Machine learning for identifying botnet network traffic

Stevanovic, Matija; Pedersen, Jens Myrup

Publication date:
2013

Document Version
Accepted author manuscript, peer reviewed version

Machine learning for identifying botnet network traffic

(Technical report)

Matija Stevanovic and Jens Myrup Pedersen
Networking and Security Section, Department of Electronic Systems
Aalborg University, DK-9220 Aalborg East, Denmark
Email: {mst, jens}@es.aau.dk

***Abstract*—During the last decade, a great scientific effort has been invested in the development of methods that could provide efficient and effective detection of botnets. As a result, various detection methods based on diverse technical principles and various aspects of botnet phenomena have been defined. Due to promise of non-invasive and resilient detection, botnet detection based on network traffic analysis has drawn a special attention of the research community. Furthermore, many authors have turned their attention to the use of machine learning algorithms as the mean of inferring botnet-related knowledge from the monitored traffic. This paper presents a review of contemporary botnet detection methods that use machine learning as a tool of identifying botnet-related traffic. The main goal of the paper is to provide a comprehensive overview on the field by summarizing current scientific efforts. The contribution of the paper is three-fold. First, the paper provides a detailed insight on the existing detection methods by investigating which bot-related heuristic were assumed by the detection systems and how different machine learning techniques were adapted in order to capture botnet-related knowledge. Second, the paper compares the existing detection methods by outlining their characteristics, performances, and limitations. Special attention is placed on the practice of experimenting with the methods and the methodologies of performance evaluation. Third, the study indicates limitations and challenges of using machine learning for identifying botnet traffic and outlines possibilities for the future development of machine learning-based botnet detection systems.**

***Keywords*—Botnet, Botnet detection, State of the art, Traffic analysis, Machine learning**

I. Introduction

The growing reliance on the Internet has introduced numerous challenges to the protection of the privacy, integrity and security of user data. During the last two decades, the use of the Internet and Internet-based applications has experienced a tremendous expansion to the point at which they have become an integral part of our lives, supporting a wide range of services, such as banking, commerce, healthcare, public administration and education. Although convenient, the use of Internet-based services poses a number of security challenges. The main security threat and the main carrier of malicious activities on the Internet is malicious software, also known as malware. Malware implements a variety of malicious and illegal activities that disrupt the use of a compromised computer and jeopardize the security of the user's data. In parallel with the development and expansion of Internet-based services, malware has also undergone a tremendous development, improving it's mechanisms of propagation, malicious activity, and resilience to take-down efforts.

The latest incarnation of malware is the notorious bot malware. Bot malware is a state of the art malware class that successfully integrates advanced malicious techniques used by other contemporary malware classes, such as viruses, trojans, rootkits, worms, etc [1], [2]. Furthermore, bot malware has one strength comparing to other malware classes. The advantage of bot malware is an ability to communicate with an attacker through a specially deployed Command and Control (C&C) communication channel [3]–[5]. Once loaded onto a client computer the bot malware compromises the vulnerable machine and, using the C&C channel, puts it under the remote control of the attacker. The attacker is popularly referred to as the Botmaster or Botherder, while compromised hosts are known as Bots or Zombies [6]. Using a deployed C&C channel botmaster can remotely control the behaviour of the bot malware, making the operation of the bot more flexible and adaptable to the botmaster's needs. A Botnet is a usually large collection of computers that are infected with the specific bot malware.

Controlled and coordinated by the botmaster, botnets represent a collaborative and highly distributed platform for the implementation of a wide range of malicious and illegal activities. Botnets may range in size from a couple of hundred to several million bots [7], [8]. In addition, botnets can span over home, corporate and educational networks, while covering numerous autonomous systems operated by different Internet Service Providers (ISPs). Estimations of a number of bot-infected computers globally differ greatly, where some recent cyber-security studies [1], [9] claim that more than 16% of computers connected to the Internet have are infected with some kind of bot malware, thus being actively or passively involved in the malicious activities of botnets. Since botnets include such a large number of bots, they have enormous bandwidth and computational power at their disposal. However the power of botnets is not only determined by the sheer size of botnets but also by malicious activities they implement. Some of the malicious activities botnets implement are sending SPAM e-mails, launching Distributed Denial of Service (DDoS) attacks, malware and adware distribution, click fraud, the distribution of illegal content, collecting of confidential information and attacks on industrial control systems and other critical infrastructure [1], [10], [11]. On this basis it can be concluded that botnets are rightfully regarded as the most

powerful tool for implementing cyber-attacks today [12].

