid choice for the P2P botnet detection systems as well. Moreover, some of the data sets employed in this paper are used by other researchers in the literature. For example, Zhao et al. achieved DRs between 97.9 to 99.9% on the ISOT (Uvic) data set [19] and Beigi et al. obtained DRs between 75% to 99% on different combination of data sets including ISOT (Uvic), Virut (CVUT), ZeroAccess (CVUT) and NSIS (CVUT) [3]. This shows the performance achieved by our ML detection approach is not only comparable to other approaches in the literature but also outperformed those approaches in some cases.

Comparing the C4.5 complexity results, we notice higher complexity (time and solution) on the P2P botnets. However, this might be very much caused by the differences of the botnets' behaviour or even the data sets with different number of samples. Hence, to shed more light into this, we analyzed and compared the trained C4.5 classification models to understand if there are any obvious characteristics in the models that can specifically point out the differences of the P2P and the HTTP botnets. Table 7 shows the top features (on the first

Table 4: Classification results of the P2P botnets

	Data Set	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
C4.5	ZeroAccess (CVUT)	99.94%	100%	0.1%	99.9%	0%
	Kelihos (CVUT)	99.93%	99.9%	0.1%	99.9%	0.1%
	NSIS (CVUT)	99.23%	99.3%	0.8%	99.2%	0.7%
	ISOT (Uvic)	99.83%	99.8%	0.2%	99.8%	0.2%
Bayesian Networks	ZeroAccess (CVUT)	99.29%	99.7%	1.1%	98.9%	0.3%
	Kelihos (CVUT)	93.16%	93.8%	7.4%	92.6%	6.2%
	NSIS (CVUT)	95.94%	93.6%	1.8%	98.2%	6.4%
	ISOT (Uvic)	96.49%	99.2%	6.2%	93.8%	0.8%
SBB	ZeroAccess (CVUT)	99.39%	99.63%	0.84%	99.16%	0.37%
	Kelihos (CVUT)	97.74%	99.19%	3.7%	96.29%	0.8%
	NSIS (CVUT)	94.09%	92.11%	3.9%	96.1%	7.8%
	ISOT (Uvic)	93.12%	97.36%	11.11%	88.89%	2.64%

Table 5: Complexity analysis of the classifiers on the P2P botnets

	Data Set	Time Complexity (sec)	Solution Complexity
SBB	ZeroAccess (CVUT)	279.6	38
	Kelihos (CVUT)	295.55	30
	NSIS (CVUT)	235.439	58
	ISOT (Uvic)	214.76	17
C4.5	ZeroAccess (CVUT)	94.85	135
	Kelihos (CVUT)	1149.57	849
	NSIS (CVUT)	5.38	275
	ISOT (Uvic)	82.57	525

three levels of the trees) of the C4.5 decision trees with the highest information gain. The description of the features named in this work can be found in Table 6 while the description of all the features supported by Tranalyzer are available at [1]. This analysis indicates that: (i) Packet-based and Byte-based features (such as minPktSz or UppQuartilePl) are used by C4.5 for both types of botnets. (ii) Various connection-based features (such as ConnSrcDst or ConnSrc) are utilized for all of the P2P botnets. (iii) TTL-based features (such as ipMinTTL) are mostly used by the P2P botnets. (iv) Inter-arrival based features are only used for the HTTP botnets.

Going one step further, we also analyzed the features that are most frequently utilized by C4.5 in Table 8. The analysis also confirms that the packet-based and the byte-based features are frequently used for both types of botnets. On the other hand, the inter-arrival based features are only selected and used for the HTTP botnets. Finally, the connection-based features are utilized for the P2P botnets more than the HTTP botnets. However, unlike our observation on features with the highest information gain, TTL-based features are used for both

