Survey on Network-based Botnet Detection Methods

Article *in* Security and Communication Networks · January 2013

1 author:

Sebastián García
Czech Technical University in Prague
35 PUBLICATIONS **46** CITATIONS

SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.800

RESEARCH ARTICLE

Survey on network-based botnet detection methods

Sebastián García*, Alejandro Zunino and Marcelo Campo

ISISTAN Research Institute-CONICET, Faculty of Sciences, Universidad Nacional del Centro de la Provincia de Buenos Aires, University Campus, Paraje Arroyo Seco (BBO7001B), Tandil, Buenos Aires, Argentina

ABSTRACT

Botnets are an important security problem on the Internet. They continuously evolve their structure, protocols and attacks. This survey analyzes and compares the most important efforts carried out in a network-based detection area. It accomplishes four tasks: first, the comparison of previous surveys and the proposal of four new dimensions to analyze their classification schemes; second, a new classification and comparison of network-based botnet detection proposals, which includes the definition of 20 desired properties of every botnet detection paper; third, an extensive comparison between the most representative detection proposals; and fourth, the description of the most important problems and highlights in the area. We conclude that the area has achieved great advances so far, but there are still many open problems. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
botnets; network detection; survey; information security; network behavior

***Correspondence**
Sebastián García, ISISTAN Research Institute-CONICET, Faculty of Sciences, Universidad Nacional del Centro de la Provincia de Buenos Aires, University Campus, Paraje Arroyo Seco (BBO7001B), Tandil, Buenos Aires, Argentina.
E-mail: sgarcia@conicet.gov.ar

1. INTRODUCTION

Botnets have been the source of most security problems on the Internet almost since 2003 [1]. The amount of attacks [2–5], digital identities stolen and computers infected [6,7] have motivated researchers to create better detection methods. For some countries, this is a matter of national security.[†]

Botnets are detected using different characteristics of the network traffic, for example, using networks statistics [8], communication protocols [9], suspicious traffic behavior [10], graphical representations of behaviors [11], actions in honeypots [12], behavioral features [13], collaborative feedback in large networks [14] and malicious actions [15]. However, botnets evolve and thus make obsolete most detection methods.

Several surveys on the botnet phenomenon have been created [16–18]. Although these surveys have helped to better understand botnets, they usually do not include an analysis of the botnet detection methods.

Our survey aims at analyzing, classifying and comparing the most relevant network-based botnet detection methods.

This survey is divided into four major parts: first, an analysis of previous surveys; second, a new classification and comparison of detection proposals; third, an analysis of detection proposals; and fourth, an analysis of the most important issues found in the area.

Our contributions can be summarized as follows:

- A novel comparison and summary of previous surveys in the area.
- A novel comparison and classification of botnet network detection characteristics and properties.
- An analysis of the *desired properties* of botnet detection papers.
- An analysis and discussion of the most important proposals.
- A discussion of open problems in the botnet detection area.

We conclude that the area has accomplished great results, but it still has open problems.

The rest of the paper is organized as follows. Section 2 describes and analyzes previous botnet detection surveys.

[†] http://www.theregister.co.uk/2011/06/14/making_hacking_tools_should_be_criminal_act_say_eu_ministers/

Section 3 proposes a novel classification of botnet detection methods. Section 4 analyzes the most relevant detection papers. Section 5 discusses the state of the area, and Section 6 presents the conclusions.

2. PREVIOUS SURVEYS

As far as we know, there are no previous surveys on botnet detection methods. However, several surveys on botnets include a brief analysis of detection methods. These surveys are analyzed in this section to describe how the detection methods are classified.

Botnet detection methods are only classified in two categories in [19]: honeynets and passive traffic.

Several data sources for botnet detection are enumerated in [16]. Also, a separation between detection techniques and measurement studies is proposed. Behavior analysis is included in the last group. In addition, the survey uses a table to relate *data sources*, *proposed techniques found in the literature* and *invariant bot behaviors*. However, these last behaviors are not deeply analyzed, and the final schema is rather confusing. For example, the *detection techniques* of the comparison table have no relation with the *detection techniques* previously discussed in the same paper. Finally, the invariant botnet properties proposed (propagation, communication and attack) are not described or defined.

The detection techniques are classified into four classes in [20]: *signature-based*, *anomaly-based*, *Domain Name System (DNS)-based* and *mining-based* techniques. This is the first survey to use *capabilities* in a comparison table of detection techniques: ability to detect unknown bots, capability of botnet detection regardless of botnet protocol, encrypted command-and-control (C&C) channels and structure, real-time detection and accuracy. Some of these are included in our survey for comparison. The main contribution is the description of four detection classes and the analysis of more than 13 papers.

The evadability of detection methods is studied in [21]. The *evasion cost* is proposed as a measure of how good each method is. This cost represents the complexity of the evasion technique and the utility lost by the botnet when the evasion technique is successful. Eight detection characteristics are proposed: Basis, Hub, Internet Relay Chat (IRC), Flow-chars, Time, Net-Det, Syntax and Taint. These characteristics are the most complete detection methods topology presented to date.

A discussion on how to use traffic measurement methods to deal with network security issues is presented in [22]. This is the first study that describes the current limitations of the intrusion detection field. It warns against four common problems: simulated datasets that are not good enough, lack of normal traffic models, detection features that tend to be overfitted and a few validity and reliability problems in the evaluation criteria. It is not included in our survey comparison table because it does not compare detection methods.

Several botnet detection and tracing methods are analyzed in [23]. They are separated into honeypot-based, IRC-based and DNS-based methods. The IRC-based category is separated into *traffic analysis-based* and *anomaly activities-based* methods.

A comprehensive botnet topology is presented in [24]. It includes infection mechanisms, C&C models and detection methods among others. However, the analysis only distinguishes between *signature-based* algorithms and *anomaly-based* algorithms.

Botnet attacks and threats are analyzed in [25]. It states the importance of discovering abnormal behaviors. These behaviors are categorized as network based, host based and global correlated. Unfortunately, no further analysis of these behaviors is performed.

A topology of network-based and anomaly-based detection systems is presented in [17]. Detection types are classified into *learnt models* (where normal behavior representation is obtained automatically) and *specification models* (where normal behavior representation is obtained manually). Learnt models detect anomalies by comparing new traffic against three cases: rules of normality, models of normality and normality statistics profiles. The models of normality use data mining, neural networks or time series analysis, among others. The specification models create models of normal network behaviors on the basis of how protocols are normally used (protocol based), different protocols states (state based) and which protocol transactions are legal (transaction based). This survey is not particularly applied to botnets, but it is closely related to common techniques in the botnet detection area.

The detection methods are separated into honeynet-based and passive traffic monitoring in [26]. Passive traffic monitoring includes the behavior-based, DNS-based and data mining detection. Furthermore, it expands the behavior-based detection to include signature-based and anomaly-based detection.

The detection sources are separated into *honeynets* and *intrusion detection system (IDS)* in [18]. *IDS* sources are separated into *signature-based* and *anomaly-based* categories. The *anomaly-based* category is separated into *host-based* and *network-based* categories. The *network-based* category is separated into *active monitoring* and *passive monitoring*. This is the only survey that proposes a classification for active botnet monitoring.

2.1. Survey comparison

The aim of our survey is to analyze and compare detection methods. However, the analysis of the previous surveys showed that it was difficult to find how they categorized the detection methods. Each survey emphasized different aspects of the papers. Therefore, it was difficult to find out a common comparison criterion.

The comparison structure of each survey was studied to extract the main *dimensions* in which information was organized. These *dimensions* are used to compare the

DOI: 10.1002/sec

previous surveys. Then, in Section 3, they are used again to help compare the botnet detection papers.

Table I shows which information is analyzed on each survey.

The following is the summary of the *dimensions*:

- Detection sources: where the base information comes from (e.g., application logs from a public network, NetFlow logs from a private network or network packets from a honeypot).
- Detection features: the characteristics used to build the topology. Papers were analyzed and classified using these features (e.g., *encrypted* botnet detection, botnet *protocol* detected and *syntax* needed for detection).
- Detection techniques: how researchers gain access to the features proposed for detection (e.g., anomaly or signature based).
- Detection algorithms: algorithms used to obtain results (e.g., Bayesian statistics and neuronal networks).

The analysis of the surveys also revealed some limitations from the perspective of classifying detection methods. These limitations helped shape our survey:

- All the surveys use different terminologies, which makes the comparison difficult. What a survey calls *method* is called *technique* in another. What a survey calls *bot* is called *botnet* in another.
- Most surveys focus on few dimensions or do not have dimensions at all. Whereas one survey includes the *botnet sources* dimension, another only describes the *botnet protocols*. The *not included* labels in Table I show this issue.

Table I. Survey comparison.

Survey	Detection sources	Detection features	Detection techniques	Detection algorithms
[16]	Network packets, DNS logs, darknet data and traffic flows	Not included	Behavior (attack and cooperative behaviors) and signature based	Not included
[21]	Not included	Net-det, Syntax, Taint, Time, Basis, Hub, IRC and Flow-chars	Not included	Not included
[20]	Not included	Unknown bots, protocol, encrypted, real time, accuracy and Net-det	Signature, anomaly, DNS and mining based	Not included
[23]	Honeynets and network packets	Not included	Signature and anomaly based	Not included
[24]	Not included	Not included	Signature and anomaly based	Not included
[19]	Honeynets and network packets	Not included	Not included	Not included
[25]	Not included	Not included	Behavior based (network and host based and global correlated)	Not included
[17]	Not included	Not included	Anomaly-based learnt models (model, rule and statistical based) and specification models (protocol, state and transaction based)	Data mining, neuronal networks, pattern matching, expert systems, Bayesian statistics, covariance, matrices, chi-squared and statistics
[26]	Honeynets and network traffic	Not included	Behavior, DNS and data mining based	Not included
[18]	Honeynets and IDS	Not included	Signature and anomaly based (host and network based [active and passive monitoring])	Not included

DNS, Domain Name System; IDS, intrusion detection system; IRC, Internet Relay Chat.

- Most surveys tend to describe the papers rather than analyze them. What a botnet detection paper stated as truth is not even doubted in other surveys.
- Most surveys do not reference enough papers to support their comparison structure.

These limitations suggest the need for a new, broader and up-to-date survey. A comprehensive topology and comparison are needed to deeply understand each detection proposal. The next section describes the details of our proposed topology.

3. CLASSIFICATION OF DETECTION PROPOSALS

This section compares the detection proposals on the basis of different points of view. The analysis carried out in previous sections allowed us to generate two comparison perspectives:

(1) In Section 3.1, the detection proposals are arranged in a topology map of network-based botnet detection characteristics.
(2) In Section 3.2, a list of the 20 desired properties of all the network-based botnet detection proposals is created.

Both perspectives cover different aspects of the proposals and contribute to the understanding of the state of the area. As far as we know, our topology map is the first one presented in the network-based botnet detection area.

In Section 3.3, these perspectives are used to compare the detection papers. To the best of our knowledge, the comparison tables in Section 3.3 are presented for the first time.

3.1. Topology map of network-based botnets detection characteristics

The first comparison perspective is a type of *topology map* where each paper can appear more than once. The map, shown in Figure 1, aims at presenting a clear view of how the detection proposals differ from each other. It allows researchers to quickly find which papers implemented each technique.

Note that in Figure 1, we used [27b] to reference the Appendix B of [27]. Also, note that some categories do not have any paper assigned. They were included because they represent well-known techniques and help to see where new papers could be assigned.