In order to successfully mitigate security threats posed by botnets, innovative and sophisticated neutralization mechanisms are required. The neutralization of botnets is realized through a set of techniques that detect the existence of botnets, analyse their behaviour, and implement appropriate defence measures [1], [10], [13]. The techniques involve technical, legal, sociological and often political aspects, defining the neutralization of botnets as an interdisciplinary and often complex undertaking. Botnet detection is one of the most important neutralization techniques as it provides an initial indication of the existence of compromised computers. Botnet detection is, in fact, the main prerequisite of all other neutralization actions. Furthermore, botnet detection is an intriguing research topic that attracts a lot of attention within the scientific community. As a result, many experimental detection methods have been reported in the literature over the last decade [9], [10], [14], [15]. These detection methods are based on numerous technical principles and assumptions about the behaviour of bots and about the patterns of network traffic produced by botnets. However, one of the most prominent classes of botnet detection methods is the class based on identifying network traffic produced by botnets. In addition to relying on traffic analysis for botnet detection, many contemporary approaches use machine learning techniques as a mean of identifying suspicious traffic.

The main assumption of the machine learning-based methods is that botnets create distinguishable patterns within the network traffic and that these patterns could be efficiently detected using machine learning algorithms (MLAs). The detection based on network traffic analysis by MLAs promises a flexible detection that does not require traffic to exhibit any anomalous characteristics. This class of detection methods does not require prior knowledge of botnet traffic patterns, but infers the knowledge solely from the available observations. Various detection methods have been developed using an array of MLAs deployed in diverse setups. These methods target different types of botnets by assuming varying botnet-related heuristics. Furthermore, the detection methods have not been evaluated using identical evaluation and testing methodologies. The great number of diverse detection solutions has introduced a need for a comprehensive approach to summarizing and comparing existing scientific efforts [9].

A number of authors including Hogben et al. [1], Silva et al. [9], Zhu et al. [16], Li et al. [11], Zhang et al. [17] and Liu et al. [13] have attempted to describe the field of botnet protection through series of survey papers. Although the surveys provide a comprehensive overview of the field, they only briefly address contemporary detection approaches. In parallel, several authors, such as Zeidanloo et al. [15], Feily et al. [14] and Bailey et al. [10], have also summarized scientific effort of detecting the botnets while proposing novel taxonomies of detection methods, introducing different classes of botnet detection and presenting some of the most prominent methods within the defined classes. The authors have acknowledged the potential of machine learning-based approaches in providing efficient and effective detection, but they have not provide a deeper insight on specific methods, neither the comparison of the approaches by detection performances and evaluation practice. Masud et al. [18] and Dua et al. [19] have analysed the general role of machine learning within modern cyber-security. The authors have outlined the benefits of using machine learning for discovering the existence of the malware on both network and client levels. However the authors have not provided an overview of the state of the art on botnet detection, leaving the question of current trends within the field of botnet detection unanswered.

To the best of our knowledge this paper is the first to provide up-to-date analysis of existing botnet detection methods that are based on machine learning. The paper presents the systematic overview of contemporary detection methods, with the goal of contributing to the better understanding of capabilities, limitations and opportunities of using machine learning for identifying botnet traffic. The contribution of the paper is three-fold. First, the paper provides a detailed insight on the field by summarizing current scientific efforts, thus giving the precise picture what has been done within the field. The paper analyses existing detection methods by investigating which bot-related heuristic were assumed by the detection systems and how different machine learning techniques were adapted in order to capture botnet-related knowledge. Second, the paper compares the existing detection methods by outlining their capabilities, limitations and performances of detection. Special attention is placed on practice of experimenting with the methods and methodologies of performance evaluation. Third, the paper indicates challenges and the limitations of the use of machine learning for identifying botnet traffic and outlines possibilities for the future development of machine learning-based botnet detection systems.

