INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2016; **00**:1–20

Botnet Behaviour Analysis: How would a data analytics-based system with minimum a priori information perform?

Fariba Haddaddi*, A. Nur Zincir-Heywood

Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada

SUMMARY

Botnets, as one of the most aggressive threats, has employed different techniques, topologies and communication protocols in different stages of their lifecycle since 2003. Hence, identifying botnets have become very challenging specifically given that they can upgrade their methodology at any time. Various detection approaches have been proposed by the cyber-security researchers, focusing on different aspects of these threats. In this work, five different botnet detection approaches are investigated. These systems are selected based on the technique used and type of data employed where two are public rule based systems (BotHunter and Snort) and the other three employ machine learning algorithm with different feature extraction methods (packet payload based and traffic flow based). On the other hand, four of these systems are based on a priori knowledge while one is using minimum a priori information. The objective in this analysis is to evaluate the effectiveness of these approaches under different scenarios (e.g. multi-botnet and single-botnet classifications) as well as exploring how a system with minimum a priori information would perform. The goal is to investigate if a system with minimum a priori information could result in a competitive performance compared to systems using a priori knowledge. The evaluation is shown on twenty four publicly available botnet data sets. Results indicate that a machine learning based system with minimum a priori information not only achieves a very high performance but also generalizes much better than the other systems evaluated on a wide range of botnet structures (from centralized to de-centralized botnets). Copyright © 2016 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Botnet behaviour; Data analytics; Machine Learning; Network flow analysis

1. INTRODUCTION

A network of compromised hosts (i.e. 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

*Correspondence to: Faculty of Computer Science, Dalhousie University, 6050 University Avenue, Halifax, NS B3H 4R2, Canada.

can be extracted per packet (or in some cases specific packets) or per flow[†]. 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: A packet payload based approach and two network flow based approaches (Tranalyzer-2 [4] based system and Flow Aggregation/Fraction system(FlowAF)). 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 using payload information given that they can be applied to encrypted traffic (where the payload is opaque). Our goal however is to understand how much could be gained when a system also takes advantage of payload analysis. To this end, we have employed the C4.5 decision trees for analyzing (i) the traffic flows and (ii) all the traffic including the payload. In addition to these machine learning based systems, publicly available intrusion/botnet detection systems are also investigated in this work.

In this case, we evaluate Snort [3] and BotHunter [22] as the rule based detection systems. Snort is one of the known intrusion detection and prevention systems (IDS/IPS). Being an open source tool with the ability to customize the rule set and the rule sets being updated frequently, are the main advantages of Snort. BotHunter, which is another publicly available system, utilizes the Snort sensors and customizes Snort rule set for specifically detecting botnet.

From a perspective of a human expert, some botnet detection systems use a priori knowledge to define the set of features and rules or use various cycles of analysis to specify the most useful feature sets according to a specific case or data. Bothunter, Snort, the packet payload based system and FlowAF are placed in this category. On the other hand, there are systems that use minimum a priori knowledge for this purpose. These systems would get open source tools and publicly available feature sets that are not designed for a specific problem and utilize them in botnet detection. To this end, using all of the Tranalyzer-2 exported netflow features with the C4.5 classifier, which selects the most important features with the highest information gain, would result in a botnet detection system with minimum a priori knowledge. In this work, we aim to explore how far the performance of such a system could be pushed with using minimum a priori knowledge and how such systems would perform compared to the systems with a priori knowledge.

Last but not the least, we have evaluated all of these five detection systems on twenty four different publicly available botnet data sets. To the best of our knowledge, this is the most comprehensive set of data sets that have been utilized in the evaluation of a botnet detection system. These botnets range from IRC botnets, to De-centralized HTTP botnets and Peer-to-Peer botnets.

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. RELATED WORK

Over time, botnets have employed different protocols, topologies and techniques to implement the five phases of the lifecycle [29] while avoiding detection. Hence, an arms race has started between the botnets and the detection systems. There are various techniques proposed for botnet detection due to the wide range of botnet methodologies. From the data perspective, while some techniques focus on malware source/binary analysis [6], others use host and/or network-based data [21, 46]. In this section, we focus on the network-based systems in the literature given that the systems evaluated in this research are placed in this category.

[†]Flow is defined as a logical equivalent for a call or a connection in association with a user specified group of elements [35]. 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.

Gu *et al.* proposed and developed two botnet detection frameworks called BotHunter and BotMiner[22, 21]. BotHunter is a botnet detection system based on correlation of Snort IDS alerts and bot activities. On the other hand, BotMiner is a botnet detection framework based on group behaviour analysis [21]. BotMiner uses a clustering approach to find similar Command and Control (C&C) communication behaviours, which form clusters, and then employs Snort to find the type of activity in the detected clusters. Flow features such as the number of packets per flow, the average number of bytes per packet and the average number of bytes per second are extracted and employed for the detection. BotMiner was evaluated on several traffic data sets.

Strayer *et al.* developed an IRC botnet detection system that made use of machine learning techniques (classification and clustering) [39]. First, a classification technique is used to filter the chat type of traffic and then a clustering technique is applied to find the group activities in the filtered traffic. Finally, a topology analyzer is utilized on the clusters to detect the botnets. In this three layer approach, they employed flow-based features extracted from packet headers.

Wurzinger *et al.* proposed an approach to detect botnets based on the correlation of commands and responses in the monitored network traces [43]. The two base assumptions in designing the approach were: bots receive commands from botmasters and then carry out actions in response to those commands. To identify traffic responses, the corresponding commands in the preceding traffic are located. Known content-based specifications (*i.e.* signatures) were utilized to identify the commands and responses. Then, using the command and response pairs, the detection model was built focussing on IRC, HTTP and P2P botnets. Data sets used in this paper 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 behaviour.

Francois *et al.* proposed a NetFlow monitoring framework to detect P2P botnets leveraging a simple host dependency model to track communication patterns. The system first creates a dependency graph between hosts by monitoring their interactions. Then, linkage and clustering algorithms are employed to identify similar botnet traffic patterns (*i.e.* identifying hosts that are highly linked together) [11].

Lu *et al.* proposed a two-phase botnet detection system [32]. In the first phase the system classifies the network traffic using traffic payload signatures and then a decision tree model is used in the second phase to classify the unknown traffic by the payload content of the first phase.

Kirubavathi *et al.* designed an HTTP-based botnet detection system using a multilayer Feed-Forward Neural network [30]. 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, the system extracts features related to TCP connections in specific time intervals based on the packet headers. The features are then normalized and used to train a multilayer feed-forward neural network.

Guerid *et al.* proposed a collaborative and inter-domain botnet detection system which enables real-time analysis for large scale networks [23]. The system has several connected probes in which each probe correlates the information gathered from its network and the anonymized data received from other probes. The probe then uses two layers of analysis: (i) a community structure layer and (ii) a C&C server detection layer. The first layer detects the infected bots and groups them into communities with the same botnet behaviour. This detection is based on participation in the same ongoing attacks or in the similarity of their abnormal network traffic. The second layer receives the communities from the first layer and identifies the associated malicious servers.

Zhao *et al.* investigated a botnet detection system based on flow intervals [49]. Flow features of traffic packets were extracted based on several time intervals and utilized by several ML algorithms to find the best combination of flow features, time interval and ML algorithm. Accordingly, decision tree classifier was finally selected as the preferred classifier to detect botnets with a proposed feature set using a time interval of 180 seconds. The authors focussed on P2P botnets (such as Waledac) which employ the HTTP protocol and a fast-flux based DNS technique.

Dietrich *et al.* presented an approach (CoCoSpot) to detect botnet C&C channels based on traffic analysis features [10]. In this approach, flow features of TCP and UDP traffic are extracted from the packet header and payload. A hierarchical clustering technique is applied to the flows to form the

clusters which are then labeled manually by the authors as malicious or normal. Using the clusters and cluster centroids, a classifier is trained to be used for classifying the unknown test traffic.

Yan *et al.* proposed PeerClean as a three layer peer-to-peer botnet detection system which investigates the traffic flow statistics and network connection patterns for this purpose [44]. The first layer clusters the hosts based on similar traffic patterns using the flow features. The second layer applies a dynamic group behaviour analysis (DGBA) on the first layer clusters to extract the group-level aggregated connection features. An SVM classifier is then used to reveal the botnet clusters. Finally the third layer identifies the botnet types of the detected botnet clusters.

Stevanovic *et al.* proposed three traffic analysis methods to detect botnet behaviours using the TCP, UPD and DNS protocol as the main carrier of botnet C&C communication [38]. After extracting features for each of these protocols, a random forest classifier is utilized to generate the detection models. For the TCP and UDP-based methods conversation features (flow features) are extracted from the packet headers without using the IP addresses while for the DNS-based method, features are extracted from the DNS queries and responses. The novel idea of extracting the conversation features is to calculate the features based on a sub-sample of the packets in each conversation.

Wang *et al.* proposed a two-stage approach to detect botnets [40]. The first stage detects network anomalies which are linked to the botnet presence in the network while the second stage detects the infected machines (bots) based on these anomalies. Two anomaly detection methods were used for the first stage: (i) a flow-level anomaly detection method which quantifies the flow data and monitors the histograms of data and (ii) a graph-based packet-level anomaly detection method which aggregates the packet-level data into graphs and then monitors the distribution degree. In the second stage, highly interactive nodes and their relationships are identified by constructing a Social Correlation Graph.