The topology map is divided into the following categories to cover the different aspects of the papers:

- Detection algorithms
  It differentiates between the type of algorithms.
  - Supervised: infers a function from supervised (labeled) training data, also known as classifiers.
  - Semi-supervised: uses both labeled and unlabeled data for training.
  - Unsupervised: finds hidden structures in unlabeled data.
  - Signal processing: analyzes signals (filtering, correlation, spectrum analysis, pattern recognition, etc.)
  - Heuristics rules: usually based on ad hoc techniques, such as manually finding the best threshold.
- Detection techniques
  It differentiates the main technique used for detection.
  - Anomaly based: uses anomaly-based techniques to detect behavior patterns.
    * Bot behavior: detects the behavior of one bot with data from one bot alone.
    * Botnet behavior: detects the behavior of a group of bots acting as a botnet (not individually).
    * Temporal behavior: detects behavior changes over time; involves time measurement.
      · Peer to peer (P2P)
      · Hypertext Transfer Protocol (HTTP)
      · IRC
      · DNS
      · Generic: detects any protocol.
      · Port scan
      · Network metrics: uses network metrics (e.g., bytes per second and packets per minute)
      · Simple Mail Transfer Protocol (SMTP)
  - Fingerprint based: uses a string or byte sequence for detection.
- Detection sources
  It refers to *where* the packets were captured and not *how* they were captured.
  - Normal packets: verified normal packets.
    * Virtual internal networks: captures verified normal packets from an internal and *controlled* network.
    * Real networks: captures packets from real networks; traffic should

DOI: 10.1002/sec

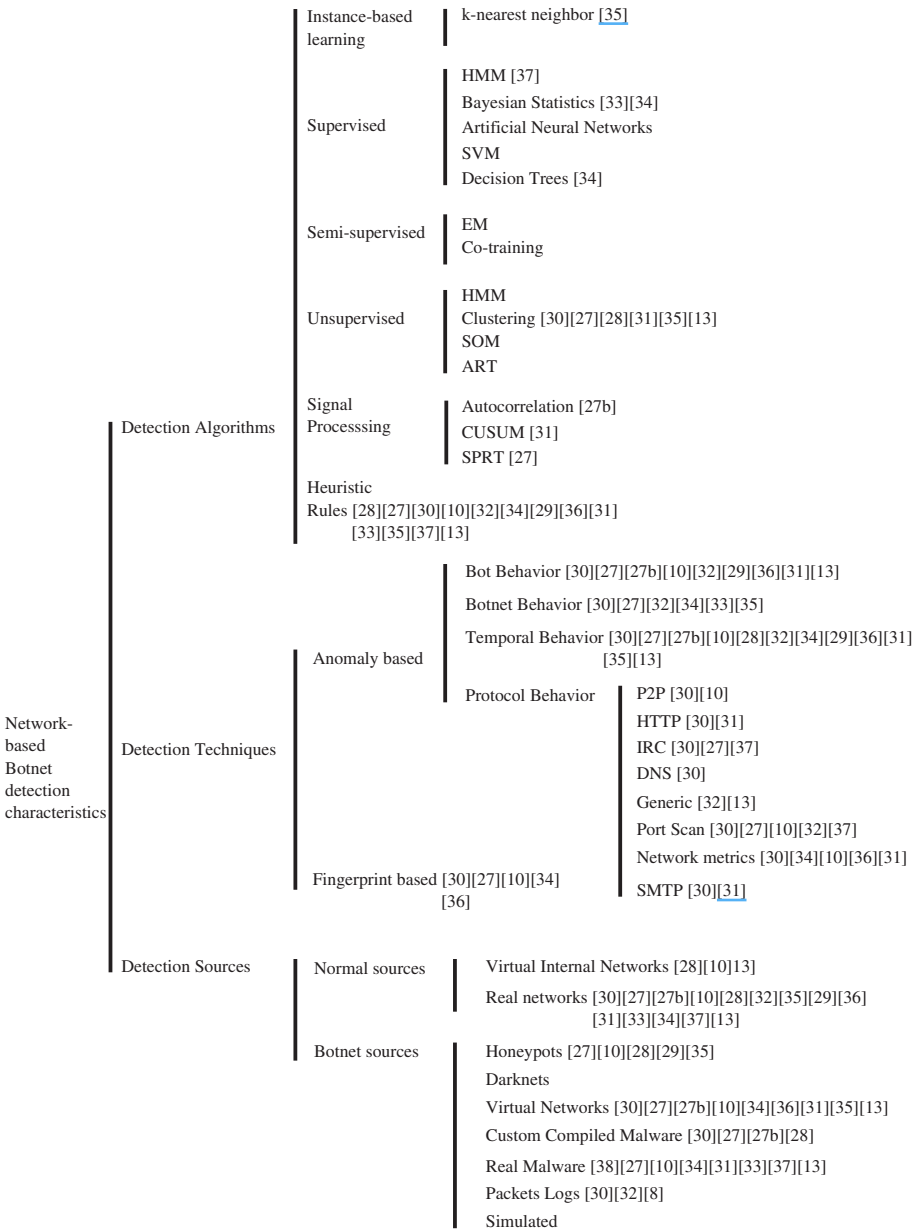

Figure 1. Topology map of network-based botnet detection characteristics. ART, adaptive resonance theory; CUSUM, cumulative sum; DNS, Domain Name System; EM, expectation–maximization; HTTP, Hypertext Transfer Protocol; HMM, hidden Markov model; Internet Relay Chat; P2P, peer to peer; SOM, self-organizing map; SMTP, Simple Mail Transfer Protocol; SPRT, sequential probability ratio test; SVM, support vector machine.

be labeled *background* if it is not verified.

– Botnet packets: refers to the source of botnet packets.

  * Honeypots: honeypots connected to the Internet.
  * Darknets: from darknets.
  * Virtual networks: from *a controlled* virtual network. Malware is manually installed.
  * Compiled malware: modified versions of known malware.
  * Real malware: uses unmodified real binary malware to infect their computers.
  * Packets logs: uses packet logs.

∗ Simulated: simulated data.

The topology map presented is useful in visualizing how each proposal tackled the botnet detection problem. However, it is not enough to understand what each paper proposed. The next subsection uses another point of view to describe the classification of papers in the topology.

3.2. Desired properties

The botnet detection proposals have a common set of properties that can be used to better understand them. For example, they can be compared on the basis of how they verify the dataset used, how they filter the data or how they report results. Most detection proposals have focused on the description of the detection method and the true positive metrics. However, they have not considered the importance of other properties, such as the training dataset verification, a complete error rate analysis, the description of the assumptions or the diversity of the training dataset. These are equal or more important than the positive results report.

This subsection describes what we consider as the 20 most important *desired properties* of a network-based botnet detection proposal. These properties help to understand and compare each proposal. They show what is missing on each paper and what is being overemphasized.

These *properties* were created partially based on ideas from previous surveys. They are separated into five groups for better visualization and description. The following are the property description:

(1) First group: verification issues
Verification methods applied to the datasets.

   (a) Training dataset diversity: how diverse are the botnet training data.
   (b) Training dataset verification: how is the botnet and normal training data verification performed.
   (c) Validation dataset verification: how is the botnet and normal validation data verification carried out.
   (d) Reproducibility: can the proposal be verified and reproduced?

(2) Second group: results issues
Results achieved in the paper, how they were obtained and expressed.

   (a) Experimental setup: how were the experiments designed?
   (b) Accuracy-based performance metrics: how well did the proposal perform?
   (c) Results comparison: were the results compared?
   (d) Capture in a host or in a network: does it need the proposal to capture from a network or a single host?

(3) Third group: theoretical background
Hypothesis and assumptions in the paper.

   (a) Paper hypothesis: how was the hypothesis verified?
   (b) Assumptions: which assumptions were made?

(4) Fourth group: detection characteristics
The main characteristics of the detection.

   (a) Statical bot detection: which bot statistical features does this proposal detect?
   (b) Unknown botnet detection: does it detect unknown botnets?
   (c) Encrypted botnet detection: does it detect botnet that use encrypted connections?
   (d) Real-time detection: does it detect botnets in real time?
   (e) Protocol-dependent features: does detection depend on a protocol?

(5) Fifth group: detection method
The detection method itself.

   (a) Preprocessing: how was the dataset prepared?
   (b) Main detection method: which is the main detection method proposed?
   (c) Differentiation from other attacks: how were other attacks, such as port scanning, differentiated?
   (d) Malicious actions detected: which malicious actions does it detect?
   (e) Automatic botnet identification method: how are botnets automatically detected?

The next subsection uses these properties to analyze and compare papers.

3.3. Paper comparison

This subsection compares the detection papers on the basis of the two previous perspectives. Tables are used for the main topics to help visualize the differences.

The comparisons performed in the following subsections are a good resource to quickly understand the motivations and contexts of the papers. They give hints about the details of the works and help to better understand the proposals.

Before the comparison, a common confusion about the use of the terms *bot detection* and *botnet detection* should be addressed. The difference becomes important when dealing with detection methods. It can be safely assumed

DOI: 10.1002/sec

Table II. Comparison of anomaly-based behavior detection techniques.

Papers	Bot behavior	Botnet behavior	Temporal behavior	Protocol behavior
BotMiner [28]	Pattern of *packets per flow* and *mean bytes per packet*; connects to many SMTP servers; ask many MX records	Same attack is ordered to every bot; synchronized attacks; bots have similar traffic patterns	*Mean bytes per second* and *flows per hour*	Port scan detection; many MX records asked; packet and byte transfer; fph, ppf, bpp and bps
BotSniffer [27]	Binary download?; send SPAM?	Attacks are synchronized; IRC answer messages are synchronized	Attacks are in the same time windows; IRC responds to PRIVMSG in the same time window	Port scan detection
BotHunter [29]	Attack signature prior knowledge	None	Time windows are used	Port scan detection
Appendix B of [27]	Connects to the C&C	None	Bot connects to HTTP C&C periodically	None
N-gram [30]	None	None	Features are computed within a time interval	Features are computed for each protocol
Stability [10]	P2P bot flows are stable in average bytes per flow	None	Botnet flows are stable within a time window	Botnet P2P protocol is stable
Models [31]	Attack signature prior knowledge; traffic patterns	None	Features are analyzed in time slices	Eight traffic features
Unclean [32]	SPAM and phishing	Botnets infect the same unclean networks	Bots appear in the same networks over time	Port scan detection
FluXOR [33]	None	Anomalous changes in the features of domains	None	None
Tight [34]	None	Botnet controls bots at the same time	C&C server sends orders at the same time	bpp, bps and pps
Tamed [35]	None	Infect the same OSs; bots use the same C&C; bots use the same payload	Use time windows	None
Incremental [36]	Generates same traffic in different time windows	None	Traffic is similar in different time windows	None
Markov [37]	None	None	None	Port scan detection
Synchronize [13]	Bots synchronize their traffic	None	Traffic is aggregated in time windows	None

bpp, average bytes per packets; C&C, command and control; fph, flows per hour; HTTP, Hypertext Transfer Protocol; IRC, Internet Relay Chat; MX, mail exchanger; OSs, operating systems; P2P, peer to peer; ppf, packets per flow; SMTP, Simple Mail Transfer Protocol.

that the methods needed to detect a whole botnet are different from the methods needed to detect one infected computer alone. When we try to detect a bot alone, there is less traffic to analyze, and there is no synchronization or correlation with other bots. Although the detection techniques can be similar, the design and goal of the techniques are different.

3.3.1. Comparison of anomaly-based behavior detection techniques.

Table II shows the classification of papers in the anomaly-based subcategory of the detection techniques category of the topology map. Data in this table were found after an in-depth analysis of the proposals, because most of them did not show this information explicitly. The relevance of each value in Table II is different. For example, in *N*-gram [30], the features are computed within a time interval, and thus, it is considered that it uses temporal behaviors. However, time windows are used only to avoid the continuous computation of the features, and thus, they are not so important. In contrast, in BotHunter [29], time windows are a major part of the proposal. Without them, it would not work properly.

Table III clearly shows which protocol is used on each proposal.

3.3.2. Detection sources comparison.

Tables IV and V show a comparison of normal and botnet data sources for detection. Each proposal trains or verifies its methods using one or more datasets. The

Table III. Botnet protocol detected.

Papers	HTTP	IRC	P2P	Generic
BotMiner [28]	√	√	√	
BotSniffer [27]	√	√		
BotHunter [29]		√		
Appendix B of [27]	√			
N-gram [30]		√		
Stability [10]			√	
Models [31]	√	√	√	
Unclean [32]				√
FluXOR [33]				√
Tight [34]		√		
Tamed [35]	√	√		
Incremental [36]		√	√	
Markov [37]				√
Synchronize [13]				√

HTTP, Hypertext Transfer Protocol; IRC, Internet Relay Chat; P2P, peer to peer.

design of the capture methodology and the origin of these datasets are very important to understand the conditions under which the method was validated.

In these tables, the term *compiled* is used with reference to binary bots compiled by the authors from public source codes. Compiled malware has to be configured before being used. For example, the C&C server and the encryption passwords must be set. Consequently, these malware are not equal to the ones in the wild. However, the implications of using modified malware are normally underestimated. Most proposals do not describe the modifications carried out. The use of custom-compiled malware can be a good approach if the limitations are clearly stated.

These tables also use the term *virtual* with reference to the use of virtual machines or virtual networks to execute the binaries. The use of virtualization technology is commonly accepted for malware execution, but it should be noted that some malware detects virtual machines to avoid being executed. This could potentially bias the type of malware that the proposal could analyze.

Finally, the terms *training* and *testing* are used with reference to the training and testing phases of the method, respectively. When only one of these labels appear, it means that the paper did not consider or need the other phase.

The analysis of Tables IV and V shows that most proposals lack some type of dataset. If an algorithm needs to be trained, at least independent training and testing datasets should be used, and a validation dataset is highly recommended. Furthermore, few proposals have training and testing datasets with both normal and botnet captures.

These tables show that some proposals use the same datasets for training and testing. This practice should be avoided for statistical reasons [38]. The tables also help to compare the amount of data used on each proposal. Whereas some methods used only one bot binary in a virtual network, others captured more than 15 bot families in several universities.