Table 6: Brief description of some of the Tranalyzers' features

Feature	Description
connSrc	Number of connections from source IP to different hosts
connDst	Number of connections from destination IP to different hosts
connSrcDst	Number of connections between source IP and destination IP
numBytesSnt	Number of transmitted bytes
numBytesRcvd	Number of received bytes
bytePS	Send bytes per second
minPktSz	Minimum layer3 packet size
maxPktSz	Maximum layer3 packet size
numPktsSnt	Number of transmitted packets
numPktsRcvd	Number of received packets
RangePl	Range of packet lengths
pktPS	Send packets per second
UppQuartilePl	Upper quartile of packet lengths
pktAsm	Packet stream asymmetry
tcpOptPktCnt	TCP options Packet count
tcpWS	TCP Window Scale
tcpAveWinSz	TCP average window size
tcpInitWinSz	TCP initial window size
tcpPSeqCnt	TCP packet seq count
tcpRTTAckTripAve	TCP Ack Trip Average
ipMinTTL	IP Minimum TTL
ipMaxTTL	IP Maximum TTL
ipTTLChg	IP TTL Change count
MinIat	Minimum inter-arrival time
skewIat	Skewness of inter-arrival times
lowQuartileIat	Lower quartile of inter-arrival times
tcpMSS	TCP Maximum Segment Length
ipMaxdIPID	IP Maximum delta IP ID

types of botnets. In short, connection-based features seem to be more important for the P2P botnet detection. This might be because of the frequent connections between the peers on the network. Intuitively, this makes sense to us. On the other hand, inter-arrival based features are more likely to be significant in HTTP botnet detection. This might be because of the automated way that C&C servers and the bots are configured to communicate. In other words, bots and servers are setup to talk based on specific time intervals. Moreover, as our previous works have also indicated, packet based and byte based features are important given that botnets behave differently in terms of the number of packets and bytes sent/received compared to the behaviours of the legitimate users.

Although we have observed discrepancy between the HTTP and the P2P botnet detection models in terms of features with the highest information gain or frequency, all the aforementioned important categories of features were used by the C4.5 classifier for both types of botnets in order to build the detection

Table 7: Features with the highest information gain

	Data Set	Features
P2P	ZeroAccess (CVUT)	ConnSrcDst, minPktSz, ConnSrc, tcpWS, ipMinTTL
	Kelihos (CVUT)	ipMinTTL, pktPS, tcpAveWinSz, tcpOptPktCnt, Duration, ipTTLChg, connDst
	NSIS (CVUT)	minPktSz, pktPS, connSrcDst, pktAsm, connSrc, numBytesRcvd
	ISOT (Uvic)	connSrcDst, UppQuartilePl, connDst, ipMinTTL
HTTP	Zeus (NIMS)	tcpInitWinSz, pktPS, MinIat, SkewIat, pktPS, numPktsSnt
	Citadel (NIMS)	tcpRTTAckTripAve, bytPS, MinIat, ipMaxTTL, connDst
	Conficker (CAIDA)	numPktsRcvd, numBytesSnt, minPktSz, Duration
	Virut (CVUT)	maxPktSz, tcpPSeqCnt, pktPS, lowQuartileIat, pktAsm, RangePl, connDst

Table 8: Features with the highest frequency

	Data Set	Features
P2P	ZeroAccess (CVUT)	connSrc, connDst, numBytesSnt, ipMinTTL, minPktSz, numBytesRcvd
	Kelihos (CVUT)	numBytesRcvd, numBytesSnt, connSrc, MinPktSize, connDst, tcpInitWinSz
	NSIS (CVUT)	ipMinTTL, connDst, MinPktSz, numBytesSnt, connSrc, numBytesRcvd
	ISOT (Uvic)	connSrcDst, ipMinTTL, MaxPktSz, numBytesRcvd, connDst, numBytesSnt
HTTP	Zeus (NIMS)	pktPS, ipMaxIPID, pktAsm, MinIat, ipMaxTTL
	Citadel (NIMS)	MinIat, ipMaxTTL, numBytesSnt, minPktSz, RangeIat
	Conficker (CAIDA)	ipMinTTL, TcpInitWinSz, Duration, ipMaxdIPID, connDst, tcpMSS
	Virut (CVUT)	ipMinTTL, numpktRcvd, connDst, TcpInitWinSz, connSrc, MaxPktSz

model. In other words, they all have been appeared in the decision trees of both types of botnets at some point. Hence, we cannot exclusively assign/relate any categories of features to P2P or HTTP botnets. This might be caused by the fact that P2P botnets can use HTTP protocol as the base of communication (forming some similarities). Kelihos and ZeroAccess are good examples of such kind of P2P botnets, which are employed in this work.