In summary, network-based detection systems can be categorized in different ways: (i) some use group behaviour analysis [21, 39, 23] while others are focussed on single user behaviour and not on finding behaviours that are shared between a group of users. Using group behaviour analysis limits the applicability of the detection approach to the large networks which are being monitored. In other words, if the network under investigation/monitoring does not include multiple infected bots, group-based monitoring approaches might not be very useful. However, the approaches that are not focussed on group behaviour analysis can be effective regardless of the number of infected hosts on the network. (ii) From the data perspective, some of the systems use the packet payload and the header information [41, 43], while others employ only the packet header information (*e.g.* flow-based systems) [47, 49, 10]. The importance of the approaches in the second group can be understood better knowing that the most recent aggressive botnets employ encryption to hide themselves and their information from the detection systems. Moreover, there are several studies on flow-based botnet detection systems in which each has proposed a different set of features [47, 49]. Some studies have analyzed the feature selection algorithms to extract the most effective feature sets. Such feature selection processes can cause the models to be focussed on specific type(s) of botnet(s) which may not be very effective for other types. (iii) While some of the approaches are focussed on botnets with specific communication structures or protocols [11, 30, 44], there are approaches which can be applied to various botnet structures and protocols [21, 46, 40].

This work presents an extended evaluation and comparison of five botnet detection systems. These systems are selected from different groups regarding the above categorization. The goal is to demonstrate how each of these systems would perform on various types of botnets under different scenarios (e.g. multi-class classification scenario).

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 the features extracted [29]. Some systems/ approaches [22, 43] require both the payload and

the header section of the packets to extract the necessary features while others [8, 49, 29] 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. However, having access to payload information can effect the system performance. Therefore, to analyze the effectiveness of such systems, we aim to evaluate and analyze the following systems for botnet detection: (i) Packet payload based system; (ii) Two flow-based systems; (iii) Snort intrusion detection system and (iv) BotHunter botnet detection system.

3.1. Systems employed

Snort and BotHunter are two publically available intrusion/botnet detection systems. These two systems are selected given that BotHunter is the botnet detection system used in the literature [20, 45] for performance evaluation purposes and Snort is a highly employed malicious behaviour detection system in literature as well as in industry [9, 45]. Among the approaches that use packet headers information only, flow-based feature extraction methods have been highly employed in the recent literature [8, 29, 41]. Hence, two flow-based systems are also employed and evaluated in this work: a system proposed, developed and evaluated in [29] (refer to Tranalyzer-2 flow-based system hereafter) and another flow-based detection system proposed by Zhao et al. [49] (called FlowAF hereafter). In addition to the above systems, a packet payload-based system inspired by a system proposed by Mohaisen et al. [33] is employed as well.

Regarding the level of human expert involvement, Snort, BotHunter, the packet payload based system and FlowAF are considered systems with a priori information. These systems use the experts knowledge, to define the rules and features used for detection purposes. However, the Tranalyzer-2 flow-based systems use minimum a priori knowledge to extract a wide range of features which can be used for traffic analysis in general.

3.1.1. BotHunter : 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 that all botnet infection processes are similar and can be illustrated by a botnet lifecycle model. Hence, it uses a modified version of Snort with its plugins to detect the 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 IP address checking into the 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 anomalous traffic pattern and payload detection. Recently, these plugins have been 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 the 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.

3.1.2. Snort : Snort is an intrusion detection and prevention system that analyzes packet payloads as well as packet header data to detect any evidence of harmful actions which match predefined signatures (rule sets) based on a priori knowledge [3]. Some of these pre-defined rules/signatures take advantage of payload information while 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. As discussed in more detail in Section 4, the VRT rule set was used. These two main rule sets have come with many rules which aim to cover all possible network conditions. Users should be careful to enable only those rules that fit their network conditions and alert priority settings and disable the

Table I. 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

others. Since 1998, Snort has been known and used in the network security area given that it is a cross-platform open source IDS/IPS which can be modified to fit the network security challenges and needs as shown by the BotHunter research.

3.1.3. Packet payload-based System : Some of the works in the literature proposed specific packet analysis methods to detect botnet behaviour [25, 41]. These systems have focussed 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.* [25] extracted the domain name from the DNS packets to detect automatically generated malicious domain names while Mohaisen et al. [33] introduced a set of features focussing on the Zeus botnet. The features introduced by Mohaisen *et al.* based on a priori knowledge are employed in the evaluations of the proposed packet payload-based system. Table I presents the selected features for this approach.

Since the C4.5 classifier employed in this work, it can be applied to only 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 [33]. Once the features are extracted from each data set, C4.5 is applied for classification.

3.1.4. Tranalyzer-2 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 [37][8][41]. In such approaches, communication packets are aggregated into flows and then statistics are calculated. The critical phase of such systems is the flow exporting.

In this research, we develop a flow based botnet detection system based on our previous work [29]. The previous work analyzed the effect of flow exporters reporting Tranalyzer as the best performing flow exporter among the five tools (Maji, YAF, Softflowd, Tranalyzer and Netmate). Given that a new version of Tranalyzer (v. 0.5.9) has become publicly available since the previous work, this version is utilized here. This version is referred to as Tranalyzer-2 hereafter.

After extracting the flow features , the C4.5 classifier is employed to detect the botnet behaviour. This classifier is chosen as the best performing classifier from a set of machine learning algorithms (SVM, SBB, C4.5, ANN, KNN, Bayesian Networks, and Naive Bayes) that have been evaluated in our previous works. These are algorithms used widely in the literature ([39, 8, 49, 46, 36]) for network traffic analysis. It should be noted here that we employ all of the features exported by Tranalyzer-2 as inputs to this 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 future work to introduce such features. Overall, Tranalyzer-2 based system is using a wide range of features and no a priori knowledge has been used in the process of feature selection. C4.5 classifier, on the other hand, is selecting the most informative feature set without any human interference. This is the reason that this system can be considered as minimum a priori based detection system.

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

Table II. Classification Results– with a flow interval of 300.

	System	DR	Botnet		Legitimate	
			TPR	FPR	TNR	FNR
C4.5	Zhao *et al.* [49]	-	98.3%	0.1%	99.9%	1.7%
	FlowAF (balanced)	99.9%	99.9%	0.1%	99.9%	0.1%
	FlowAF (unbalanced)	99%	98%	0.4%	99.6%	2.0%

not require any collector which makes it very easy to use. Tranalyzer-2 supports 98 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, 29]. Tranalyzer-2 has five additional features compared to Tranalyzer-1[‡].

3.1.5. Flow aggregation/fraction-based System (FlowAF) : Zhao *et al.* proposed a botnet detection system based on flow intervals [49]. Depending on the value used for the flow interval, this system can result in flow aggregation or flow fraction. In other words, if the interval value is greater than the duration of a flow, corresponding flows (based on the 5-tuple information) will be aggregated. Otherwise, the flow will be divided into smaller chunks. In Zhao's system, a set of Flow features were utilized with several ML algorithms in which a decision tree classifier was selected as the preferred classifier for detecting botnets. The authors focussed on P2P botnets (such as Waledac) that employ the HTTP protocol and a fast-flux-based DNS technique. Furthermore, based on their proposed approach, a web-based detection system was implemented to be used both in offline detection as well as live detection. To evaluate their proposed approach and web-based detection system, they employed a combination of normal and attack traffic, some of which was generated in the lab, some was from Honeynet project traces and some was from the Lawrence Berkeley National Laboratory (normal traffic) data sets. In this work, this data set is referred to as ISOT (Uvic). Although their proposed detection approach resulted in up to 99% detection rates with a false positive rate around 2%, they also tested their system with unseen botnet data and obtained detection rates up to 100% while having a false positive rate of about 80% in some cases.

As discussed earlier, most of the flow-based botnet detection systems have introduced their own set of features which are evaluated mostly on specific types of botnet. Although Zhao *et al.* have introduced their preferred set of features, they have also evaluated the effect of flow aggregation and flow fraction. Table III shows the feature set used by Zhao *et al.* Therefore, their system was chosen as one of the systems for the evaluations in this work. Based on the information given in [49], a program was created to implement the flow extraction using flow intervals, referred to as FlowAF. First, FlowAF was evaluated on the same ISOT (Uvic) data set that was used and published by Zhao *et al.* to validate the implementation. Given that the RepTree classifier with 10-fold cross-validation and a flow interval of 300 sec was used for the evaluation of this approach at [46], the same setting is utilized here, too. Table II shows the results of FlowAF and the original implementation in [49]. Furthermore, Zhao et al. did not mention whether the data set was balanced or unbalanced in [49] so we ran FlowAF with both of the configurations. It is likely the authors employed the unbalanced setting since they only mentioned using the ISOT (Uvic) data set (which is an unbalanced data set). The results presented in Table II demonstrate that our FlowAF implementation did perform similarly to the original implementation– specially with the unbalanced setting. Hence, FlowAF will likely be a good representative of the system proposed by Zhao *et al.* [49] for the evaluations and comparisons in this research.