The overall amount of data sources used during training and testing can be seen in Figures 2 and 3, where the gray-scale corresponds to each detection method. Two conclusions are presented from the analysis of the graphics. First, most methods prefer controlled environments over real networks to capture botnets. This happens during both the training and testing phases. However, the implications of this decision, such as the type of attacks not captured,

Table IV. Comparison of normal sources for detection.

Papers	Normal packets
BotMiner [28]	Testing: 10-day university campus
BotSniffer [27]	Testing: university campus IRC; university campus complete
BotHunter [29]	Testing: university campus, 100 Mb/s (two captures); production public/17 network (10 days)
Appendix B of [27]	Testing: one HTTP-only capture (~17 GB); four full data capture (~33 GB)
N-gram [30]	Training: 1-h Wi-Fi traffic
	Testing: 1-h Wi-Fi traffic; one virtual IRC (60 clients); one real IRC public network
Stability [10]	Training: four complete university; two virtual P2P traffic
	Testing: Same as training
Models [31]	Testing: 1 /21 university network; 1 /20 university network
Unclean [32]	Training: 47 million public network IPs
	Testing: Same as training
FluXOR [33]	Training: 50 SPAM domains and normal mails
	Testing: Same as training
Tight [34]	Training: Crawdad Wi-Fi data
	Testing: Same dataset
Tamed [35]	Testing: two university /16 nets
Incremental [36]	Testing: internal network capture
Markov [37]	Training and testing: 1 month of 11 C-class university nets
Synchronize [13]	Testing: one host port scanning, two hosts (7-h normal usage)

HTTP, Hypertext Transfer Protocol; IP, Internet Protocol; IRC, Internet Relay Chat; P2P, peer to peer.

Table V. Comparison of botnet sources for detection.

Papers	Botnet packets
BotMiner [28]	Testing: three compiled virtual IRC (four clients each); one IRC real log (259 clients); two HTTP compiled virtual (four and one client); two P2P virtual (82 and 13 clients)
BotSniffer [27]	Testing: one honeynet-captured IRC bot; three compiled virtual IRC (five clients and two with four clients); two compiled virtual HTTP (four clients and one client); two real IRC text logs
BotHunter [29]	Testing: 10 IRC bots in a virtual network; honeynet: 26 hosts; 2019 infections in 3 weeks; unknown amount of bots
Appendix B of [27]	Testing: four compiled virtual HTTP bots (one client each); one compiled virtual HTTP bot (four clients); one compiled virtual HTTP bot (one client)
N-gram [30]	Training: one IRC botnet from a honeypot (10 clients); one compiled internal virtual IRC bot (60 clients) Testing: one IRC botnet from a honeypot (10 clients); one compiled internal virtual IRC bot (60 clients)
Stability [10]	Training: honeynet P2P storm bot; 13 different versions Testing: Same as training
Models [31]	Training: two third-party IP text report, two public NetFlow logs, bot IPs from IRC logs Testing: bot IPs (unknown source)
FluXOR [33]	Training: 75 SPAM mail domains Testing: training dataset with cross-validation; domains from Web browsing
Tight [34]	Training: internal virtual network, one bot (74 traces) Testing: same as training
Tamed [35]	Training: four virtual bots (21 traces), three honeynet botnets
Incremental [36]	Testing network captures: three internal virtual bots, four third-party virtual P2P bots
Markov [37]	Training and testing: same as normal
Synchronize [13]	Testing: two virtual bots (21 h); five compiled bots in an 18-host university network (one botnet family)

HTTP, Hypertext Transfer Protocol; IP, Internet Protocol; IRC, Internet Relay Chat; P2P, peer to peer.

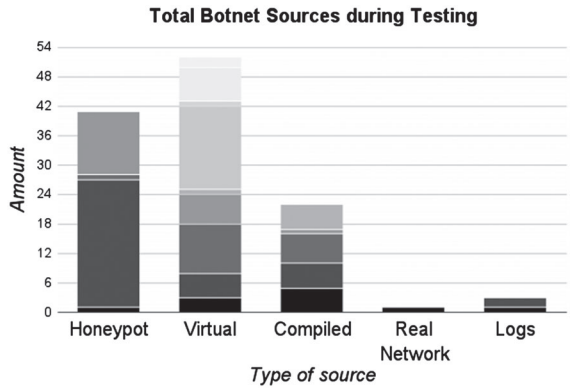

Figure 2. Sum of botnet sources for all the experiments during testing. Gray scale corresponds to each method.

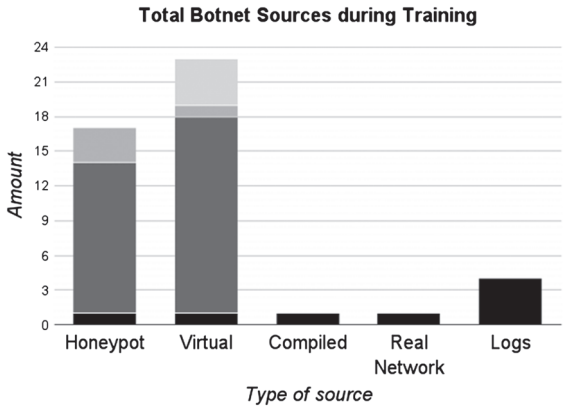

Figure 3. Sum of botnet sources for all the experiments during training. Gray scale corresponds to each method.

have not been estimated. Second, training phases use much less sources than testing phases. This is probably due to the unsupervised methods used.

3.3.3. Detection algorithms comparison.

Table VI shows a comparison of the algorithms and techniques used on each paper. It was rather difficult to create this table because some proposals did not describe the algorithms explicitly. All the papers have used heuristic-based rules at some point of the analysis. The (sup) reference means supervised technique, the (unsup) reference means unsupervised technique and the (sp) reference means signal processing.

3.3.4. Accuracy-based performance metrics comparison.

Accuracy-based performance metrics, as stated in [39], can be measured using multiple techniques. Table VII describes the definition and explanation of the two-class problem metrics. It helps to better understand what the papers reported:

- *TP*, or true positives, are the number of times the model predicts positive when the example label is positive.

Table VI. Detection algorithms comparison.

Papers	Algorithms
BotMiner [28]	*X*-means and hierarchical clustering (unsup)
BotSniffer [27]	SPRT, threshold random walk (sp), hierarchical clustering (unsup)
BotHunter [29]	—
Appendix B of [27]	Autocorrelation (sp)
N-gram [30]	*K*-means and *X*-means clustering (unsup), decision tree (sup)
Stability [10]	—
Models [31]	CUSUM (sp), hierarchical clustering (unsup)
Unclean [32]	—
FluXOR [33]	Naïve Bayes classifier (sup)
Tight [34]	Naïve Bayes, j48 decision trees, Bayesian networks (sup)
Tamed [35]	*K*-means clustering (unsup)
Incremental [36]	—
Markov [37]	HMM (sup)
Synchronize [13]	EM clustering (unsup)

CUSUM, cumulative sum; EM, expectation–maximization; HMM, hidden Markov model; SPRT, sequential probability ratio test, sp, supervised; unsup, unsupervised.

Table VII. Two-class problem quantities.

		Predicted	
		+	–
Actual	+	*TP*	*FN*
	–	*FP*	*TN*

FN, false negative; FP, false positive; TN, true negative; TP, true positive.

- *FN*, or false negatives, are the number of times the model predicts negative when the example label is positive.
- *FP*, or false positives, are the number of times the model predicts positive when the example label is negative.
- *TN*, or true negatives, are the number of times the model predicts negative when the example label is negative.

Metrics can be described on the basis of these definitions:

- *Per cent correct* or *accuracy* is the portion of the test examples that the model predicts correctly: $(TP + TN)/(TP + FN + FP + TN)$.
- *Error rate* is the portion of the examples in the test set that the model predicts incorrectly: $(FN + FP)/(TP + FN + FP + TN)$.
- *Precision* is the portion of the test examples predicted as positive that were really positive: $TP/(TP + FP)$.
- *True-positive rate* (TPR) or *recall* is the portion of the positive examples that the model predicts correctly: $TP/(TP + FN)$.
- *True-negative rate* (TNR) is the portion of the negative test examples that the model predicts correctly: $TN/(FP + TN)$.
- *False-negative rate* (FNR) is the portion of the positive test examples that the classifier predicts falsely as negative: $FN/(TP + FN)$. Also $1 - TPR$.
- *False-positive rate* (FPR) is the portion of the negative test examples that the classifier predicts falsely as positive: $FP/(FP + TN)$. Also $1 - TNR$.
- *F-measure* is the harmonic mean of precision and recall. The balanced equation is $2*Precision*Recall/(Precision + Recall)$.

In [39], it is stated that other researchers have made the case that evaluations that use accuracy metrics are problematic, for they do not show how well a model predicts instances by class. Maloof [39] also stated that "if a testing set contains many more negative examples than positive examples, high accuracy could be due to the model's exceptional performance on the majority". Data in Table VIII and the analysis of the *training dataset diversity* extracted from Tables V and IV should help to evaluate this problem.

Table VIII shows that none of the proposals reported all the four basic metrics. If a proposal only reports the TPR value, it is incredibly difficult to understand the significance of its results. Moreover, the table shows that two of the papers did not report any performance metrics at all.

Most of the metrics discussed in [39] were not reported in the papers. In consequence, whenever possible, we computed some of these values ourselves. Our calculations of the missing metrics can be prone to errors, so the labels used in Table VIII are shown in the following paragraphs. They help to better distinguish the original values from the calculated values.

References for the computed, non-original values were as follows:

- *G*: Some metrics were reported for some experiments. However, no final value was reported for all the experiments. If *G* appears with a percentage number, it

Table VIII. Accuracy-based performance metrics comparison.

Papers	FPR (%)	FNR (%)	TPR (%)	TNR (%)	Percent correct	Error rate (%)	F-measure (%)
BotMiner [28] (A)	0.1875 (G)	—	96.82 (G)	81.25 (D)	—	—	—
BotSniffer [27]	0.1600 (G)	—	100.00 (G)	84.00 (D)	—	—	—
BotHunter [29]	—	(G)	(G)	—	—	—	—
Appendix B of BotSniffer [27] (A)	(G)	—	(G)	—	—	—	—
N-gram [30] (A)	8.1400 (D)	1.64 (D)	98.36 (G)	91.86 (G)	—	—	—
Stability [10]	0.0000 (D)	0.00 (D)	100.00 (G)	100.00 (G)	—	—	—
Models [31] (B)	—	—	88.00	—	—	—	—
Unclean [32]	1.2100 (G)	1.22 (D)	98.78 (G)	87.90 (D)	—	—	—
FluXOR [33]	0.0000	—	—	100.00 (D)	—	—	—
Tight [34]	15.0400 (G)	2.49 (G)	75.10 (D)	84.96 (D)	—	—	—
Tamed [35] (A)	—	6.25 (D)	93.75 (G)	—	—	—	—
Incremental [36] (A)	14.1500 (G)	0.00 (G) (B)	100.00 (G)	85.85 (D)	—	—	—
Markov [37] (A)	13.0000 (G, B)	11.00 (G, B)	88.00 (G, B)	86.00 (G, B)	87.00 (G, B)	51.0 (G, B)	96.00 (G, B)
Synchronize [13] (A)	0.7000	3.40	95.86 (G)	95.87 (G)	95.87 (G)	4.1 (G)	97.44 (G)

FNR, false-negative rate; FPR, false-positive rate; TNR, true negative rate; TPR, true-positive rate.

means that we could manually compute an averaged total value by closely analyzing the paper (if more than one method was used, we use the best result). The *G* alone means that we could not deduce a total value from the isolated experiments.

- —: No value or computation was reported.
- *A*: This paper does not have an automated method to decide whether it detected a botnet or not. Results were obtained by counting the data labels.
- *B*: Results were reported, but details about the evaluation process were not given. We could not assess if the numbers were correctly computed.
- *D*: No value was reported for this metric, but we could calculate some approximation as the difference with its counter value. $FPR = 1 - TNR$, $FNR = 1 - TPR$, $TNR = 1 - FPR$ and $TPR = 1 - FNR$.

Table VIII is one of the most important outcomes of this survey. It clearly shows that some results have not been properly computed. None of the proposals reported all the values. Only one proposal used an alternative method to report results, such as receiver operating characteristic curves [37]. Most of the papers did not compute a total value for each metric (where the G reference was used). Instead, they reported separate values for each experiment. Furthermore, most of the papers did not compute some values at all, forcing us to calculate the missing metrics using its counterpart value (the D reference). Finally, none of the papers reported the percentage correct and the error rate metrics.

3.3.5. Unknown botnet detection capability comparison.

Table IX compares unknown botnet detection capabilities, that is, the ability to detect botnets that were not present in the training dataset. For several reasons, this is perhaps the most difficult *desired property* to extract: first, because most papers do not describe which botnets were used on each dataset; second, because most of the proposals were not designed to detect unknown botnets and therefore they did not describe this ability; and third, because the *unknown botnet* concept can be analyzed from different perspectives. An unknown botnet can be considered when the following occur:

Table IX. Unknown botnet detection comparison.