In summary, we aimed to evaluate the combination of Tranalyzer feature set and C4.5 classification algorithm on P2P botnets given that this combination was shown to be effective on HTTP botnets. The analysis, evaluation and results show that this approach is as effective for P2P botnets as it was for HTTP botnets. Moreover, we could not highlight any obvious differences be-

Table 9: HTTP and P2P botnets versus legitimate behaviour

	Data Set	DR	Botnet		Legitimate		Complexity	
			TPR	FPR	TNR	FNR	Time	Solution
C4.5	HPL	99.9%	99.99%	0.1%	99.9%	0.1%	13171.78	5781

tween the HTTP and P2P classification models to demonstrate the differences of botnet behaviours. Thus, the new research question to answer is: How would this combination handle a traffic trace that consists of P2P botnets, HTTP botnets and legitimate traces? In other words, can this combination differentiate botnet behaviours (P2P and HTTP) from legitimate behaviours? To this end, we generated a new balanced data set combining all the eight botnet data sets employed in this work labeled as "botnet" and the LBNL legitimate data set labeled as "legitimate". We refer to this data set as HPL (HttpP2pLegitimate) with 3005999 botnet samples and 3005999 legitimate samples. Table 9 shows the results of this classification indicating that C4.5 can classify the HTTP and the P2P botnet behaviours from legitimate behaviours using Tranalyzer feature set. Features with the highest information gain in this classification model are: numBytesSnt, tcpPSeqCnt, connSrcDst, connDst, ipOptCnt, connSrc, minPktSz, ipMinTTL, MinIat.

The result also shows that the HTTP and the P2P botnets do have some similarities since C4.5 can put them together as one class. This can be based on the HTTP protocol that is used by the P2P botnets or the similar automated nature of botnet behaviours in general. Hence, to understand if these botnets have enough distinct behaviours that can be used to differentiate them despite the similarities, we run another experiment. In this new experiment, we generated a multi-class data set, called HPL-multiClass which has nine classes: one legitimate class and eight botnet classes (four HTTP and four P2P classes). Since some of the data sets used in this work are much larger than the others (such as Conficker), we kept the new multi-class data set unbalanced, which consists of all of the samples mentioned in Table 1. The result of this experiment is shown in Table 10, indicating that C4.5 performed well while being able to differentiate eight different botnet behaviours from legitimate behaviours (with overall DR of 99.88%). However, given that this is an unbalanced multi-class classification, a classwise average DR can better demonstrate the performance. To this end, we utilized *Score* measure, which summarizes the classwise detection rates of a classifier over all classes. Score criteria is defined as, Eq. 6, [13]:

$$Score = \frac{1}{|C|} \sum_{c \in C} DET_c \tag{1}$$

where $DETc$ is the detection rate for class c. Based on Score, we conclude that the HTTP, P2P botnets and legitimate traces employed in this work can be differentiated with the accuracy of 97% despite the similarities shown in the previous experiment (Table 9). Moreover, the results indicate that the TPR of

Table 10: HPL-multiClass classification results

	Data Set	TPR	FPR
P2P	ZeroAccess (CVUT)	99.8%	0%
	Kelihos (CVUT)	99.8%	0%
	NSIS (CVUT)	96.8%	0%
	ISOT (Uvic)	99.4%	0%
HTTP	Zeus (NIMS)	92.8%	0%
	Citadel (NIMS)	91.0%	0%
	Conficker (CAIDA)	100%	0%
	Virut (CVUT)	99.8%	0%
Legitimate	LBNL	99.9%	0.1%
Overall DR = 99.88% Score = 97.7%			

Zeus (NIMS), Citadel (NIMS) and NSIS (CVUT) are not as high as the other five botnets. Analyzing the confusion matrix showed that: (i) almost all the Zeus (NIMS) miss-classified samples are classified as Citadel (NIMS), (ii) almost all of the Citadel (NIMS) miss-classified samples are classified as Zeus (NIMS), and (iii) most of the NSIS (CVUT) miss-classified samples are classified as Legitimate, Conficker (CAIDA), Virut (CVUT) and Kelihos(CVUT). We believe that the Citadel and Zeus miss-classification pattern is caused by the fact that Citadel botnet is an enhanced version of the Zeus botnet, and therefore they share some similarities in behaviours. On the other hand, NSIS (CVUT) miss-classification pattern showed that this botnet has some similarities with HTTP botnets in terms of behaviour, which should be further investigated.