[‡]Additional Features are: ICMP Status, TCP Ack Trip Jitter Average, TCP Ack Round trip average Jitter, Subnet number of source IPv4, Subnet number of destination IP.

Table III. Selected feature set in [46].

Feature	Description
SrcIP	Flow source IP address
SrcPort	Flow source port number
DstIP	Flow destination IP address
DstPort	Flow destination port number
Protocol	Transport layer protocol or mixed
APL	Average payload packet length for time interval
PV	Variance of payload packet length for time interval
PX	Number of packets exchanged for time interval
PPS	Number of packets exchanged per second in time interval T
FPS	The size of the first packet in the flow
TBP	The average time between packets in time interval T
NR	The number of reconnects for a flow
FPH	Number of flows from this address over the total number of flows generated per hour

3.2. Machine learning algorithm

C4.5 is a decision tree algorithm, which is a non-parametric supervised learning method, first developed by Quinlan. This algorithm is an extension to ID3 algorithm which aims to find the small decision trees (using pruning) and then convert the trained tree into an if-then rule set. C4.5 constructs decision trees based on a training data set, where each exemplar is a set of already classified (labeled) attributes, by applying the Information Entropy concept. The algorithm employs a normalized information gain criterion to select attributes from a given set of attributes to determine the splitting point. 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 [5].

3.3. Data sets

In this work, twenty four publicly available botnet data sets and one legitimate publicly available data set are employed. To the best of our knowledge, this is the most comprehensive collection of data sets used for any evaluation of this nature. These data sets have been selected over a period of five years representing botnets with different communication architectures and protocols. This is to evaluate the systems on the botnet evolution and investigate to what extent the systems can cope with the newer botnets that come out over time.

In the botnet category: (i) Zeus (Snort): The snort Sourcefire vulnerability research team lab has provided three sample traffic log files for the Zeus botnet in 2010 [1]. Two of them are very small (only 110 and 1105 packets) and therefore could not be used in these experiments. Hence, "Sample_1" is the only Zeus traffic file utilized from the Snort archive in this work. (ii) NETRESEC data sets: The NETRESEC repository provides sample traffic log files for the Citadel, Cutwail, Kelihos and Zeus botnets [2]. These data sets are referred to as Citadel (NETRESEC), Cutwail (NETRESEC), Kelihos (NETRESEC), Zeus (NETRESEC). (iii) The Malware Capture Facility Project is a research project defined by the Czech Technical University Agent Technology Center (ATG) for capturing, analyzing and publishing malware traffic [19]. Zeus [17], Kelihos [15], Neris [16], NSIS [18], Rbot [12], ZeroAccess [14] and Virut [13] are the botnets selected from this repository. (iv) The CAIDA organization has captured and made publicly available a Conficker botnet data set [7]. This traffic log, captured over three days, was collected by the CAIDA UCSD network telescope when the Conficker botnet was active in 2008. Hereafter, this data set will be referred to as the Conficker (CAIDA). (v) In addition to the publicly available log files, several botnet traffic traces were generated in the NIMS lab at Dalhousie University. These log files can be categorized into two major types: Sandbox log files and Domain-based HTTP log files. In the Sandbox group, the Zeus toolkit (version 1.2.7.19 and 2.1.0.1) and Citadel toolkit (version 1.3.5.1) have been employed to generate botnet data samples. This method of generating data sets have

Table IV. The sandbox configuration.

Data set	No. of bots	No. of servers	Server type	Description
Zeus-T1 (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 1.2.7.19
Zeus-T1-W (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 1.2.7.19 Web Injection enabled
Zeus-T2 (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 2.1.0.1
Zeus-T2-W (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 2.1.0.1 Web Injection enabled
Citadel-T1 (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 1.3.5.1
Citadel-T1-W (NIMS)	12 Windows machines	2	Linux and Windows	Zeus botnet version 1.3.5.1 Web Injection enabled

been widely used in the literature [43, 34, 41]. Hereafter, these data sets will be referred to by the names used in the first column of Table IV. In Domain-based HTTP category, the focus is on the botnets, which employ the HTTP protocol as their communication protocol. Rather than running botnet binaries in sandbox environments, publicly available lists (from legitimate resources) of C&C domain names were employed for generating the representative botnet/legitimate traffic. These data sets are analyzed, evaluated and confirm in [27]. Hereafter, the domain-based generated data sets will be named with a "-D" extension: *e.g.* like "Conficker-D (NIMS)".

3.4. Performance criteria

In data 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 work, DR reflects the number of correctly classified specific botnet exemplars in a given traffic file/domain name list and is calculated using $DR = \frac{TP}{TP+FN}$; whereas TP (True Positive) is the number of botnet exemplars that are classified correctly and FN (False Negative) is the number of botnet exemplars that are classified incorrectly (as normal). FPR, on the other hand, shows the number of normal exemplars that are classified incorrectly as botnets and is calculated using $TPR = \frac{FP}{FP+TN}$; whereas TN (True Negative) is the number of normal exemplars that are classified correctly.

Although typically, classifiers are evaluated using the DR, given an unbalanced data set or a multi-class data set, this metric can be misleading. In this regard, a classwise detection rate is defined as: $DET_c = \frac{TP_c}{FN_c+TP_c}$; where $DETc$ is the class $c \in C(all\ classes)$ detection rate and TPc and FNc are the True-Positive and False-Negative counts for class c. Finally, to summarize the classwise detection rates of a classifier over all classes [31], the average DR criteria is defined by:

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

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**: This is estimated on a common computing platform. **2) Solution complexity**: The tree size for C4.5 is considered as the unit of measurement.

4. EVALUATIONS AND RESULTS

The evaluation and results of the aforementioned five botnet detection systems are summarized below.

Packet payload-based and Flow-based systems. The Weka [42] implementation of C4.5 was employed for the evaluation of the packet payload-based and the two flow-based systems in this section. Twenty-four balanced data sets were used in this evaluation with ISP (WiSNet) representing the normal behaviour. Table XI shows the detailed classification results of these three systems. Figure 1 displays the Score performance specifically. It should be noted that the IP addresses and Port numbers of the flow features were removed in order to design the Tranalyzer-2-based system. This should increase the generalization abilities of the detection systems for unseen behaviour. Hence, two versions of the FlowAF system were included in this section, one using all of the features

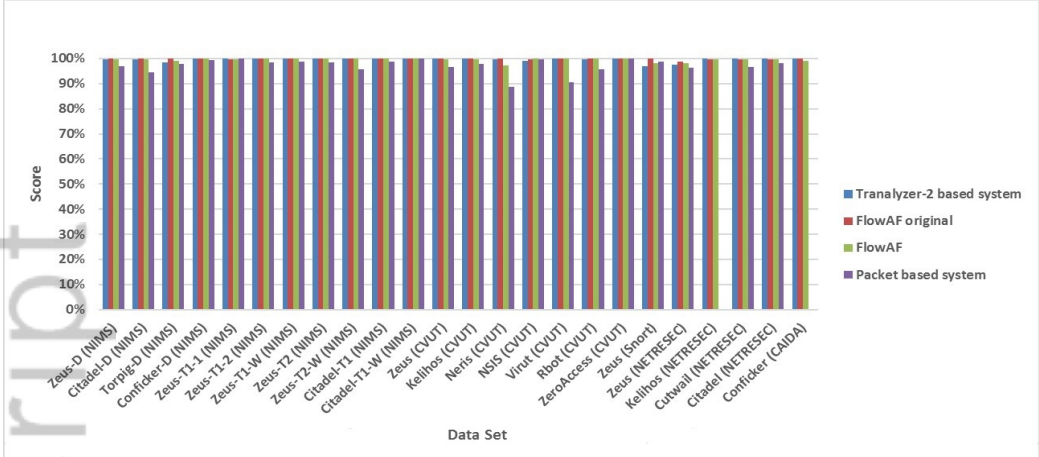

Figure 1. Score classification results

introduced in [49] (referred to as FlowAF Original here) and another excluding the IP addresses and Port numbers (referred to as FlowAF here). FlowAF would be a better base of comparison for the Tranalyzer-2 feature set.

The results in Figures 1 and 2 and Table XI demonstrate the following:

(i) The packet-based detection system is the worst performing system among the four. The main drawback of this approach is that it cannot be used when the packet payload data is not accessible, which is the case for Conficker (CAIDA) here. Moreover, packet-based detection systems performed very well and showed very promising results in terms of FPRs in comparison with the flow-based system in [24]. However, this is not the case in the evaluations here as well as in [26]. This is because these payload features are more focussed on the botnets that use HTTP as their communication protocol. Hence, they should be crafted and modified when being applied to other types of botnets. Otherwise, they do not generalize well given that packet payloads hold specific information based on the packet communication protocol. However, the flow features being analyzed in the flow-based detection systems use the packet headers. These features are generated based on the aggregation of several packets forming a connection and are more focussed on the general characteristics of the traffic (packets) being sent/received (such as size and timing) rather than the details of each packet. Hence, they could generalize better over different types of botnet and therefore, could be utilized in various scenarios of different types of botnet.