Papers	HTTP	IRC	P2P	Generic
BotMiner [28]	⊖	⊖	⊖	⊛*
BotSniffer [27]	⊖	⊙ †	⊖	⊖
BotHunter [29]	⊖	⊙ ‡	⊖	⊖
Appendix B of [27]	⊖	⊖	⊖	⊖
N-gram [30]	⊖	⊖	⊖	⊖
Stability [10]	⊖	⊖	⊙	⊖
Models [31]	⊙	⊙	⊙	⊖
Unclean [32]	⊖	⊖	⊖	⊖
FluXOR [33]	⊙	⊙	⊙	⊙ §
Tight [34]	⊖	⊙	⊖	⊖
Tamed [35]	⊙	⊙	⊖	⊖
Incremental [36]	⊖	⊖	⊖	⊖ ¶
Markov [37]	⊙	⊙	⊙	⊙ ‖
Synchronize [13]	⊙	⊙	⊙	⊙ **

HTTP, Hypertext Transfer Protocol; IRC, Internet Relay Chat; P2P, peer to peer.

*If they scan ports or use a known attack and at the same time show a network pattern correlation with other bots.

†For botnets that also scan ports or attack.

‡If they show two distinct outbound port scans.

§If they use fast-flux techniques.

¶The evaluation and method are not clear.

‖If they scan Transmission Control Protocol ports.

**If they use Transmission Control Protocol.

Table X. Comparison of protocol-dependent features.

Papers	HTTP	DNS	SMTP	TCP	ICMP	UDP	IRC	IDS	P2P	OS
BotMiner [28]	—	√	—	—	—	—	—	—	—	—
BotSniffer [27]	√	√	√	√	—	—	—	—	—	—
BotHunter [29]	—	—	—	—	—	—	√	√	—	—
Appendix B of [27]	√	—	—	—	—	—	—	—	—	—
N-gram [30]	√	√	√	√	√	√	√	—	√	—
Stability [10]	√	√	—	√	—	√	—	—	√	—
Models [31]	√	—	√	—	—	—	—	—	—	—
Unclean [32]	—	—	—	—	—	—	—	—	—	—
FluXOR [33]	—	—	—	—	—	—	—	—	—	—
Tight [34]	—	—	—	√	—	—	√	—	—	—
Tamed [35]	—	—	—	√	—	√	—	—	—	√
Incremental [36]	—	—	—	√	√	√	—	—	—	—
Markov [37]	—	—	—	√	—	—	—	—	—	—
Synchronize [13]	—	—	—	√	—	—	—	—	—	—

DNS, Domain Name System; HTTP, Hypertext Transfer Protocol; ICMP, Internet Control Message Protocol; IDS, intrusion detection system; IRC, Internet Relay Chat; OS, operating system; P2P, peer to peer; SMTP, Simple Mail Transfer Protocol; TCP, Transmission Control Protocol; UDP, User Datagram Protocol.

- The protocol is unknown.
- The attack is unknown.
- The family is unknown, which implies unknown protocol, C&C servers and attacks.
- The variant is unknown, which implies unknown attacks or unknown C&C servers but known protocol.

These issues made it difficult to decide whether a method could detect unknown botnets. However, this property is crucial to know if other researchers can use the proposal safely.

Table IX shows how likely it is that the method detects an unknown botnet. The analysis of this table reveals that few proposals are prepared to detect unknown botnets. Most papers have strong constraints, for example, only analyzing IRC botnets showing suspicious behavior. It is worth noting that two proposals seem capable of detecting unseen botnets almost without constraints [28,31]. However, none of them have carried out experiments to confirm this capability. The references for Table IX are as follows: ⊖ means there are no chances of detecting this type of unknown botnet, ⊙ means there are good chances of detecting this type of unknown botnet and ⊛ means there is a high probability of detecting this type of unknown botnet. The conditions that should be fulfilled to detect unknown botnets on each paper are explained in the footnotes.

3.3.6. Comparison of protocol-dependent features.

Most of the proposals base their detection methods on the recognizance of statistical protocol features. Most of the times, this is not a bad practice. However, depending too much on it could lead to a rigid method. Usually, the proposals do not explicitly analyze this dependence. Also, they tend to assume that their statistical features can find all the traffic they are trying to capture. For example, several proposals detect the IRC protocol by capturing only the traffic using the Transmission Control Protocol (TCP) port 6667. However, it is not uncommon to find IRC servers working on other ports as well.

Table X shows which protocols are statically detected on the proposals. Most of them use the TCP or User Datagram Protocol (UDP) port numbers to filter the protocols. The IDS column identifies the use of statical IDS rules. The operating system column identifies the detection of the operating system with statical rules. The analysis of Table X shows that several proposals detect botnets using only the TCP. However, botnets have been reported to use UDP-based protocols as well. The exact details about these statical filters are explained on each proposal subsection in Section 4. This table helps us understand how the proposals were designed and also know how to avoid being detected by such methods.

3.3.7. Dataset diversity comparison.

This category refers to the various datasets that were captured and the types of botnets analyzed. It is important to have a diverse dataset because every botnet has unique characteristics. The applicability of each proposed method could be analyzed by studying which botnet was included in the training and testing datasets. Tables IV and V shows that several proposals only used one or two botnets for training and testing. Others used the same dataset for training and testing. This last practice should be avoided for statistical reasons [38].

In the next section, each method is described in depth.

4. DETAILED ANALYSIS OF PAPERS

Previous sections described a topology map of detection characteristics, a set of desired properties of botnet detection papers and a set of tables to compare the detection proposals. This section analyzes the context on which each

Table XI. Generic method comparison.

Papers	Verify	Preprocess	Compare
BotMiner [28]	⊙	⊙	⊖
BotSniffer [27]	⊙	⊙	⊖
BotHunter [29]	⊙	⊖	⊖
Appendix B of [27]	⊙	⊙	⊖
N-gram [30]	⊖	⊙	⊖
Stability [10]	⊙	⊛	⊖
Models [31]	⊙	⊛	⊖
Unclean [32]	⊖	⊙	⊖
FluXOR [33]	⊛	⊙	⊖
Tight [34]	⊙	⊛	⊖
Tamed [35]	⊛	⊙	⊖
Incremental [36]	⊙	⊛	⊖
Markov [37]	⊙	⊛	⊖
Synchronize [13]	⊛	⊛	⊖

proposal was created. It also summarizes the steps of each detection method and highlights each proposal difficulty, bias, assumption, hypothesis, advance, contribution and result. Each paper is described and analyzed in the next subsections.

Table XI shows a generic comparison of all the proposals before their detailed analysis. It compares three properties: verify, preprocess and compare. Verify refers to the verification of the datasets. Preprocess refers to the preprocessing of the datasets. Compare refers to the comparison of the results with those of other proposals. Each property can be fully developed (⊛), moderately developed (⊙) or not elaborated at all (⊖). Therefore, a quick comparison can be performed before reading the description. The analysis text of each proposal further describes these properties.

4.1. BotSniffer

In the BotSniffer [27] paper, three different botnet detection approaches are presented. The first two are analyzed in this subsection and the third one in Section 4.2. Figure 4 shows the architecture of the BotSniffer proposal.

The first two detection approaches share a common starting base: sniff network packets group together hosts that connect to the same remote server and port and separate these groups in time windows. With these data, the first approach looks for hosts that had triggered some attack fingerprint (such as SPAM sending, binary downloading and port scanning). If more than half of the group had performed attacks, then the group is marked as a possible botnet. The sequential probability ratio testing algorithm is used to decide whether it is a botnet.

The second proposal looks for hosts that answered similar IRC protocol responses using the DICE[‡] distance as a similarity function. IRC responses are clustered on the basis of this similarity measure. If the biggest cluster is more than half the size of the group, then the group is marked as a possible botnet. The sequential probability ratio testing is used again to decide whether the group is really a botnet.

This paper is one of the most cited papers in the botnet detection field. It presents several behavioral techniques to detect botnets that accomplish good results. However, much new data and botnets have been found since its publication.

The dataset used for validation may be too scarce for generalizing the technique. Only one real IRC botnet was captured. The rest of the dataset is composed of one IRC text log and five custom-compiled LAN botnets. The dataset was verified using three methods, but the first two are not effective. The first verification was under the assumption of a clean normal capture because of a well-administered network. However, even well-administered networks can have infected computers. The second was the use of the same *BotSniffer* method over the dataset. It is commonly not considered a good practice to verify a dataset using the same proposed method. The third verification was performed using the BotHunter [29] proposal. This was a good verification. However, it was only used over the normal captures.

During the preprocessing stage, hard and soft whitelists were used to filter the dataset. Unfortunately, there is no analysis of the bias introduced by these lists. For example, it was found that some nonbotnet Web sites made synchronic requests that are similar to a botnet request. This possible interference was solved by whitelisting the Web sites. Protocols different than TCP were filtered out.

Performance metrics were computed for FPRs and TPs (not TPR). However, FNR and TN metrics were not computed. The missing metrics could be a consequence of the difficulty to have a labeled dataset. This incompleteness made it difficult to compare the reported results with those of other papers.

The FPR was computed over a dataset that only had normal labels. Under some circumstances, such a metric could be biased. Normally, it is recommended to compute the FPR with a dataset having both normal and botnet labels. Using both labels could be important, as the FPR calculation needs to divide by the total instances analyzed. This issue could be caused by the difficulty of obtaining a labeled dataset.

On the other hand, a mixed capture was used to indicate the TPR. It would have been very helpful to know exactly how the mixture was carried out. In addition, there was no analysis of FNRs or TNs.

The density-check method is based on the assumption that IRC botnets use very similar messages and that it is unlikely that humans write very similar messages. The results obtained support this idea.

The algorithm assumes that the more humans captured in an IRC channel, the more randomness involved. Therefore, it could be less likely to obtain a homogeneous crowd,

[‡] https://secure.wikimedia.org/wikipedia/en/wiki/Dice's_coefficient

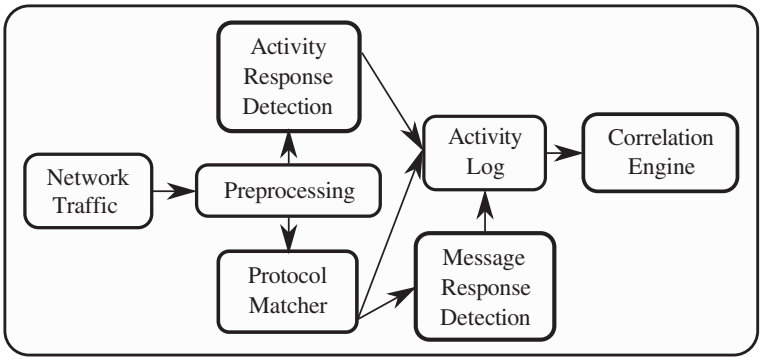

Figure 4. BotSniffer architecture.

that is, more users are less likely to talk similarly than few users. However, in [40], it was stated that "Stability is an important and noteworthy feature of human action (...)." This may support the idea that more users are more likely to talk similarly. Furthermore, we believe that this assumption of the proposal was the main reason for the 11 FPs in the experiments.

It was also assumed that the TCP would be the primary protocol used for C&C channels. However, it could be important to add the UDP or Internet Control Message Protocol as well.

The dataset used was not made public, perhaps because of privacy issues. This made it impossible to reproduce the method.

It was not within the goals of the paper to compare botnet attacks and manually automated attacks. However, such a comparison may help to better evaluate the results obtained.

The proposal is stated to have implemented a real-time correlation engine, but there was no analysis or further mention.

The detection conditions of the proposal need traffic from more than one host to detect botnets.

The paper uses a method called threshold random walk (TRW) within the *response crowd activity check* algorithm and within the *response crowd density check* algorithm to decide whether there is strong evidence of a botnet infection. It states that the TRW method needs a large amount of response crowds to decide properly. Although it is true that the *response crowd activity check* algorithm generates many response crowds, it is also true that the *response crowd density check* algorithm does not. Unfortunately, this last algorithm was not deeply analyzed in the proposal, and therefore, it may not be suitable for the TRW method. On the other hand, the main detection method was described to be a protocol-independent approach; however, it depends on four conditions: first, on detecting the HTTP; second, on detecting the IRC protocol; third, on detecting the DNS mail exchanger records for SPAM sending; and fourth, on detecting the SMTP.

4.2. Appendix B of BotSniffer

Appendix B of [27] proposes to identify HTTP C&C channels by detecting a repeating and regular visiting pattern from one single bot. The proposal uses signal encoding and autocorrelation approaches. HTTP requests are captured, and a time series of two tuples (timestamp and amount of bytes transferred) is generated. The sign of the *bytes transferred* value represents the packet direction. This time series is analyzed using autocorrelation techniques to find out its time-spatial properties, such as the presence of a periodic signal. Autocorrelation is computed for each lag, and consequently, an autocorrelation series is obtained. Some experiments were conducted to prove this method, which resulted in several TPs and some FPs. If the proposal would have considered the differentiation between manual (or automated) attacks and botnet attacks, then a more realistic scenario could have been analyzed.