5 Conclusions

A Botnet, which is a network of infected hosts remotely controlled by a botmaster, is considered as one of the main cybersecurity challenges given the variety, the high infection rate and the extended range of malicious tasks. Being able to upgrade any part of the structure on the fly is one of the reasons why this type of malware could sustain itself since 2003. To this end, detection systems also require automatic and intelligent mechanisms to cope with the updates. In this work, we employed three machine learning algorithms, namely C4.5, Bayesian Networks and SBB, to generate botnet detection models for several types of botnets such as Zeus, Citadel, Virut, Conficker, Kelihos and ZeroAccess. These botnets can be categorized into two main groups: HTTP and P2P. To represent the traffic traces, Tranalyzer flow exporter was utilized which aggregate the packets into traffic flows and extract their features.

Our main objective in this work was to investigate the possibility of detecting P2P botnets with our previously proposed approach on HTTP botnets. As the results indicate, the combination of Tranalyzer feature set and the C4.5 ML

algorithm can also be effective in P2P botnet detection. We obtained the DR of up to 99.95%. Additionally, similarities and differences of these botnets were further investigated by two multi-botnet classification scenarios. Again the detection performances were very promising, reaching up to 99.9%. Moreover, the detection models are analyzed and compared using the features selected and utilized by the C4.5 decision trees in order to highlight the behavioural differences between the HTTP and the P2P botnets, if possible. In this case, the features with the highest information gain and the highest frequency are investigated. The analysis showed that some of the features could potentially be more useful in P2P botnet detection (such as connection-based features) while some others can be better to describe the HTTP botnets (such as inter-arrival based features). This is to say that these features cannot be specifically assigned to one type of botnet (HTTP or P2P). However, the degree of importance might be different.

6 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. Tranalyzer. http://tranalyzer.com/.
2. E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.
3. E. B. Beigi, H. Jazi, N. Stakhanova, and A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *Communications and Network Security (CNS)*, 2014.
4. CAIDA Conficker. http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml.
5. M. Feily and A. Shahrestani. A survey of botnet and botnet detection emerging security information. In *Systems and Technologies*, 2009.
6. S. Garcia. Malware capture facility project, cvut university. https://agents.fel.cvut.cz/malware-capture-facility, February 2013.
7. S. Garcia, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers and Security*, 45:100–123, 2014.
8. F. Haddadi, D. L. Cong, L. Porter, and A. N. Zincir-Heywood. On the effectiveness of different botnet detection approaches. In *ISPEC*, 2015.
9. F. Haddadi, D. Runkel, A. Zincir-Heywood, and M. Heywood. On botnet behaviour analysis using GP and C4.5. In *Gecco comp.*, 2014.
10. F. Haddadi and A. N. Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems journal*, 2014.
11. F. Haddadi and A. N. Zincir-Heywood. Botnet detection system analysis on the effect of botnet evolution and feature representation. In *Gecco comp.*, 2015.

12. V. Kirubavathi and R. Nadarajan. Http botnet detection using adaptive learning rate multilayer feed-forward neural network. In *Information Security Theory and Practice: security, privacy and trust in computing systems and ambient intelligent ecosystems*, 2012.
13. P. Lichodzijewski and M. I. heywood. Coevolutionary bid-based genetic programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*, 9:331–365, 2008.
14. RFC 2722. http://tools.ietf.org/html/rfc2722, October 1999.
15. S. T. Vuong and M. S. Alam. Advanced methods for botnet intrusion detection systems. In *Intrusion Detection Systems*, 2011.
16. K. Wang, C. Huang, S. Lin, and Y. Lin. A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks*, 55:3275–3286, 2011.
17. P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically generating models for botnet detection. In *14th European conference on research in computer security (ESORICS)*, 2009.
18. J. Zhang, R. Perdisci, U. S. W. Lee, and Z. Luo. Detecting stealthy p2p botnets using statistical traffic fingerprints. In *Dependable Systems and Networks (DSN)*, 2011.
19. D. Zhao, I. Traore, B. Sayed, W. Lu, a. G. S. Saad, and D. Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers and Security*, 39, 2013.