(ii) FlowAF Original performed slightly better than the Tranalyzer-2-based system and FlowAF. This can be justified by the additional IP address and port number information that is utilized by FlowAF Original given that the FlowAF and Tranalyzer-2-based system performances are very similar as shown in Figure 2. The FPR analysis in Figure 3 confirms that the Tranalyzer-2-based system and FlowAF have a similar performance while FlowAF Original performed better.

In order to understand how the two FlowAF systems would perform in differentiating several botnet behaviours from normal behaviour, a new balanced data set was generated combining the botnet data sets employed in this work, labelled as 'botnet' and the ISP (WiSNet) legitimate data set labelled as 'legitimate'. This data set is referred to as BvL (Botnet vs. Legitimate). BvL is a balanced data set with over 2.5 million instances. Table V shows the result of this experiment. Based on this result, FlowAF Original still out performed FlowAF. However, this time the Tranalyzer-2-based system performed better than FlowAF and had a performance more similar to FlowAF Original. This is an advantage for the Tranalyzer-2-based system which can achieve the same performance as FlowAF Original without using the IP addresses and port numbers which would create a more generalized system given that it is common practice by the attackers to spoof IP addresses and dynamically change port numbers. These results indicate as well that different types and versions of botnets do have some similarities since C4.5 can put them all together as one class. This can be because of the similar automated nature of botnet behaviour in general.

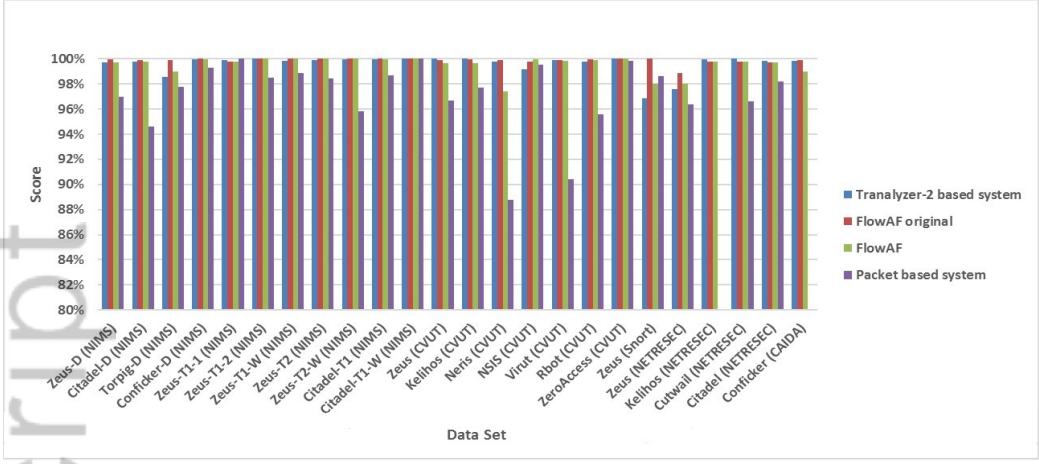

Figure 2. Score classification results (zoomed)

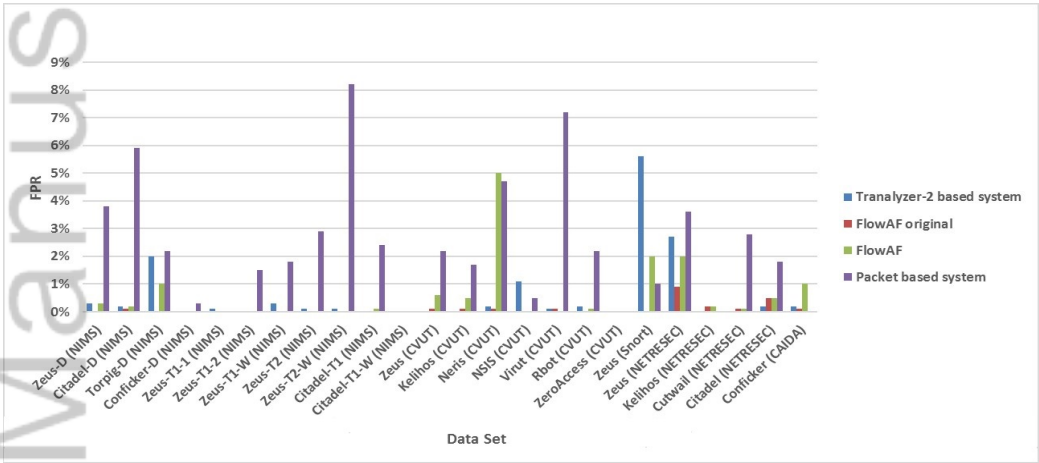

Figure 3. FPR classification results

In order to understand whether these botnets have enough distinct behaviours that can be used to differentiate them despite the similarities, another experiment was run. In this new experiment, a multi-class data set was generated (called BvL-multiClass) which has twenty-four classes: one legitimate class and the twenty-three botnet classes utilized in the analysis of this chapter. Since some of the data sets used in this work are much larger than the others (such as the Conficker-D and CVUT data sets), the new multi-class data set was kept unbalanced, consisting of all of the flow samples. In this step, both the FlowAF and Tranalyzer-2 systems were run against the BvL-MultiClass data set to test how these systems would perform on a botnet multi-class data set. Table VI and Figure 4 present the results of this experiment. The results indicate the Tranalyzer-2-based system outperformed the two FlowAF systems with an overall Score result of 88.6%.

In short, using specific and confined payload feature sets can only be useful in specific cases (i.e. for certain types of botnet) for which they are actually designed, and therefore, they should be modified when facing different situations. This can be the case as well when dealing with a limited number of features in the flow-based systems. However, the advantage of the flow-based detection system employed in this work is that it provides a wide range of features and then uses a classifier which could choose the proper set of features for each case/evaluation. This is the main reason behind the consistency of the good performance in the Tranalyzer flow-based detection system which has been evaluated under various botnets with different architectures and topologies and scenarios such as multi/single-class classifications, protocol filtering [29, 28]. The performance of

Table V. Botnet versus legitimate behaviour using the C4.5 classifier.

	Data Set	Score	Botnet		Legitimate		Complexity	
			TPR	FPR	TNR	FNR	Time (sec)	Solution
FlowAF original	BvL	99.95%	99.9%	0%	100%	0.1%	228.26	1109
FlowAF	BvL	98.9%	98.1%	0.4%	99.6%	1.9%	176.7	491
Tranalyzer-2	BvL	99.95%	99.9%	0%	100%	0.1%	2013.02	3025

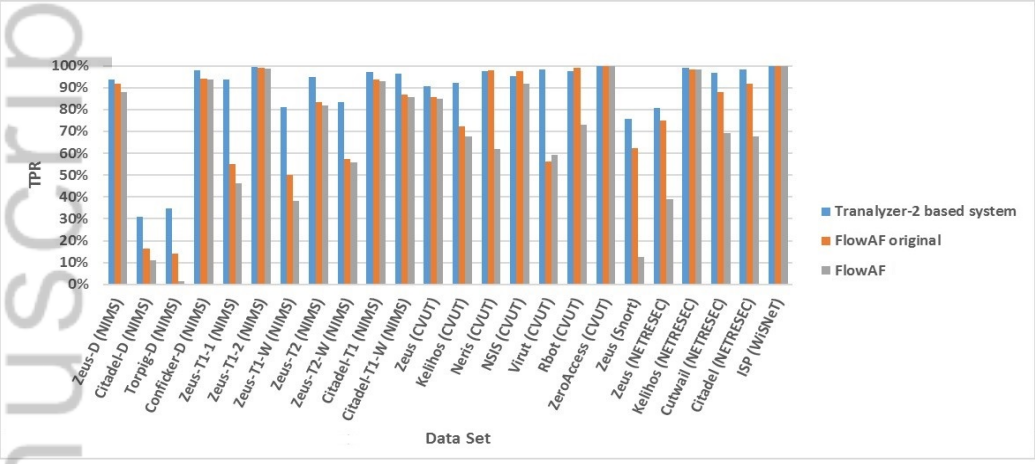

Figure 4. Multi-Class classification results.

the Tranalyzer-2 based system is higher (e.g. [34, 43]) or similar (e.g. [30, 48, 21]) to the results reported in the literature, Table VII.

BotHunter. This botnet detection tool provides Snort installation with a customized malware rule set from the ET (Emerging Threats) and DNS/IP blacklist. BotHunter should be run in batch mode when the data is in the form of pre-capture traffic log files. In this mode, two output files are created: BotHunter's Snort alert file (used as input for the BotHunter correlator), and bot profiles. As for the configuration, the trusted network/monitored network should be set for each run. Indeed, such a requirement necessitates the users to have information about the data set or the monitored network (if using BotHunter in live mode). In this research, the information provided by the sources of the log files was used to set the trusted network.