The analysis of this proposal shows that the dataset used for verification purposes is only composed of custom-compiled botnets in a virtual internal network. This dataset might lack real botnets in the wild. The normal dataset used is the same as in the original BotSniffer proposal 4.1. In this dataset, the BotSniffer program itself is used to test the normality of the BotSniffer validation dataset. It is not correct to use the same method for validation purposes.

Different datasets are used to compute the FPs and the detection performance. However, in [41], it is stated that "Also, it is important to make sure that the data used for estimating a model and the data used later for testing and applying a model come from the same, unknown, sampling distribution." Perhaps, different datasets are used because of the difficulty to obtain more trusted data. Also, this might be the cause why the proposal does not compute FNs and TNs. Moreover, it may be why there are no comparisons with other proposals, and therefore, the results could not be verified.

Fortunately, this proposal is capable of detecting botnets by capturing packets in one host.

4.3. BotMiner

The BotMiner detection framework [28] has three different phases of analysis. The first phase groups together hosts with similar activity patterns. The second phase groups together hosts that achieve similar attacking patterns, and the third phase groups together hosts on the basis of similarities from the other two phases. The first phase uses flow information such as the Internet Protocol (IP) addresses, ports and the *network profile* of the flow. It computes statistical measures such as *flows per hour*, *packets per flow*, *average bytes per packets* and *average bytes per second*. A two-step *X*-means clustering method is used to create clusters of hosts that behaved similarly in the network. The second phase uses the Snort IDS to extract attacks from each host and to group hosts by attack type. Among the malicious actions detected are SPAM, exploit attempts, port scans and binary download. The third phase computes a score for each attacking host on the basis of previous similarities and uses it to calculate a similarity function between hosts. Finally, a dendrogram is created using this function to find out the best cluster separation. The most important statical characteristics used were the mail exchanger DNS queries to detect SPAM and the 25 TCP ports for SMTP.

The proposal uses both virtualized and real botnet captures. Unfortunately, there is no information about how the botnet dataset was verified. The normal captures were verified using the BotHunter and BotSniffer software. In the preprocessing stage, some well-known Web sites are whitelisted. However, the implications of the filter are not exposed. It also forces us to maintain a list of well-known hosts, which could be error prone and time-consuming. The results seem encouraging. However, they could not be reproduced because the dataset was not made public. Comparisons with other papers were not carried out. Unlike other proposals, this work uses one novel idea to differentiate between botnets and manual attacks: botnets act maliciously and always communicate in a similar way, but the manual attacks only act maliciously.

4.4. BotHunter

The BotHunter framework [29] recognizes the infection and coordination dialogue of botnets by matching a state-based infection sequence model. It captures packets in the network egress position using a modified version of the Snort IDS with two proprietary detection plug-ins. The proposal looks for evidence of botnet life cycle phases to drive a bot dialogue correlation analysis. IDS warnings are tracked over a temporal window and contribute to the infection score of each host. The host is labeled as a bot when certain thresholds are overcome. This proposal associates inbound scans and intrusion alarms with outbound communication patterns.

An infection is reported when one of two conditions is satisfied: first, when an evidence of local host infection is found and evidence of outward bot coordination or attack propagation is found and, second, when at least two distinct signs of outward bot coordination or attack propagation are found.

Five experiments were conducted. The first experiment was carried out on a virtual test bed and used one VMWare Linux, one IRC server and two infected Windows instances. The traffic of this experiment was injected. The second experiment was performed on a honeynet and used nine Windows XP, 14 Windows 2000 and three Linux from a Drone Manager. The third experiment was carried out on a college network and was used for normal packet capture. The fourth experiment used a university campus for normal packet capture, and the fifth experiment was performed in a production network with 130 IP addresses for 10 days. Only one experiment was carried out with normal and botnet data at the same time, probably because of the difficulty to mix two types of datasets together.

Some normal and botnet captures were not completely verified, probably because of the difficulty of performing such a verification. However, the verification of the dataset is the best way to assess whether a variable represents what it is intended to measure [42].

The performance metrics reported in the proposal are incomplete. This might be caused by the limited labels included in each dataset. In the first experiment, a TPR of 100% was reported, but no FPs or TNs were computed. In the second experiment, an FP error of 95.1% and an FN error of 4.9% were reported. In the third experiment, a TPR of 100% was reported. However, no FPs, FNs or TNs were reported. In the fourth experiment, an error of 0.81 FPs per day was reported. However, no FNs or TPs were reported. In the fifth experiment, one FP per 10 days was reported. However, no FNs or TNs were reported. Finally, the proposal did not compare the results with other papers.

The BotHunter proposal is based on a statical IDS; thus, some of the detections are static. For example, the IRC protocol is identified by means of the 6667 port, and some IP addresses of known botnet servers are embedded into the Snort configuration file. Among the bot statical characteristics detected, the sequence of bytes in the binary download and the Snort statical fingerprints are used by the proposal.

The proposal did not publish the dataset used. This might be due to privacy issues or nondisclosure agreements. There are also no details about the Statistical Scan Anomaly Detection Engine and Statistical Payload Anomaly Detection Engine port scanning detection algorithms. Consequently, the reproduction of the paper might be difficult to accomplish.

The proposal does not seem to be designed to detect bots using only the traffic from one host. Moreover, it is not designed to differentiate between botnets and manual (or automatic) attacks. However, as the model is based on the life cycle of botnets, it is very probable that it could work fine for this situation. The method has two major advances. First, it seems capable of analyzing, detecting and reporting botnets in real time. Second, it is the only proposal that was published as a product.

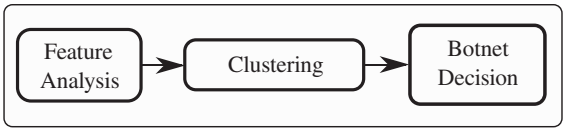

Figure 5. *N*-gram architecture.

4.5. N-gram

The *N*-gram work [30] proposes an unsupervised, classification-based and IRC-based bot traffic detection method. Figure 5 shows the architecture of the proposal. It has two phases: classification of application traffic and detection of bots. The first phase has several stages. The first stage classifies traffic into known applications using signature-based payloads and known ports (signatures were generated previously). The second stage classifies unknown traffic from the previous step using a decision tree based on temporal-frequent characteristics. This tree differentiates known application communities (the proposal have the temporal-frequent characteristics of known flows). The third stage uses an anomaly-based approach to cluster the traffic. To differentiate malicious traffic from normal traffic, the proposal first creates profiles for known applications; these profiles use a one-gram technique that builds a 256-position vector holding the number of times that every byte appeared in the traffic payload.

The second phase has several stages, and according to the paper, it starts with the feature analysis stage. However, no information is given about this stage, seriously limiting the analysis of the proposal. The real first stage, then, clusters the unknown traffic (which still remains unknown after the first phase) using the *K*-means, *unmerged X-means* and *merged X-means* algorithms. The second stage assigns the label *botnet*, in the merged *X*-means algorithm, to the cluster with the lower standard deviation. There is no information about how the labeling is performed in the other methods. The proposal states that the unmerged *X*-means is the normal *X*-means and that the merged *X*-means "tries to find the botnet cluster as cluster with lowest standard deviation and then gather as many instances that represent botnet IRC as possible from remaining clusters."

The proposal assumes that bots reply to commands quickly, that bots communicate synchronously and that botnet IRC content is less diverse than normal IRC. The analysis shows that the training dataset might not be large enough to be considered representative of the problem. It has only one real botnet, one simulated botnet and two normal captures. The paper does not describe any verification of the normal captures.

The proposal is difficult to reproduce, as the labeling process is not described, the performance metrics analysis is incomplete and the dataset was not made public. There is also no comparison between the results presented and other papers. The analysis shows that the method is not completely unsupervised, because the centroids of the *k*-means algorithm were manually selected. This introduces a bias and makes the method unreproducible. The method presents a novel and simple idea to detect botnet clusters automatically: to search for the cluster with the lowest standard deviation.

4.6. Unclean

The Unclean proposal [32] predicts future hostile activity from past network activity. The term *spatial uncleanliness* is defined as the propensity of bots to be clustered in unclean networks, and the term *temporal uncleanliness* is the propensity of networks to remain unclean for extended periods. The proposal is divided into two phases.

In the first phase, a dataset is used to evaluate both unclean properties. Although there is no detailed information about the capture procedure, the dataset has two types of information: the first type is external reports of phishing, port scans and SPAM activity, and the second type is some internal network captures. External reports from third parties are used as *ground truth* for the experiments. The internal capture was performed by observing a public network. The spatial uncleanliness property is evaluated by comparing the population of IP addresses in an unclean report against the normal expected random population of the Internet. If the hypothesis is true, then the IP addresses of the bots should be more densely packed than the randomly selected addresses. The temporal uncleanliness property is evaluated by testing how an old report of unclean addresses predicts compromised IP addresses in the future. It is expected that unclean old reports are better predictors of future IP addresses than randomly chosen IP addresses.

In the second phase, the paper analyzes the performance impact of blocking these future bot IP addresses to differentiate between innocent, unknown and hostile IP addresses. An FP is considered each time an *innocent* IP address is blocked, and a TP is defined each time an unclean IP address is blocked. Unknown IP addresses are not used, probably adding a bias to the results. Results supported the hypothesis of spatial and temporal uncleanliness in the experiments.

An analysis of the validation experiments for the first phase suggests that the results are incomplete. A TPR of 90% was reported, but no FP, TN, FN, FPR or FNR were reported. Also, results are not compared with those of any other paper. Two assumptions were made: first, that bots are always used to attack and, second, that if a 5-month-old unclean report can predict current unclean activity, then a recent report should be more effective. The analysis of this proposal shows that the verification of the external reports was not described and that the dataset was not published. Also, it was not in the design of the proposal to differentiate between botnets and other types of attacks.

4.7. Tight

The Tight proposal [34] aggregates traffic to identify hosts that are likely part of a botnet. The hypothesis is that some

evidence of botnet IRC C&C communication activity can be found by monitoring IRC traffic at a core location. The training dataset is composed of 600 GB of wireless normal traffic from the CRAWDAD archive and 74 flows of a compiled botnet from an internal test bed. The proposed method has five steps. First, flows are mixed in a common dataset (unfortunately, there is no information about how the mixture was performed). Second, a whitelist and a blacklist are used to filter the flows. Third, flows are classified according to their chat-like characteristics. Fourth, correlations are used to find similar C&C flows. Fifth, a topological analysis of the flows is carried out to identify common connection endpoints and to manually determine which is the traffic between the controller and the *rendezvous point*.

Several assumptions are made in the paper: first, that the IRC-based botnet command messages sent by the botmaster are brief and text based; second, that two different flows of the same application behave similarly; and, third, that botnet C&C channels do not sustain bulk data transfers. The analysis shows that some filters overfits the data, such as deleting the high-bit-rate flows. However, these filters were not verified, and they seem to be created specifically to filter out the already-known botnet traffic.

There is an issue regarding the dataset. When the botnet packets were obtained, the secure shell [43] traffic generated from the test bed setup was also captured. This biases the capture. Almost every dataset has a bias, but it is important to enumerate and consider them [42]. Furthermore, the normal captures were not verified, and the dataset was not published.

Another bias is present in the results. Some results were obtained by applying the classifiers to one dataset; other results were obtained by applying the classifiers to another different dataset. It is commonly accepted [42] that there should be at least three types of datasets: training, testing and validation. An evaluation methodology should also be used [44].

The design of the proposal is considered to detect in real time, but unfortunately, no experiments were carried out to support this idea. A major highlight of the paper is that it focuses on novel behavioral features: a bot communicates with a C&C server, the server is controlled by a botmaster and the bots synchronize their actions.

4.8. Stability

In Stability [10], P2P botnets are detected using flow aggregation and stability measurements. The hypothesis is that the proposal "can further distinguish the structured-P2P-based bot from normal clients by detecting the stability of I-flow." The dataset was created by capturing background traffic from a university campus and later injecting real botnet traffic into it. The botnet traffic was captured in a honeynet, and thus, it is verified as malicious by the definition of a honeypot. Figure 6 shows the architecture of the proposal.

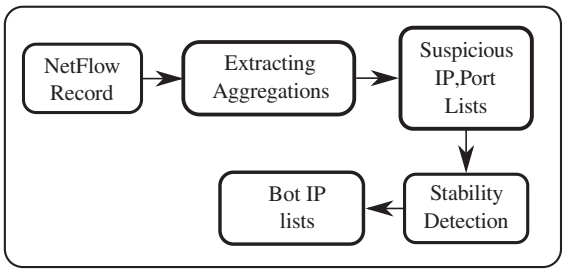

Figure 6. Stability architecture. IP, Internet Protocol.

This proposal is divided into two phases. In the first phase, the flow features are extracted. In the second phase, these features are used to compute the stability of the flows. Before the first phase, the dataset is divided into 1-h time windows.

The first phase is composed of several steps. In the first step, protocol flows are aggregated on the basis of the double two-tuple IP addresses and ports. In the second step, for each aggregated flow, two novel values are computed: *flows per aggregation flow* (fpf) and *average bytes per aggregation flow* (abf). In the third step, an entropy value that uses the fpf as a random variable is computed. A high fpf and an abf lower than 300 bytes were used to detect bot flows.

The aim of the second phase is to compute the flow stability with two sliding windows: one serves as a baseline and the other as a detection window. In the first step, the distance between the abf of both windows is compared against a predefined threshold. Every time the distance overcomes the threshold, a *flow change* is counted. In the second step, windows are moved forward, and the *stability index* of the flow is computed. If the *stability index* is above a threshold of 90%, then a botnet is detected.

The proposal assumes that common DNS ports are not going to be used for C&C channels and that the common P2P ports are enough to detect that protocol.

The analysis of the proposal shows that the preprocessing stage adds a bias to the results. The first bias is added when DNS traffic is filtered out after discovering that it has a *stability index* similar to that of P2P botnets. The second bias is added when port 28000 is filtered after discovering that it is being misdetected by the algorithm.

Reproduction of the method is not possible because of two reasons. First, the paper uses several unspecified ad hoc experimental values. Second, the dataset was not made public. Unfortunately, the normal traffic was not validated.

4.9. Incremental

In Incremental [36], the botnet activity is monitored on the basis of similarities between the traffic *feature streams* of several hosts.

The proposal is divided into five steps. The first step captures both normal and botnet traffic from an internal network. P2P botnet traffic is obtained from a third party.

The captures in this step are simulated, as the internal network did not have access to the Internet.

The second step reduces the volume of normal data by applying some filters. Non-TCPs are filtered out, several whitelists and blacklists of common Web sites are applied and packets with more than 300 bytes are deleted. The third step creates the *feature streams*. Traffic is divided into 5-min time windows, and two features are computed: *average bytes per packets* and packet amount.

The fourth step computes a distance measure between streams using an incremental discrete Fourier transformation technique. The novel characteristic of this technique is that it does not need to recalculate all the coefficients every time a new value arrives.

In the fifth step, the Snort IDS is used to manually verify the activity of the suspect hosts (hosts with high similarity streams). In this step, it is also decided whether this host is part of a botnet or not. Unfortunately, no information about this step is given.

The proposal assumes that simulated botnet traffic behaves like real botnet traffic and also that the traffic from their network is normal. The proposal was not designed to differentiate between botnet and other attacks.

The analysis shows that the results are unclear. No FNs were computed, and a perfect detection accuracy is reported. The paper also states that the "false positive rate is still relative low because we *will* analyze their activities to confirm the final result." Unfortunately, the confirmation is not in the proposal. The results are not compared with those of other papers.

The proposal uses several filters in the preprocessing step. However, it does not seem that the bias added by these filters would have been evaluated.

Different distance measures are used in the proposal. However, there is no information about the distances called *protocol distance measure* and *start time distance measures*. Consequently, it is difficult to reproduce the method.

During the validation experiments, normal traffic and botnet traffic were mixed. Unfortunately, the mixture process is not explained.

This proposal has two highlights. First, it seems capable of monitoring botnets in real time using the time series-like *feature streams*. Second, the *feature stream* implementation, although not novel, is an important improvement in the area.

4.10. Models

The Models [31] proposal identifies single infected machines using previously generated detection models. These models are composed of two parts. The first part extracts the strings received by the bot on the network to find the commands. The second part tries to find the responses (attacks) to the command sent. Models are stored for later use in real networks. The novel idea is to search for command signatures (string tokens) within the network traffic and then search for the responses to these commands. This technique correctly separates botnets from other attacks.

The proposal is separated into five phases.

The first phase consists of several steps. In the first step, traffic from 50 bots is captured. In the second step, bot responses are located within the traffic. This is carried out by separating the traffic into 50-s time windows and by computing eight features for each time window. The features are the number of packets, cumulative size of packets in bytes, number of unique IP addresses contacted, number of unique ports contacted, number of non-ASCII bytes in the payload, number of UDP packets, number of HTTP packets and number of SMTP packets. In the third step, the features are normalized. In the fourth step, the change point detection algorithm uses these features to detect where the traffic changes. This algorithm uses the cumulative sum method to detect the changes.

Once the change point is detected, in the fourth step, data are extracted from the traffic to create the profiles. Bot command profiles are composed of string snippets extracted from the 90-s time windows where the change point detection took place: the number of UDP packets, number of HTTP packets and number of SMTP packets and IP addresses. The final response profile is the average of these vectors over the total time of the bot response.

In the second phase, models for commands and responses are created. A model has two parts: the strings for command detection and a network-level description for response detection.

In the first step of the command model creation, snippets that contain the same tokens are grouped together using the longest common subsequence algorithm. In the second step, response traffic payloads are clustered together using hierarchical clustering. In the third step, snippets from the first-step clusters are grouped according to the second-step traffic clusters (snippets likely to generate the same response). In the fourth step, the precision of the method is improved by deleting the tokens found in previously generated and unverified normal captures.

Still, in the second phase, in the first step of the response model creation, the element-wise average of individual profiles is computed on each cluster. In the second step, some FPs are filtered out using a threshold. If none of the following thresholds are overcome in 50 s, then the response is discarded: 1000 UDP packets, 100 HTTP packets, 10 SMTP packets and 20 different IP addresses.

Once the models are created, the proposal implements them. In the third phase, previous models are mapped into the Bro IDS. Bro is configured to work in two stages. In the first stage, if a Bro command signature is matched, then Bro changes to stage 2. In stage 2, a bot detection is achieved if any of the following traffic thresholds are overloaded: the number of UDP packets, number of HTTP packets, number of SMTP packets or number of unique IP addresses.

In the fourth phase, an evaluation is performed to obtain the performance metrics. A total of 416 binary bots (18 bot families) were obtained from the Anubis service along with

30 nondescribed storm captures. The evaluation process is performed in several steps. In the first step, detection models are created using a training set composed of 25% of the total traces. In the second step, results are obtained using a testing set composed of 75% of the total traces.

In the fifth phase, two real-world experiments are carried out. First, the already configured Bro IDS is used in a university campus network, and later, the same Bro IDS is used in an organizational network.

The HTTP is detected by filtering the TCP port 80, and the SMTP is detected by filtering the TCP port 25.

The analysis shows that the bot families are separated by hand while creating the models, probably because, once created, the models are useful for a long time. On the other hand, the FPs are manually verified. These issues make the verification of the method difficult.

The proposal seems to use a good dataset, but the information included is not enough to verify its diversity. Also, the third-party Anubis dataset is not verified. Moreover, there is no information about the preprocessing of the dataset.

A highlight of this proposal is that the results are compared with those of the BotHunter proposal using the Models proposal [31] dataset.

4.11. FluXOR

The FluXOR proposal [33] detects and monitors fast-flux-enabled domains. Given a fully qualified domain name, its goal is to verify whether it concealed a fast-flux service network and, in such a case, to identify all the agents that are part of the network. Although not strictly a botnet detection technique, it is one of the few proposals that could detect fast-flux botnet domains. Suspicious domains are detected when traces of fast-flux characteristics are found in the DNS and WHOIS information of the domain. A fast-flux behavior is detected with nine features that describe the properties of the domain, such as the degree of availability and the heterogeneity of the hosts in the DNS A records. The experiment setup includes a collector module, which harvests domains from different sources; a monitor module, which monitors domains; and a detector module, which feeds the classifier algorithm from the Waikato Environment for Knowledge Analysis suite [44]. This proposal does not have a detailed performance metrics analysis, so it is not possible to evaluate how well it performed. A 0% FP was reported; however, no other results were given.

The analysis shows that the malware domains are extracted from SPAM mails and normal domains are extracted from Web history. However, there is no description about the verification of the domains. Unfortunately, the dataset was not made public, probably because of the privacy and security issues related to the domain names. On the other hand, it may be possible to reproduce the algorithm, as the Waikato Environment for Knowledge Analysis framework is publicly available [44].

The design of the proposal does not include real-time detection capabilities. However, under some conditions, it could be possible to detect fast-flux botnets within a very short time.

4.12. Tamed

In the Tamed proposal [35], it is hypothesized that "even stealthy, previously unseen malware is likely to exhibit communication that is detectable." The proposal aims to "identify infected internal hosts by finding communication *aggregates*, which consist of flows that share common network characteristics." This is the first proposal that does not try to detect botnets themselves but presents a coherent and useful set of features in which future work can rely their detection methods on. The work is divided into two phases: the definition of their aggregate feature function and the evaluation of experiments.

In the first phase, three different aggregation functions are defined: destination aggregation function, payload aggregation function and common platform function. The destination aggregate function finds suspicious destination subnetworks for which there are a large number of interactions with the internal network (and the IP addresses involved). This is performed by comparing a baseline past network record with the current network record. The function is composed of two steps. First, the past normal traffic is used to remove the periodic communications. Second, the principal component analysis algorithm is used to find the most important components, and then, a clustering technique is used to find the hosts that connect to the same combinations of destinations.

The payload aggregate function uses the *edit distance with substring moves* to output a normalized ratio. This ratio indicates how many distances are below a given threshold between two payloads (and thus need training). The proposal also contributes an algorithm for approximating the fraction of relevant record pairs that satisfy this distance (using a variant of the *k*-nearest neighbor).

The *common platform function* uses two heuristics for fingerprinting host operating systems passively: time-to-live fields and communication characteristics (such as Windows computers connecting to the Microsoft Time Server). This last function returns the largest fraction of internal hosts that can be identified with the same operating system.

The proposal assumed that the C&C server can be in a different network and that bots will not use the Tor* network.

The analysis shows that the normality of the traffic captured at the university is not verified but granted as normal.

In one of the experiments, the traffic from one botnet was merged with traffic from the university. The IP addresses of the bots were substituted with the IP addresses of hosts in the university. The intention was to have IP

* https://www.torproject.org/

addresses with normal behavior patterns and botnet behavior patterns at the same time. However, no analysis was performed about how this change could affect the common platform aggregate function. This issue might be the reason of some FNs during one experiment. Unfortunately, the dataset was not made public, so it is difficult to reproduce the method.

The proposal stated that a 100% TPR was achieved with the *exception* of one experiment. In this isolated exceptional experiment, an 87.5% TPR was reported. The results of all the experiments should be computed together to calculate the performance metrics. There cannot be any experiment exceptions while computing the results.

The proposal was not designed to detect bots using only the traffic from one host. Also, it was not designed to differentiate between botnets and other types of attacks.

4.13. Markov

Port scan activities of botnets are detected in [37]. The scanning phases are represented with text symbols, and a hidden Markov model (HMM) is used for training and detection. The work is divided into four phases. In the first phase, a system is used to capture information from the network and generate network logs. Unfortunately, this system is not described. Data were captured for 1 month in 11 C-class university networks. Incidentally, an unknown amount of hosts was found to be infected.

In the second phase, the amount of TCP packets present in certain time windows with SYN and ACK bits is computed. Unfortunately, no information about the time windows is given. The amount of packets is plotted, and each distinct shape of traffic is assigned a different text symbol. This phase supposedly ends up with a string of letters (one gram) that represent each port scan (observations).

In the third phase, the training phase, the Baum–Welch algorithm is used for 6 h to find a Markov model that maximizes the probability of each sequence of observations.

In the fourth phase, strings are extracted from the rest of the traffic (supposedly of unknown type). However, there is not much information about this. Then, the Viterbi algorithm is used to find the most probable sequence of states in an HMM from each group of observed states, that is, to find the model that better explains the observations. The result of the Viterbi algorithms is used as a type of score. We suppose that they selected the sequence of stored port scan strings that maximizes the score. At some point, a comparison is made between a botnet pattern and an unknown pattern using a similarity function. Finally, the early detection of port scans is defined by these similarity values. However, no information about any threshold or decision boundary is given.

The proposal assumes that the botnets only scan TCP ports.

The analysis shows that the dataset used is not verified. There is no clear description of the dataset. Moreover, it is stated that the dataset contains both unknown data and botnet data, but it is not labeled. Also, results were not compared with those of other papers.

Unfortunately, it is difficult to completely understand the method. There is no information about how the port scan detection is performed. Also, the proposal states that the method behaves fine with Monte Carlo simulations, but there is no information about this. The experimental setup is not described. Furthermore, a bigger detection system is described, with botnet policy management and classifications steps, but there is no information or explanation about it. The information given about the automatic detection of botnets does not have enough details.