Table VIII shows the results of BotHunter on the twenty four 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. Finding the source of the attacks or at least the remote hosts that are utilized by the C&C servers is another important phase of detection. Therefore, the remote host analysis is included in this table as well. These remote machines can be the malicious C&C servers or new targets of the botnet that the infected machine aims to infect. In order to develop the bot profiles, BotHunter correlates the Snort alerts (shown in the second column) and finally generates the bot profiles revealing the malicious hosts. In Table VIII, the cells which two numbers are separated by "/" shows the count of IP addresses detected vs. the total number of IP addresses in each column. Moreover, the DR of each cell is provided in parentheses while the overall detection rate of BotHunter, including the infected hosts and the remote hosts, is presented in Table X. This table consists of two sets of overall DR: IP-based and Flow-based. The IP-based set (shown in the 'IP-based' column) is based on the IP addresses detected from Table VIII. However, to have a better understanding of how the detected IP addresses by BotHunter might reflect on the traffic flows, the flow-based overall DR results have been included. In this case, the botnet flows exported by

Table VI. BvL-MultiClass classification results.

Data Set	Tranalyzer-2		FlowAF Original		FlowAF	
	TPR	FPR	TPR	FPR	TPR	FPR
Zeus-D (NIMS)	93.8%	1%	91.7%	1.8%	87.8%	1.8%
Citadel-D (NIMS)	31.1%	0.1%	16.3%	0.1%	11.0%	0.1%
Torpig-D (NIMS)	34.7%	0%	14.1%	0%	1.5%	0%
Conficker-D (NIMS)	97.8%	0%	94.1%	0.2%	93.6%	0.2%
Zeus-T1-1 (NIMS)	93.9%	0%	55.2%	0%	46.1%	0%
Zeus-T1-2 (NIMS)	99.6%	0%	98.9%	0%	98.6%	0%
Zeus-T1-W (NIMS)	81.2%	0%	50.2%	0%	38.1%	0%
Zeus-T2 (NIMS)	95%	0%	83.4%	0%	82.0%	0%
Zeus-T2-W (NIMS)	83.2%	0%	57.3%	0%	56.0%	0%
Citadel-T1 (NIMS)	97.2%	0%	93.9%	0.1%	93.1%	0.1%
Citadel-T1-W (NIMS)	96.4%	0%	86.8%	0%	85.6%	0%
Zeus (CVUT)	90.6%	0.5%	85.5%	3.3%	85.0%	4.3%
Kelihos (CVUT)	92.1%	0.6%	72.2%	1.9%	67.8%	2.2%
Neris (CVUT)	97.4%	0.1%	98.1%	0.7%	61.8%	0.9%
NSIS (CVUT)	95.4%	0%	97.5%	0%	91.8%	0%
Virut (CVUT)	98.3%	0.1%	56.1%	0.1%	59.3%	0.3%
Rbot (CVUT)	97.7%	0.1%	99.1%	0%	73.0%	0.6%
ZeroAccess (CVUT)	100%	0%	99.9%	0%	99.9%	0%
Zeus (Snort)	75.7%	0%	62.5%	0%	12.5%	0%
Zeus (NETRESEC)	80.8%	0%	74.9%	0%	38.8%	0%
Kelihos (NETRESEC)	99.2%	0%	98.2%	0%	98.5%	0%
Cutwail (NETRESEC)	96.9%	0%	87.9%	0%	69.1%	0%
Citadel (NETRESEC)	98.5%	0%	91.8%	0%	67.8%	0%
ISP (WiSNeT)	100%	0.1%	100%	0.1%	99.7%	1.9%
	DR = 97.32% Score = 88.6%		DR = 92.46% Score = 77.73%		DR = 89.37% Score = 67.43%	

Table VII. Performances reported in the literature.

Proposed systems	DR	FPR
Kirubavathi *et al.* [30]	up to 99%	1%
Zhao *et al.* [48]	99%	0.01%
Rerdisci *et al.* [34]	up to 79%	$\sim$ 0%
Wurzinger *et al.* [43]	88%	11%
Gu *et al.* [21]	up to 100%	$\sim$ 0%

Tranalyzer-2 were labelled as 'detected' based on the IP addresses detected by BotHunter as shown in Table VIII. The 'Flow-based' column of Table X demonstrates the results of this analysis.

Based on the performance of BotHunter presented in Tables VIII and X, the observations listed below can be made.

(i) BotHunter output is dependent to the Snort-generated alerts. This dependency causes the BotHunter performance to be affected by the Snort performance. Moreover, the Snort sensor that is utilized by BotHunter is a customized version of Snort. Hence, its rule set will not get updated automatically once a new version of the Snort rule set is made publicly available. In this experiment, the Snort sensor did not generate any alerts for four of the data sets and therefore, no infected machine was detected by BotHunter.

(ii) No alert or bot profile was raised for the 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

Table VIII. Detailed BotHunter detection results.

Data Set	# Snort alerts	# Bot profiles	# Infected hosts	# Remote hosts
Zeus-D (NIMS)	11	0	0/1 (0%)	0/369 (0%)
Citadel-D (NIMS)	0	0	0/1 (0%)	0/87 (0%)
Torpig-D (NIMS)	0	0	0/1 (0%)	0/60 (0%)
Conficker-D (NIMS)	8	0	0/1 (0%)	0/1920 (0%)
Zeus-T1-1 (NIMS)	486	77	12/12 (100%)	2/2 (100%)
Zeus-T1-2 (NIMS)	29984	24	12/12 (100%)	2/2 (100%)
Zeus-T1-W (NIMS)	847	181	12/12 (100%)	2/2 (100%)
Zeus-T2 (NIMS)	8914	64	12/12 (100%)	1/1 (100%)
Zeus-T2-W (NIMS)	802	85	12/12 (100%)	1/1 (100%)
Citadel-T1 (NIMS)	32939	24	12/12 (100%)	1/1 (100%)
Citadel-T1-W (NIMS)	3363	22	12/12 (100%)	1/1 (100%)
Zeus (CVUT)	26076	0	0/3 (0%)	0/9392 (0%)
Kelihos (CVUT)	9935	0	0/1 (0%)	0/25671 (0%)
Neris (CVUT)	3006	526	10/10 (100%)	69/18369 (0.4%)
NSIS (CVUT)	96	6	3/3 (100%)	10/5364 (0.2%)
Virut (CVUT)	171	6	1/1 (100%)	8/1798 (0.5%)
Rbot (CVUT)	40521	8	1/10 (10%)	42/30046 (0.1%)
ZeroAccess (CVUT)	0	0	0/2 (0%)	0/15495 (0%)
Zeus (Snort)	26	0	0/1 (0%)	0/35 (0%)
Zeus (NETRESEC)	23	1	1/1 (100%)	7/28 (63.6%)
Kelihos (NETRESEC)	2	0	0/1 (0%)	0/268 (0%)
Cutwail (NETRESEC)	416	0	0/1 (0%)	0/162 (0%)
Citadel (NETRESEC)	216	0	0/1 (0%)	0/1 (0%)
Conficker (CAIDA)	0	0	0/360191 (0%)	0/80380 (0%)
ISOT (Uvic)	831	16	4/5 (80%)	40/15000 (0.3%)

use the payload of the traffic (packets) for detecting the botnets, they could not perform well on this data set.

(iii) BotHunter could detect all the infected machines successfully as well as the remote hosts of the NIMS sandbox-generated data sets such as Zeus-T1-1 (NIMS). That is because the payload is provided and all the phases of the botnet lifecycle are present in these data sets.

(iv) Although Snort did create high severity alarms on Zeus (Snort) (such as "E4[rb] TROJAN Zeus POST Request to CnC"), BotHunter did not report any bot profile. This shows that even when Snort does a good job of raising alerts on the anomalies, BotHunter may still not be able to create a profile for the infected machine. This could be because a correlation of the different types of alerts (representing the different phases of the lifecycle) is required by BotHunter in order to form a bot profile.

(v) Overall, BotHunter did not perform well in detecting the remote hosts. This makes sense because the focus of this tool is on collecting evidence to find the infected machines in a known trusted network.

(vi) Given the overall flow-based detection performance in Table X, the Tranalyzer-2 flow-based system outperformed BotHunter.

Snort. As discussed earlier, Snort supports two public rule sets: VRT and ET. To run Snort, the first thing required is to determine the rule set that will be used. In this section, the VRT rule set was used because: (1) it is the official rule set for Snort which gets updated frequently, and (2) ET, is the one used by BotHunter. In this evaluation, Snort version 2.9.7.3 with the VRT rule set update on July 2, 2015 was used. Tables IX and X show the performance of Snort on the twenty-four data sets. Similar to the previous observations [24], Snort raises a lot of alerts for big data sets that contain considerable numbers of malicious traffic. This makes any post-analysis of the results very complicated. Hence, SnortSnarf was used to produce HTML output from Snort alerts. Tools like this

Table IX. Detailed Snort detection results.

Data Set	# Snort alerts	# Infected hosts	# Remote hosts	Description
Zeus-D (NIMS)	2265	1/1 (100%)	55/369 (14.9%)	INDICATOR-COMPROMISE Suspicious .cc dns query (classification: A Network Trojan was detected)
Citadel-D (NIMS)	1298	1/1 (100%)	44/87 (50.6%)	INDICATOR-COMPROMISE Suspicious .cc dns query (classification: A Network Trojan was detected)
Torpig-D (NIMS)	156	1/1 (100%)	13/60 (21.7%)	INDICATOR-COMPROMISE Suspicious .cc dns query (classification: A Network Trojan was detected)
Conficker-D (NIMS)	4064	0/1 (0%)	143/1920 (7.5%)	–
Zeus-T1-1 (NIMS)	789	12/12 (100%)	2/2 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus-T1-2 (NIMS)	19199	12/12 (100%)	2/2 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus-T1-W (NIMS)	756	12/12 (100%)	2/2 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus-T2 (NIMS)	6064	12/12 (100%)	2/2 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus-T2-W (NIMS)	766	12/12 (100%)	2/2 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Citadel-T1 (NIMS)	29845	12/12 (100%)	1/1 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Citadel-T1-W (NIMS)	1623	12/12 (100%)	1/1 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus (CVUT)	155763	3/3 (100%)	2800/9392 (29.8%)	MALWARE-CNC Win.Trojan.Zeus v3 DGA DNS query detected, MALWARE-CNC Win.Trojan.Zeus outbound connection (classification: A Network Trojan was detected)
Kelihos (CVUT)	79839	1/1 (100%)	9888/25671 (38.5%)	MALWARE-CNC Win.Trojan.Zeus outbound connection, MALWARE-CNC Win.Trojan.Pushdo variant outbound connection (classification: A Network Trojan was detected)
Neris (CVUT)	76590	10/10 (100%)	2184/18369 (11/.9%)	MALWARE-CNC Possible Zeus User-Agent - Download, INDICATOR-OBFUSCATION potential Javascript unescape obfuscation attempt detected, PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt (classification: A Network Trojan was detected)
NSIS (CVUT)	3390	3/3 (100%)	577/5364 (10.8%)	BLACKLIST URI request for known malicious URI (classification: A Network Trojan was detected)
Virut (CVUT)	6149	1/1 (100%)	175/1798 (9.7%)	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt, MALWARE-CNC Possible Zeus User-Agent - Download, MALWARE-CNC Win.Trojan.Bulknet variant outbound connection (classification: A Network Trojan was detected)
Rbot (CVUT)	367899	10/10 (100%)	3404/30046 (11.3%)	INDICATOR-COMPROMISE IRC message on non-standard port, MALWARE-OTHER generic IRC botnet connection, POLICY-SOCIAL IRC message (classification: A Network Trojan was detected)
ZeroAccess (CVUT)	176374	2/2 (100%)	15481/15495 (99.9%)	MALWARE-CNC Win.Trojan.ZeroAccess outbound communication (classification: A Network Trojan was detected)
Zeus (Snort)	37	1/1 (100%)	7/14 (50%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Zeus (NETRESEC)	163	1/1 (100%)	7/11 (63.6%)	MALWARE-CNC Win.Trojan.Zeus outbound connection, MALWARE-CNC Win.Trojan.FareIt variant outbound connection (classification: A Network Trojan was detected)
Kelihos (NETRESEC)	30	1/1 (100%)	9/268 (3.4%)	MALWARE-CNC Win.Trojan.Fareit variant outbound connection (classification: A Network Trojan was detected)
Cutwail (NETRESEC)	3823	1/1 (100%)	162/162 (100%)	MALWARE-CNC Win.Trojan.Pushdo variant outbound connection (classification: A Network Trojan was detected)
Citadel (NETRESEC)	225	1/1 (100%)	1/1 (100%)	MALWARE-CNC Win.Trojan.Zeus variant outbound connection (classification: A Network Trojan was detected)
Conficker (CAIDA)	7244	6457/360191 (1.8%)	430/80380 (0.5%)	(Classification: Potential Corporate Privacy Violation)
ISOT (Uvic)	102755	2/5 (40%)	2326/15000 (15.5%)	INDICATOR-COMPROMISE Suspicious .cc dns query, PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt (Classification: Attempted User Privilege Gain)

are intended for diagnostic inspection and tracking down problems given the high number of Snort alerts. SnortSnarf selected/filtered any alert with a high priority that was raised on botnet-related classes of alerts (such as [Classification: A Network Trojan was detected) Priority 1]).

In Table IX, column '# Snort alerts', '# Infected hosts' and '# Remote hosts' present the number of Snort alerts generated, the number of detected infected hosts (based on the IP address) and the number of detected remote hosts, respectively. Additionally, a description of the major Snort Alert that highlighted the botnet behaviour is provided in the 'Description' column. Conficker-D (NIMS) is the only data set for which a very specific botnet behaviour rule was not triggered by Snort. The results are summarized below.

(i) Snort performed really well in detecting the infected hosts. Specifically, it has a 100% DR for tewenty-two data sets out of the twenty-five.

Table X. BotHunter and Snort overall performance on botnet data samples.

Dataset	BotHunter Overall Detection		Snort Overall Detection	
	IP-based	Flow-based	IP-based	Flow-based
Zeus-D (NIMS)	0%	0%	15.1%	30.1%
Citadel-D (NIMS)	0%	0%	51.1%	27.9%
Torpig-D (NIMS)	0%	0%	23%	43.4%
Conficker-D (NIMS)	0%	0%	7.5%	0.07%
Zeus-T1-1 (NIMS)	100%	100%	100%	100%
Zeus-T1-2 (NIMS)	100%	100%	100%	100%
Zeus-T1-W (NIMS)	100%	100%	100%	100%
Zeus-T2 (NIMS)	100%	100%	100%	100%
Zeus-T2-W (NIMS)	100%	100%	100%	100%
Citadel-T1 (NIMS)	100%	100%	100%	100%
Citadel-T1-W (NIMS)	100%	100%	100%	100%
Zeus (CVUT)	0%	0%	29.8%	80.8%
Kelihos (CVUT)	0%	0%	38.5%	73.6%
Neris (CVUT)	0.4%	19.63%	11.9%	44.13%
NSIS (CVUT)	0.2%	0.7%	10.8%	17.8%
Virut (CVUT)	0.5%	0.47%	9.8%	6.75%
Rbot (CVUT)	0.14%	1.56%	11.4%	19.63%
ZeroAccess (CVUT)	0%	0%	99.9%	99.8%
Zeus (Snort)	0%	0%	53.3%	52.8%
Zeus (NETRESEC)	66.6%	12%	66.6%	20%
Kelihos (NETRESEC)	0%	0%	3.7%	2.4%
Cutwail (NETRESEC)	0%	0%	100%	100%
Citadel (NETRESEC)	0%	0%	100%	100%
Conficker (CAIDA)	0%	0%	1.6%	1.3%
ISOT (Uvic)	2.9%	60.3%	15.5%	26.1%

(ii) The C&C DNS-based behaviour of the domain-based generated data sets (with the '-D (NIMS)' extension) were detected by Snort for three data sets out of four in this category.

(iii) The NIMS sandbox-generated data sets [such as Zeus-T1-2 (NIMS)] were all detected as a 'Zeus botnet variant' by Snort. Although there are Citadel data sets in that category, given that Citadel is in fact a variant of the Zeus botnet, Snort could very well detect the type of botnet behaviour in these data sets. In this regard, Snort could as well recognize the type of botnet behaviour for Zeus (CVUT), Rbot (CVUT), ZeroAccess (CVUT), Zeus (Snort), Zeus (NETRESEC), Cutwail (NETRESEC) and Citadel (NETRESEC). It should be noted that Cutwail and Pushdo can be considered to be the same botnet type. By contrast, there are some of the botnets for which the type is not identified correctly (e.g. detecting Kelihos (CVUT) as Zeus or Pushdo botnets). However, recognizing the torjan behaviour of these data sets is still a promising result.

(iv) The Snort performance in detecting the remote hosts is much better than the BotHunter's since in some of the data sets there was a 100% detection rate. However, it underperformed the two flow-based detection systems in this section (namely, Tranalyzer-2-based system and FlowAF) given the overall flow-based DR shown in Table X.

4.1. Discussion and Highlights

In summary, automatic pattern discovery in large traffic data sets representing different traffic behaviours is the main advantage of the packet payload-based and flow-based systems. Regardless of whether a packet payload-based system or a flow-based system is used, having a specific feature set (pre-defined with a priori knowledge) might limit the high performance of a system under certain scenarios or situations for which it is actually designed. Hence, it should be modified when

facing different situations. This is the case for the packet payload-based approach and FlowAF as demonstrated in this work. Providing a wide range of features (with minimum a priori knowledge) to a pattern discovery algorithm (in this case C4.5), would enable the algorithm to select the proper set of features based on the scenario. Presumably, this is the main reason behind the consistency of high performance in the Tranalyzer-1 and Tranalyzer-2 flow-based detection systems.

BotHunter correlates Snort alerts corresponding to the botnet lifecycle to find the profiles of the infected machines. Hence, when the Snort rule set is not updated properly based on the type of botnet or when the specific phases of the botnet lifecycle are not presented in the data, BotHunter is not able to detect the bots. 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. Having said this, if the pre-defined customized Snort rules (defined with a priori knowledge) cannot detect the botnet behaviour, BotHunter would not be able to work properly.

On the other hand, Snort rule sets are updated more frequently and they are not just focussed on detecting the infected hosts based on tracing the existence of the botnet lifecycle. Instead, Snort monitors all network communications and flags any suspicious communication that matches its pre-defined rules (using the VRT rule set). Hence, the performance of Snort depends on the quality of the rules generated based on a priori knowledge. In the evaluations, Snort outperformed BotHunter in the detection of the bot machines as well as the remote hosts but it did not perform better than the flow-based systems.