4.14. Synchronize

Network synchronization and clustering techniques are used to detect bots in [13]. There are five phases. In the first phase, data are captured. Three different test beds were used to capture five datasets. Three of the datasets correspond to botnet captures and two to nonbotnet captures. The nonbotnet captures include a manual port scan and normal traffic. Verification of the captures is carried out for both botnet and nonbotnet datasets. The botnet binaries are verified with the VirusTotal Web service [45] and the EUREKA! automated Malware Binary Analysis Service [46]. The botnet traffic is manually verified by experts. The nonbotnet traffic is verified by a fresh installation of the operating system, antivirus programs and the Snort IDS.

In the second phase, the TCP flows are extracted using the tcptrace tool.

In the third phase, flows are divided into 1-s time windows, and three features are used to aggregate them: the amount of unique source IP addresses in a time window, the amount of unique destination IP addresses in a time window and the amount of unique destination ports in a time window. Each instance represents a time window with the aggregated features.

In the fourth phase, the expectation–maximization algorithm is used to cluster the instances. A dataset with both types of traffic was obtained by merging botnet and nonbotnet data. Four experiments were conducted: first, to separate between botnets and port scanning activities; second, to separate between botnet and nonbotnet traffic from one host; third, to separate between botnet and nonbotnet traffic in an unbalanced dataset; and fourth, to separate between a bot and a network of nonbotnet hosts.

The fifth phase analyzes the results. The proposal considers the usual definition of FPs and FNs. However, it also includes a novel definition of error metrics based on the administrator perspective, that is, to compute an FP only when a nonbotnet IP address is detected as a botnet at least once during a period and also to compute an FN when a botnet IP address is always detected as a nonbotnet in a time window.

The proposal assumes that botnets tend to generate new flows almost constantly and that botnets' most typical characteristics are maliciousness, being remotely managed and

DOI: 10.1002/sec

synchronization and also that botnets only use the TCP protocol.

The analysis shows that the dataset included nonbotnet data and manual attacks. However, it is rather small, and it does not cover a large proportion of botnet behaviors. Only five captures were made, and none of them was made in a large network.

Unfortunately, there is no justification of the 1-min time window used. The results were not compared with those of other papers.

One of the highlights is that all the tools and datasets used were made public on the Internet. These datasets might allow other researchers to verify the method. Another highlight is the differentiation between botnet and port scanning traffic.

Unfortunately, no automatic botnet detection method was established, meaning that the final decision is left to experts.

The next section discusses and analyzes the main issues found in the area.

5. DISCUSSIONS ABOUT THE NETWORK-BASED BOTNET DETECTION AREA

This section discusses the issues found in the network-based botnet detection area. The analyses of previous surveys and botnet detection papers showed which problems are solved, which algorithms are used and which questions remain to be answered. The area has obtained good results. However, it still has many issues to tackle. The following paragraphs discuss some general issues, and the next subsections discuss two groups of common problems.

Reproducibility

Unfortunately, most methods cannot be reproduced because of two factors. First, the datasets were not published. The following papers could not publish their datasets: [10,27–37]. Second, there was not enough information about the methods. Without the step-by-step description of the algorithms, the threshold used and the whitelists, reproduction might be unachievable. The following proposals did not publish all the necessary information to reproduce their results: [10,27–30,32,34–37].

Amount of hosts

The amount of hosts needed in the network to detect botnets should be considered. It normally depends on the design of the proposal. Most proposals need a large amount of hosts and packets to obtain meaningful results. This limitation might play a negative role in the future adoption of the method. Such amount of hosts can only be obtained by capturing packets on large networks. If the organization that uses the method has such a network, then access to the network can be also made difficult by the nondisclosure agreements that must be fulfilled. On the other hand, if the organization does not have such a network, then it can be extremely difficult to have access to the data. Therefore, some techniques might be only valuable for Internet service providers, large organizations or universities. Currently, there are only two proposals that detect bots using the traffic of one host alone: Appendix B of [27] and [13].

Features

An important phase of a botnet detection research is the preprocess of data before the feature extraction [42]. Some proposals tend to overfilter the data and to produce algorithms that work better with a specific dataset. These decisions should not add a bias to the experiments. A *data dredging bias*[†] can occur when researchers either do not form a hypothesis in advance or narrow the data used to reduce the probability of the sample refuting a specific hypothesis. An example is the whitelist of known Web sites under the assumption that there is a low probability that they will be part of a botnet in the future [27,28,36]. This decision might be taken because most of the traffic on the datasets was directed to popular Web sites. However, the assumption that these sites can be safely whitelisted should be verified, as these sites were successfully attacked in the past, such as Google and Adobe.[‡] Furthermore, another source of bias was the assumptions made to filter out the packets that did not fit the method. For example, some proposals filter out packets of the UDP or Internet Control Message Protocol without further explanations [13,27,36], narrowing the sample to match their hypothesis. In general, this type of decision is not recommended, as some bots use the UDP exclusively [6].

Experiments

Performance metrics are important in every experiment and therefore should be correctly computed [44]. However, some experiments were incorrectly created. For example, some proposals compute the TPR and FNR metrics using only a botnet capture [29,31,32]. Other proposals compute the TNR and FPR using only a normal capture [27,31,32]. These types of metrics do not help us understand the results, in the former case, because always predicting a positive outcome generates a 100% TPR and a 0% FNR and, in the latter case, because always predicting a negative outcome generates a 100% TNR and a 0% FPR. More interesting results would be obtained if both normal and botnet captures were used in the experiments.

Metrics

Two important issues were found regarding the accuracy-based performance metrics. The first issue, shown in Table VIII, is that most proposals did not compute the entire set of metrics. This means that

[†] https://secure.wikimedia.org/wikipedia/en/wiki/Data-snooping_bias

[‡] http://www.wired.com/threatlevel/2010/01/operation-aurora/

most proposals cannot be compared. It is difficult to understand how well the methods performed.

The second issue is that some proposals did not clearly describe the design of its experiments. For example, in some papers, it was impossible to tell if they detected 90% of botnet IP addresses, 90% of botnet flows or 90% of botnet actions. There is a huge difference between detecting 90% of botnet IP addresses within a 5-min time window and detecting 90% of botnet flows within a 5-min time window.

In the next subsections, the discussion is extended to more specific topics.

5.1. Issues related to the datasets

The importance of using a good dataset, as stated in [34], is sometimes underestimated. A dataset with few botnets could lead to very limited methods. The following paragraphs describe the issues found.

First, proposals tend to overlook the verification of the datasets. A dataset in [37] was used for training and validation without verification. Furthermore, a proposal jeopardized the dataset by including in their capture the secure shell traffic used to set up the virtual environment [34].

Second, the amount of botnets captured is not evaluated. Some proposals [30,34] captured a very small amount of botnets. If the proposal tried to detect HTTP botnets, then a large amount of different HTTP botnets should be captured for the sample to be representative of the population.

Third, the amount of time spent on the capture is not evaluated. The complete behavior of botnets cannot be captured in a few hours. However, most proposals tend to capture botnets during a short time [10,13,27,28,30,33–36]. Depending on the design of the proposal, this could lead to an incomplete view of the botnet behavior. It is recommended to capture traffic during long periods.

Fourth, labels are not verified. Some proposals captured traffic from an internal network and labeled it as *normal* without verification [10,27,29–31,34–37]. However, it is common to find infected computers inside real networks. Without verification, this type of traffic should be considered *background* instead. Botnet captures are usually labeled as *malicious* only because a malware binary was used, but no verification is carried out. There is no definition of what should be considered as botnet traffic. Should the traffic generated by a malware process or the one generated by an infected computer be considered as a botnet? Because an infected computer also generates normal traffic, the answer is not straightforward.

Fifth, botnets are not distinguished from other types of traffic besides normal. However, it could be helpful to distinguish botnets from manual and automated attacks. If a proposal uses common attacks to detect botnets, care should be taken not to have FPs. Only one proposal was designed to use the botnet attack responses and signatures to better differentiate botnets from attacks [31]. Another consideration should be made with port scanning activities. Should it be considered part of a botnet or not? Port scanning is commonly considered an attack, but it is also a well-known administrative tool in most networks. Only one proposal differentiated between port scanning and botnets [13].

Sixth, there is a need for a public botnet dataset. This is perhaps the most important conclusion of our survey. It had been stated that big data should be available for research and validation [47]. Without a public dataset, it is not possible to compare methods. Currently, proposals are forced to create their own datasets. The following reasons contribute to the lack of a common and public dataset. First, proposals use different botnet features for their analysis. Therefore, each one needs to create an appropriate dataset. Second and most important, traffic captures have much private information that is dangerous to publish. Researchers usually had to sign nondisclosure agreements with the network owner before having access to the data. Furthermore, these captures usually include malware information that should be kept private to avoid spreading the malware. The importance of a public dataset is reinforced by these limitations. A public and complete dataset could be a very valuable contribution to the area. To be useful, it should be correctly designed, it should include every type of botnet, it should be regularly updated and, most importantly, it should be correctly labeled. The creation of a common dataset is perhaps one of the most important issues that the area has yet to achieve.

The seventh issue is on the representation of network traffic. It is common to need huge amounts of disk storage early in the capture process. Researchers have to solve the problem of how to store useful data long enough. The common solution is to use network flow description methods. The idea is to reduce the volume of unused data and retain as much useful data as possible. Two standards are a common choice: IP Flow Information Export (IPFIX) and Argus network flows. The IPFIX standard was defined by the Internet Engineers Task Force, and it is based on the original NetFlow standard. Argus uses its own network flow definition for the same task. One difference between Argus and IPFIX is that Argus flows are bidirectional, whereas IPFIX flows are unidirectional. This subtle difference can have a critical impact on results and should be evaluated.

Eighth, the issue is on how to obtain good botnet binaries. The most common options are as follows:

(1) Use your own honeypots to obtain the binaries.
(2) Obtain binaries from third parties, such as antivirus or other institution honeypots.
(3) Obtain and compile the source code from the Internet.

The first choice is the best, but it can take a long time to obtain useful binaries. The second option is the most common. The third choice is good if the test bed has a connection to the Internet and no services are simulated.

DOI: 10.1002/sec

5.2. Issues related to the experiments

To achieve meaningful results, it is important to carefully design the experiments. Some papers correctly proposed to mix normal captures with botnet captures to obtain a better dataset. However, the mixture processes are not often explained [27,29,34,36]. To mix a dataset means to decide the balance of the mixture. Different balances produce different results and would have different benefits. In [39], it was advised that an unbalanced dataset can bias the results. Nevertheless, it is common to have more packets of normal activity than botnet activity. The mixture of network packets is not easy. There are two main options: to concatenate the traffic or to merge it. To concatenate is easier but more unreal. To merge is better but more difficult to obtain.

An uncommon solution to the merging issue is to change the IP address of one *botnet* host for the IP address of one *normal* host [35]. The solution tries to have an IP address with both types of behavior. However, this change could have unpredicted implications in the performance metrics. In fact, in [35], a problem is detected on the passive operating system detector results.

5.3. Issues related to the detection methods

The botnet behavior continuously changes, and detection methods should change also. Supervised methods could detect known botnets, but new botnets are still difficult to detect. Methods should adapt to the changing behavior of botnets and should be more flexible. There is a need to increase the detection of unknown botnets.

Several proposals used unsupervised methods [13,27,28,30,31,35]. However, none of them verified the methods with unknown botnets.

6. CONCLUSIONS

The most relevant papers and surveys in the network-based botnet detection area have been reviewed. The survey dimensions, the desired properties, the topology map and the papers were compared to understand the issues in the area.

These findings are important for future works. It is easier to improve our research with the knowledge of what other proposals have performed.

Our research has encountered two limitations. First, some papers did not publish the details of their algorithms. Second, there was no access to most datasets. These problems highlight the difficulty to compare papers.

Our findings suggest that the most relevant issues on previous surveys are the undefined terminology, the focus on different aspects of botnets, the narrow analysis of the papers and the small amount of papers covered.

On the other hand, the problems in the area that need attention are

- the use of unverified captures,
- the lack of public datasets,
- the small amount of botnets in the datasets,
- the wrong mix-up of packets to create datasets,
- the inaccurate outcomes of experiments,
- the whitelist of common Web sites,
- the lack of comparison with other proposals,
- the inaccurate report of performance metrics and
- the common overfit of preprocessing methods.

Among the reasons for these problems are the fast evolution of botnets and the difficulties in obtaining real and working botnet traffic.

In light of these conclusions and the available data, a final word can be said about botnet detection. The intention is not to find the best methods but to point to the probably most useful detection approaches. The complexity of botnet network behavior seems to be better detected with time-based dynamic approaches, that is, approaches that adapt to changes in behavior over time. To detect unseen botnets, it seems to be a good path to use the most general behavioral features, although some particular behaviors could be missed. It also seems to be a good approach to use hybrid detection methods, where a meta-learner can weigh and decide over the results of different behavior detection algorithms. The results obtained by these types of approaches on IDSs are encouraging [48]. Finally, if several algorithms are used in parallel along with a huge amount of data, then it may be useful to implement big-data solutions [49].

Further studies should mainly target dataset improvement and the comparison of methods with those in other papers. The comparison is perhaps the best way to improve the efforts in the area. We hope that our research will serve as a basis for future studies. We are currently in the process of creating a public and accurate dataset for the comparison of methods. We will also cover new detection methods using botnet characteristics.

REFERENCES

1. McCarty B. Botnets: big and bigger. *IEEE Security & Privacy* 2003; **1**(4): 87–90, DOI:10.1109/MSECP.2003.1219079. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1219079 [Accessed on 9 May 2013].
2. Microsoft Security Intelligence. Microsoft Security Intelligence report. *Technical Report*, Microsoft, December 2008. Available from: http://www.microsoft.com/security/sir/archive/default.aspx [Accessed on 9 May 2013].
3. Wilson C. Botnets, cybercrime, and cyberterrorism: vulnerabilities and policy issues for congress. *Congressional Research Service Reports (CRS) and Issue Briefs*, 2007. DTIC Document, Available from: http://

www.fas.org/sgp/crs/terror/RL32114.pdf [Accessed on 9 May 2013].

4. Stock B, Göbel J, Engelberth M, Freiling F C, Holz T. Walowdac—analysis of a peer-to-peer botnet. In *2009 European Conference on Computer Network Defense*. IEEE: California, 2009; 13–20. DOI:10.1109/EC2ND.2009.10. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5494343 [Accessed on 9 May 2013].
5. Zhang Y, Xiao Y, Ghaboosi K, Zhang J, Deng H. A survey of cyber crimes. *Security and Communication Networks* 2012; **5**(4): 422–43, DOI:10.1002/sec.331.
6. Microsoft Security Intelligence. Microsoft Security Intelligence report. *Technical Report*, Microsoft, June 2010. Available from: http://www.microsoft.com/security/sir/default.aspx [Accessed on 9 May 2013].
7. TrendMicro. Global threat trends 1H 2010. *Technical Report*, TrendMicro, 2010.
8. Menten LE, Chen A, Stiliadis D. NoBot: embedded malware detection for endpoint devices. *Bell Labs Technical Journal* 2011; **16** (1): 155–170, DOI:10.1002/bltj.
9. Goebel J, Holz T. Rishi: identify bot contaminated hosts by IRC nickname evaluation. In *HotBots'07: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*. USENIX Association: Berkeley, CA, USA, 2007; 8.
10. Li Z, Wang B, Li D, Chen H, Liu F, Hu Z. The aggregation and stability analysis of network traffic for structured-P2P-based botnet detection. *Journal of Networks* 2010; **5** (5): 517–526, Available from: http://academypublisher.com/ojs/index.php/jnw/article/viewArticle/0505517526 [Accessed on 9 May 2013].
11. Jaikumar P, Kak AC. A graph-theoretic framework for isolating botnets in a network. *Security and Communication Networks* 2012; **5**: 1939–0122, DOI:10.1002/sec.500, Available from: http://dx.doi.org/10.1002/sec.500 [Accessed on 9 May 2013].
12. Moon YH, Kim E, Hur SM, Kim HK. Detection of botnets before activation: an enhanced honeypot system for intentional infection and behavioral observation of malware. *Security and Communication Networks* 2012, DOI:10.1002/sec.431, Available from: http://dx.doi.org/10.1002/sec.431 [Accessed on 9 May 2013].
13. García S, Zunino A, Campo M. Botnet behavior detection using network synchronism. In *Privacy, Intrusion Detection and Response: Technologies for Protecting Networks*, Kabiri P (ed). IGI Global: Pensilvania, 2012; 122–144. DOI:10.4018/978-1-60960-836-1.ch005, Available from: http://sebastian-garcia.isistan.unicen.edu.ar/publications/garciachap_kabiri_book.pdf?attredirects=0 [Accessed on 9 May 2013].
14. Shabtai A, Potashnik D, Fledel Y, Moskovitch R, Elovici Y. Monitoring, analysis, and filtering system for purifying network traffic of known and unknown malicious content. *Security and Communication Networks* 2011; **4**(8): 947–965, DOI:10.1002/sec.229.
15. Kuwabara K, Kikuchi H, Terada M, Fujiwara M. Heuristics for detecting botnet coordinated attacks. In *2010 International Conference on Availability, Reliability and Security*. IEEE: California, 2010; 603–607. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5438029 [Accessed on 9 May 2013].
16. Bailey M, Cooke E, Jahanian F, Xu Y, Karir M. A survey of botnet technology and defenses. In *Cybersecurity Applications & Technology Conference For Homeland Security*. IEEE: California, 2009; 299–304. Available from: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4804459 [Accessed on 9 May 2013].
17. Lim SY, Jones A. Network anomaly detection system: the state of art of network behaviour analysis, *International Conference on Convergence and Hybrid Information Technology, 2008.* (ICHIT '08), Daejeon, Korea, 2008; 459–465.
18. Zadeh MJS, Zeidanloo HR, Safari M, Zamani M, Amoli PV. Taxonomy of botnet detection techniques. In *2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*. IEEE: California, 2010; 158–162. DOI:10.1109/ICCSIT.2010.5563555, Available from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5563555 [Accessed on 9 May 2013].
19. Zhu Z, Lu G, Chen Y, Fu ZJ, Roberts P, Han K. Botnet research survey. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*. IEEE: California, 2008; 967–972. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4591703 [Accessed on 9 May 2013].
20. Feily M, Shahrestani A, Ramadass S. A survey of botnet and botnet detection. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE: California, 2009; 268–273. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5210988 [Accessed on 9 May 2013].
21. Stinson E, Mitchell J. Towards systematic evaluation of the evadability of bot/botnet detection methods. In *Proceedings of the 2nd Conference on USENIX Workshop on Offensive Technologies*.

USENIX Association: Berkeley, CA, USA, 2008; 1–9. Available from: http://portal.acm.org/citation.cfm?id=1496707 [Accessed on 9 May 2013].

22. Estrada V, Nakao A. A survey on the use of traffic traces to battle internet threats. In *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE: California, 2010; 601–604. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5432468 [Accessed on 9 May 2013].
23. Liu J, Xiao Y, Ghaboosi K, Deng H, Zhang J. Botnet: classification, attacks, detection, tracing, and preventive measures, 2009; 1–12. Available from: http://mts.hindawi.com/utils/getacceptedmsfile.aspx?msid=692654&vnum=2&ftype=manuscript [Accessed on 9 May 2013].
24. Li C, Jiang W, Zou X. Botnet: survey and case study. In *2009 Fourth International Conference on Innovative Computing Information and Control ICICIC*, Vol. 0. IEEE: California, 2009; 1184–1187. DOI:10.1109/ICICIC.2009.127, Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5412718 [Accessed on 9 May 2013].
25. TrendMicro. Taxonomy of botnet threats. *Technical Report*, TrendMicro, 2006. Available from: http://www.cs.ucsb.edu/ kemm/courses/cs595G/TM06.pdf [Accessed on 9 May 2013].
26. Tyagi AK, Aghila G. A wide scale survey on botnet. *International Journal of Computer Applications* 2011; **34**(9): 9–22.
27. Gu G, Zhang J, Lee W. BotSniffer: detecting botnet command and control channels in network traffic, *Proceedings of the 15th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2008.
28. Gu G, Perdisci R, Zhang J, Lee W. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *SS'08: Proceedings of the 17th Conference on Security Symposium*. USENIX Association: Berkeley, CA, USA, 2008; 139–154.
29. Gu G, Porras P, Yegneswaran V, Fong M, Lee W. Bothunter: detecting malware infection through IDS-driven dialog correlation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. USENIX Association: California, 2007; 1–16. Available from: http://portal.acm.org/citation.cfm?id=1362903.1362915 [Accessed on 9 May 2013].
30. Lu W, Rammidi G, Ghorbani AA. Clustering botnet communication traffic based on *N*-gram feature selection. *Computer Communications* 2010; **34** (3): 502–514, in press, Available from: http://www.sciencedirect.com/science/article/B6TYP-4YWC159-1/2/ea30180085250d194d113823a5f7a4ce [Accessed on 9 May 2013].
31. Wurzinger P, Bilge L, Holz T, Goebel J, Kruegel C, Kirda E. *Automatically Generating Models for Botnet Detection*. Springer: Berlin, 2010; 232–249. Available from: http://www.springerlink.com/index/64U6455075084115.pdf [Accessed on 9 May 2013].
32. Collins MP, Shimeall TJ, Faber S, Janies J, Weaver R, Shon MD, Kadane J. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM: New York, NY, USA, 2007; 93–104.
33. Passerini E, Paleari R, Martignoni L, Bruschi D. FluXOR : detecting and monitoring fast-flux service networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Lecture Notes in Computer Science. Springer: Berlin, 2008; 186–206.
34. Strayer WT, Walsh R, Livadas C, Lapsley D. Detecting botnets with tight command and control, *Proceedings of the 31st IEEE Conference on Local Computer Networks*, 2006; 195–202. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4116547&tag=1 [Accessed on 9 May 2013].
35. Yen TF, Reiter M. Traffic aggregation for malware detection. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 5137/2008, Heidelberg SB (ed), Lecture Notes in Computer Science. Springer: Berlin, 2008; 207–227.
36. Yu X, Dong X, Yu G, Qin Y, Yue D, Zhao Y. Online botnet detection based on incremental discrete Fourier transform. *Journal of Networks* May 2010; **5** (5): 568, Available from: http://ojs.academypublisher.com/index.php/jnw/article/view/853 [Accessed on 9 May 2013].
37. Kim D, Lee T, Kang J, Jeong H. Adaptive pattern mining model for early detection of botnet-propagation scale. *Security and Communication Networks* 2011, DOI:10.1002/sec. Available from: http://onlinelibrary.wiley.com/doi/10.1002/sec.366/full [Accessed on 9 May 2013].
38. Nguyen T, Armitage G. A survey of techniques for Internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials* 2008; **10** (4): 56–76, DOI:10.1109/SURV.2008.080406. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4738466 [Accessed on 9 May 2013].
39. Maloof MA. Some basic concepts of machine learning and data mining. In *Machine Learning and Data Min-*

ing for Computer Security Methods and Applications. Springer: London, 2006.
40. Vallacher RR, Wegner DM. What do people think they're doing? Action identification and human behavior. *Psychological Review* 1987.
41. Kantardzic M, Press I. Data Mining: Concepts, Models, Methods, and Algorithms 2003; **94** (1): 3–15, Available from: http://www.lavoisier.fr/livre/notice.asp?id=OKLWR3A3SA3OWF [Accessed on 9 May 2013].
42. Ye N. *The Handbook of Data Mining*. Lawrence Erlbaum Associates: New Jersey, 2003. Available from: http://books.google.com/books?hl=en&lr=&id=vC3gdPDaf7IC&oi=fnd&pg=PR18&dq=The+handbook+of+data+mining&ots=SvD1obX1JX&sig=nod2L9tpPXWYK8QbD6JQ81pwI3M [Accessed on 9 May 2013].
43. Barrett D, Silverman R, Byrnes R. *SSH, the secure shell: the definitive guide*. O'Reilly Media, Inc.: California, 2005. Available from: http://portal.acm.org/citation.cfm?id=1199540 [Accessed on 9 May 2013].
44. Witten I. Data mining: practical machine learning tools and techniques. *Machine Learning* 2005, ISBN: 978-0-12-374856-0 Available from: http://www.cs.waikato.ac.nz/ml/weka/book.html [Accessed on 9 May 2013].
45. Hispasec. *Virus Total*. Available from: http://www.virustotal.com/es/.
46. Yegneswaran V, Saidi H, Porras P, Sharif M. Eureka: a framework for enabling static analysis on malware. In *ESORICS '08 Proceedings of the 13th European Symposium on Research in Computer Security,* April. Springer-Verlag: Berlin, 2008; 481–500. Available from: http://dx.doi.org/10.1007/978-3-540-88313-5_31 [Accessed on 9 May 2013].
47. Huberman B A. Sociology of science: big data deserve a bigger audience. *Nature* 2012; **482**(7385): 308–308, DOI:http://dx.doi.org/10.1038/482308d.
48. Rehak M, Pechoucek M, Grill M, Stiborek J, Bartos K, Celeda P. Adaptive multiagent system for network traffic monitoring. *Intelligent Systems,* IEEE 2009; **24** (3): 16–25, DOI:10.1109/MIS.2009.42.
49. Francois J, Wang S, Bronzi W, State R, Engel T. BotCloud: detecting botnets using mapReduce, *2011 IEEE International Workshop on Information Forensics and Security (WIFS)*, Tenerife, Spain, 2011; 1–6. DOI:10.1109/WIFS.2011.6123125.