5. CONCLUSIONS

Botnets are considered as one of the main security threats on the Internet due to their high reported infection rate, extensive range of malicious activities with active update capabilities. Hence, the need for botnet detection methods that can adapt to the botnet evolution is very important. To this end, many botnet detection approaches based on data analytics applying network traffic analysis are proposed in the literature. The objective of this work is to investigate the effectiveness of various detection approaches under different scenarios (e.g. multi/single-class classification, with/without using IP addresses and port numbers), anticipating that the extended analysis would assist in setting guidelines on how to select a suitable detection system. To this end, five 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 using Tranalyzer-2 flow exporter. The third system is also a traffic flow based system which uses a time interval to generate the flows (called FlowAF). Since the features used by these systems are extracted from only the header section of the packets, these approaches 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. Except for Tranalyzer-2 flow-based system which uses minimum a priori information, the other four systems use a priori information to pre-define the packet features, flow features or the rule sets.

The evaluation of all five systems on twenty-five publicly available data sets showed that the Tranalyzer-2 flow-based system and FlowAF outperformed the other three systems. Further analysis of the the performance of these two systems under binary-class and multi-class scenarios were performed as well as excluding the IP addresses and port numbers from the FlowAF feature sets. The results showed that the Tranalyzer-2-based system outperformed the original FlowAF system (using the IP address and port number information) under a multi-class scenario (different types of botnets are used/seen in real life on the Internet) without even using the IP address and port number information. It appears that not using such information as well as using a wide range of flow features (defined based on minimum a priori knowledge) for botnet detection purposes may increase the generalization capability of the Tranalyzer-2 flow-based system as demonstrated by the evaluations in this research. Moreover, the results suggest that inter-arrival based features are the

most important features that are selected and used by the C4.5 to detect botnets. It seems to imply that even recent botnets that are de-centralized HTTP-based and P2P botnets have not been able to mimic temporal characteristics of normal user behaviour.

6. ACKNOWLEDGMENTS

This research is partly supported by NSERC and 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://labs.snort.org/papers/zeus.html.
2. NETRESEC repository: publicly available pcap files. http://www.netresec.com/?page=PcapFiles.
3. Snort. https://www.snort.org/.
4. Tranalyzer. http://tranalyzer.com/.
5. E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.
6. H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Eighth Annual International Conference on Privacy, Security and Trust*, 2010.
7. CAIDA Conficker. http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml.
8. Z. B. Celik, J. Raghuram, G. Kesidis, and D. J. Miller. Salting public traces with attack traffic to test flow classifiers. In *Cyber Security Experimentation and Test (CSET)*, 2011.
9. Youksamay Chanthakoummane, Saiyan Saiyod, Nunnapus Benjamas, and Nattawat Khamphakdee. Improving intrusion detection on snort rules for botnets detection. In *Information Science and Applications (ICISA)*, 2016.
10. Christian Rossow Christian J. Dietrich and Norbert Pohlmann. Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks*, 57, 2013.
11. J. Francois, Sh. Wang, R. State, and Th. Engel. Bottrack: tracking botnets using netflow and pagerank. *Networking*, 6640:1–14, 2011.
12. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-44, 45, 51 and 52. https://mcfp.felk.cvut.cz/publicDatasets/, 2011.
13. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-54. https://mcfp.felk.cvut.cz/publicDatasets/, 2011.
14. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-28. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-28/, 2013.
15. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-3. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-3/, 2013.
16. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-42, 43 and 50. https://mcfp.felk.cvut.cz/publicDatasets/, 2013.
17. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-5. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-5/, 2013.
18. S. Garcia. Malware capture facility project, cvut university– ctu-malware-capture-botnet-53. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-53/, 2013.
19. S. Garcia. Malware capture facility project, cvut university. https://agents.fel.cvut.cz/malware-capture-facility, February 2013.
20. S. Garcia, M. Grill, J. Stiborek, and A Zunino. An empirical comparison of botnet detection methods. *Computers and Security*, 45:100–123, 2014.
21. G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: clustering analysis of network traffic for protocol- and structure- independent botnet detection. In *17th USNIX Security symposium*, 2008.
22. G. Gu, Ph. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: detecting malware infection through ids-driven dialog correlation. In *16th USENIX Security Symposium on USENIX Security Symposium*, 2007.
23. Hachem Guerid, Karel Mittig, and Ahmed Serhrouchni. Collaborative approach for inter-domain botnet detection in large-scale networks. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 2015.
24. F. Haddadi, , D. L. Cong, L. Porter, and A. N. Zincir-Heywood. On the effectiveness of different botnet detection approaches. In *Information Security Practice and Experience (ISPEC)*, 2015.
25. F. Haddadi, H.G. Kayacik, A.N. Zincir-Heywood, and M.I. Heywood. Malicious automatically generated domain name detection using stateful-sbb. In *EvoApplication*, 2012.
26. F. Haddadi, Duong-Tien Phan, and A. N. Zincir-Heywood. How to choose from different botnet detection systems? In *Analytics for Network and Service Management (AnNet)*, 2016.
27. F. Haddadi and A. N. Zincir-Heywood. Data confirmation for botnet traffic analysis. In *Foundations Practice of Security (FPS)*, 2014.
28. F. Haddadi and A. N. Zincir-Heywood. A closer look at the http and p2p based botnets from a detector's perspective. In *Foundations Practice of Security (FPS)*, 2015.

29. F. Haddadi and A. N. Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems journal*, 10:1390–1401, 2016.
30. V. Kirubavathi and R.A. 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.
31. 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.
32. W. Lu, G. Rammidi, and A. Ghorbani. Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications*, 34:502–514, 2011.
33. A. Mohaisen and O. Alrawi. Unveiling zeus. In *International World Wide Web Conference Committee (IW3C2)*, 2013.
34. R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting malicious flux service networks through passive analysis of recursive dns traces. In *ACSAC*, 2009.
35. RFC 2722. http://tools.ietf.org/html/rfc2722, October 1999.
36. M. Soysal and E. G. Schmidt. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Performance Evaluation*, 67:451.467, 2010.
37. Anna Sperotto, abd Ramin Sadre Gregor Schaffrath, Cristian Morariu, Aiko Pras, and Burkhard Stiller. An overview of ip flow-based intrusion detection. *IEEE COMMUNICATIONS SURVEYS TUTORIALS*, 12:343–356, 2010.
38. Matija Stevanovic and Jens Myrup Pedersen. An analysis of network traffic classification for botnet detection. In *Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, 2015.
39. W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. Botnet detection based on network behavior. *Advances in Information Security*, 36:1–24, 2008.
40. Jing Wang and Ioannis Ch. Paschalidis. botnet detection based on anomaly and community detection. *IEEE Transactions on Control of Network Systems*, PP, 2016.
41. K. Wang, Ch. Huang, Sh. Lin, and Y. Lin. A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks*, 55:3275–3286, 2011.
42. Weka. http://www.cs.waikato.ac.nz/ml/weka/.
43. P. Wurzinger, L. Bilge, Th. Holz, J. Goebel, Ch. Kruegel, and E. Kirda. Automatically generating models for botnet detection. In *14th European conference on research in computer security (ESORICS)*, 2009.
44. Qiben Yan, Yao Zheng, Tingting Jiang, Wenjing Lou, and Y. Thomas Hou. Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis. In *Computer Communications (INFOCOM)*, 2015.
45. Han Zhang and Christos Papadopoulos. Bottalker: Generating encrypted, customizable c&c traces. In *Symposium on Technologies for Homeland Security (HST)*, 2015.
46. J. Zhang, Ch. Chen, Y. Xiang, W. Zhou, and A. Vasilakos. An effective network classification method with unknown flow detection. *IEEE Transactions on Network and Service Management*, 10, 2013.
47. J. Zhang, R. Perdisci, U. Sarfaraz W. Lee, and Z. Luo. Detecting stealthy p2p botnets using statistical traffic fingerprints. In *Dependable Systems and Networks (DSN)*, 2011.
48. D. Zhao, I. Traore, A. Ghorbani, B. Sayed, S. Saad, and W. Lu. Peer-to-peer botnet detection based on flow intervals. In *IFIP international information security and privacy*, 2012.
49. D. Zhao, I. Traore, B. Sayed, W. Lu, andA. Ghorbani S. Saad, and D. Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers and Security*, 39, 2013.

Table XI. Classification Results

	Data Set	Score	Botnet		Legitimate		Complexity	
			TPR	FPR	TNR	FNR	Time (sec)	Solution
Packet-based	Zeus-D (NIMS)	97%	97.8%	3.8%	96.2%	2.2%	0.19	57
	Citadel-D (NIMS)	94.64%	95.2%	5.9%	94.1%	4.8%	0.12	95
	Torpig-D (NIMS)	97.8%	97.8%	2.2%	97.8%	2.2%	0.08	31
	Conficker-D (NIMS)	99.28%	98.9%	0.3%	99.7%	1.1%	0.44	37
	Zeus-T1-1 (NIMS)	100%	100%	0%	100%	0%	0.01	5
	Zeus-T1-2 (NIMS)	98.5%	98.5%	1.5%	98.5%	1.5%	0.01	5
	Zeus-T1-W (NIMS)	98.88%	99.6%	1.8%	98.2%	0.4%	0.03	5
	Zeus-T2 (NIMS)	98.45%	99.8%	2.9%	97.1%	0.2%	0.06	11
	Zeus-T2-W (NIMS)	95.8%	99.8%	8.2%	91.8%	0.2%	0.05	9
	Citadel-T1 (NIMS)	98.7%	99.9%	2.4%	97.6%	0.1%	0.07	7
	Citadel-T1-W (NIMS	100%	100%	0%	100%	0%	0.04	3
	Zeus (CVUT)	96.7%	95.6%	2.2%	97.8%	4.4%	6.03	259
	Kelihos (CVUT)	97.7%	97.2%	1.7%	98.3%	2.8%	1.3	75
	Neris (CVUT)	88.8%	82.5%	4.7%	95.3%	17.5%	1.35	73
	NSIS (CVUT)	99.53%	99.5%	0.5%	99.5%	0.5%	0.07	9
	Virut (CVUT)	90.4%	88%	7.2%	92.8%	12%	0.21	65
	Rbot (CVUT)	95.6%	93.3%	2.2%	97.8%	6.7%	0.01	5
	ZeroAccess (CVUT)	99.85%	99.7%	0%	100%	0.3%	0.08	5
	Zeus (Snort)	98.6%	98.1%	1%	99%	1.9%	0.02	5
	Zeus (NETRESEC)	96.4%	96.4%	3.6%	96.4%	3.6%	0.03	11
	Kelihos (NETRESEC)	-%	-%	-%	-%	-%	-	-
	Cutwail (NETRESEC)	96.6%	98.9%	2.8%	97.2%	4.1%	0.08	19
	Citadel (NETRESEC)	98.2%	98.2%	1.8%	98.2%	1.8%	0.02	7
	Conficker (CAIDA)	-	-	-	-	-	-	-
Tranalyzer-2 Flow-based	Zeus-D (NIMS)	99.75%	99.8%	0.3%	99.7%	0.2%	43.23	357
	Citadel-D (NIMS)	99.8%	99.8%	0.2%	99.8%	0.2%	27.19	215
	Torpig-D (NIMS)	98.55%	99.1%	2%	98%	0.9%	0.74	81
	Conficker-D (NIMS)	99.95%	99.9%	0%	100%	0.1%	152.81	151
	Zeus-T1-1 (NIMS)	99.9%	99.9%	0.1%	99.9%	0.1%	0.25	5
	Zeus-T1-2 (NIMS)	100%	100%	0%	100%	0%	14.29	11
	Zeus-T1-W (NIMS)	99.85%	100%	0.3%	99.7%	0%	0.29	11

	Zeus-T2 (NIMS)	99.9%	99.9%	0.1%	99.9%	0.1%	0.48	15
	Zeus-T2-W (NIMS)	99.95%	100%	0.1%	99.9%	0%	0.28	11
	Citadel-T1 (NIMS)	99.95%	99.9%	0%	100%	0.1%	0.99	11
	Citadel-T1-W (NIMS)	100%	100%	0%	100%	0%	0.41	5
	Zeus (CVUT)	100%	100%	0%	100%	0%	72.02	15
	Kelihos (CVUT)	100%	100%	0%	100%	0%	66.32	15
	Neris (CVUT)	99.8%	99.8%	0.2%	99.8%	0.2%	208.46	677
	NSIS (CVUT)	99.2%	99.5%	1.1%	98.9%	0.5%	5.33	177
	Virut (CVUT)	99.9%	99.9%	0.1%	99.9%	0.1%	214.48	301
	Rbot (CVUT)	99.8%	99.8%	0.2%	99.8%	0.2%	56.45	313
	ZeroAccess (CVUT)	100%	100%	0%	100%	0%	89.09	39
	Zeus (Snort)	96.88%	99.3%	5.6%	94.4%	0.7%	0.09	9
	Zeus (NETRESEC)	97.6%	97.9%	2.7%	97.3%	2.1%	0.2	33
	Kelihos (NETRESEC)	99.95%	99.9%	0%	100%	0.1%	0.39	9
	Cutwail (NETRESEC)	100%	100%	0%	100%	0%	0.5	9
	Citadel (NETRESEC)	99.85%	99.9%	0.2%	99.8%	0.1%	0.5	23
	Conficker (CAIDA)	99.85%	99.9%	0.2%	99.9%	0.1%	163.41	365
FlowAF Original	Zeus-D (NIMS)	99.95%	99.9%	0%	100%	0.1%	2.06	47
	Citadel-D (NIMS)	99.9%	99.9%	0.1%	99.9%	0.1%	79	1.24
	Torpig-D (NIMS)	99.9%	99.8%	0%	100%	0.2%	0.14	9
	Conficker-D (NIMS)	100%	100%	0%	100%	0%	45	15.02
	Zeus-T1-1 (NIMS)	99.8%	99.6%	0%	100%	0.4%	0.1	9
	Zeus-T1-2 (NIMS)	100%	100%	0%	100%	0%	0.34	5
	Zeus-T1-W (NIMS)	100%	100%	0%	100%	0%	0.07	5
	Zeus-T2 (NIMS)	100%	100%	0%	100%	0%	0.08	5
	Zeus-T2-W (NIMS)	100%	100%	0%	100%	0%	0.07	5
	Citadel-T1 (NIMS)	100%	100%	0%	100%	0%	0.12	5
	Citadel-T1-W (NIMS)	100%	100%	0%	100%	0%	0.11	5
	Zeus (CVUT)	99.9%	99.9%	0.1%	99.9%	0.1%	24.95	165
	Kelihos (CVUT)	99.95%	100%	0.1%	99.9%	0%	19.01	225
	Neris (CVUT)	99.9%	99.9%	0.1%	99.9%	0.1%	8.31	99
	NSIS (CVUT)	99.8%	99.9%	0%	100%	0.1%	0.24	5
	Virut (CVUT)	99.9%	99.9%	0.1%	99.9%	0.1%	1.47	45
	Rbot (CVUT)	99.95%	99.9%	0%	100%	0.1%	2.38	45
	ZeroAccess (CVUT)	100%	100%	0%	100%	0%	1.57	9
	Zeus (Snort)	100%	100%	0%	100%	0%	0.02	5
	Zeus (NETRESEC)	98.86%	98.6%	0.9%	99.1%	1.4%	0.05	13
	Kelihos (NETRESEC)	99.8%	99.8%	0.2%	99.8%	0.2%	0.11	9
	Cutwail (NETRESEC)	99.8%	99.7%	0.1%	99.9%	0.3%	0.15	13
	Citadel (NETRESEC)	99.7%	100%	0.5%	99.5%	0%	0.08	11
FlowAF	Zeus-D (NIMS)	99.7%	99.7%	0.3%	99.7%	0.3%	1.34	11
	Citadel-D (NIMS)	99.8%	99.8%	0.2%	99.8%	0.2%	1.26	41
	Torpig-D (NIMS)	99%	99%	0.1%	99%	0.1%	0.112	11
	Conficker-D (NIMS)	99.94%	99.9%	0%	100%	0.1%	9.31	13
	Zeus-T1-1 (NIMS)	99.8%	99.6%	0%	100%	0.4%	0.09	5
	Zeus-T1-2 (NIMS)	100%	100%	0%	100%	0%	0.08	5
	Zeus-T1-W (NIMS)	100%	100%	0%	100%	0%	0.07	5
	Zeus-T2 (NIMS)	100%	100%	0%	100%	0%	0.31	5
	Zeus-T2-W (NIMS)	100%	100%	0%	100%	0%	0.7	5
	Citadel-T1 (NIMS)	99.95%	100%	0.1%	99.9%	0%	0.1	5
	Citadel-T1-W (NIMS)	100%	100%	0%	100%	0%	0.1	5
	Zeus (CVUT)	99.67%	100%	0.6%	99.4%	0%	17.51	75
	Kelihos (CVUT)	99.65%	99.8%	0.5%	99.5%	0.2%	16.81	131
	Neris (CVUT)	97.38%	99.8%	5%	95%	0.2%	6.8	129
	NSIS (CVUT)	99.95%	99.9%	0%	100%	0.1%	0.19	11
	Virut (CVUT)	99.85%	99.7%	0%	100%	0.3%	0.89	25
	Rbot (CVUT)	99.9%	99.9%	0.1%	99.9%	0.1%	1.34	25
	ZeroAccess (CVUT)	100%	100%	0%	100%	0%	1.61	7
	Zeus (Snort)	98%	98%	2%	98%	2%	0.03	5
	Zeus (NETRESEC)	98%	98%	2%	98%	2%	0.07	11
	Kelihos (NETRESEC)	99.8%	99.8%	0.2%	99.8%	0.2%	0.1	9
	Cutwail (NETRESEC)	99.8%	99.8%	0.1%	99.8%	0.2%	0.12	7
	Citadel (NETRESEC)	99.74%	100%	0.5%	99.5%	0%	0.6	7
	Conficker (CAIDA)	99%	99%	1%	99%	1%	124.2	218