The rest of the paper is organized as follows. Section II examines the botnet phenomenon through the analysis of botnet life-cycle, C&C communication channel, and resilience techniques botnets deploy. Different aspects of botnet phenomenon are addressed in the light of their influence on the detection of botnets. Section III presents botnet detection through the analysis of basic principles of modern detection approaches. The section places special emphasis on botnet detection based on traffic analysis and the use of machine learning for identifying botnet-related traffic. Section IV introduces the principles of analysis the methods will be subjected to. State of the art on botnet detection based on machine learning is presented in Section V. This section present the most prominent modern detection approaches by analysing their characteristics, capabilities and limitations. The discussion of the presented scientific efforts and possibilities for future improvements is given by Section VI. Finally, Section VII concludes the paper by summarizing the findings of the review and outlining the opportunities for future work on machine-learning botnet detection.

II. The Botnet Phenomenon

Botnets represent a complex and sophisticated phenomenon that deploys a variety of advanced techniques of C&C communication and malicious activities. Additionally the attackers equip their botnets with a broad spectrum of resilience functionalities [17], [20]–[22] that are specially developed to make detection much harder and sometimes even impossible. An understanding of the operation and functionalities of botnets is crucial for the development of novel detection methods and for qualified reflection on contemporary detection systems.

The complexity of botnet phenomenon is best understood by analysing botnet life-cycle, C&C communication channel, and techniques ensuring the resilient and stealthy operation of botnets. The following chapters present the three main aspects of botnet phenomenon in more detail.

A. Botnet life-cycle

Botnet operation can be addressed through the analysis of botnet life-cycle i.e. the set of bot's functional phases observable during the botnet operation. The detection approaches target specific phases of botnet life-cycle, by utilizing specific heuristics of botnet behaviour within these phases. Therefore, the understanding of the botnet life-cycle is crucial to the successful analysis of the existing work on botnet detection. The botnet life-cycle has been described as a set of states by several authors, such as Silva et al. [9], Feily et al. [14], and Z. Zhu et al. [16]. These authors defined the botnet life-cycle in the similar fashion to each other, dividing the botnet operation into three distinct phases: the infection phase, the communication phase and the attack phase. Although, there are some differences in the authors' definition of the three operational phases, the botnet life-cycle can be generalized as illustrated in Figure 1.

The first phase of the botnet life-cycle is the Infection phase in which vulnerable computers are compromised by the bot malware, thus becoming zombies within a specific botnet. Usually this phase can be further divided into two sub-phases known as Initial Infection and Secondary Infection. During the initial infection sub-phase computers are infected by malicious piece of software known as a "loader". The initial infection can be realized in different ways, for instance, through the unwanted download of malware from malicious websites, through the download of infected files attached to email messages, by propagation of malware from infected removable disks, etc. The loader primary role is to assist in obtaining the bot malware binary. Upon successful initial infection, the secondary infection sub-phase start, during which the loader downloads the malware binary from an external network location and installs in on the vulnerable machine. The bot malware binaries can be downloaded using diverse protocols, such as FTP (File Transfer Protocol), HTTP (Hypertext Transfer Protocol) / HTTPS (Hypertext Transfer Protocol Secure) or some of the P2P (Peer-to-Peer) transfer protocol.

The second phase of the botnet life-cycle the Communication phase. This phase includes several botnet operational modes that entail communication between compromised computers and C&C servers. The communication phase covers communication devoted to receiving instructions and updates from the botmaster, as well as the reporting on the current status of bots. The communication covers several modes of operation: initial connection attempts to the C&C server upon successful infection phase, connection attempts by the bot after reboot of the compromised machine, periodical connection attempts in order to report the status of the infected machine, as well as the connection attempts initiated by the C&C server in order to update malware code or propagate instructions to bots. The communication between zombie and the C&C server is realized using the C&C channel that can be implemented in different ways. The C&C channel is presented in more detail in the following chapter.

The third phase of botnet life-cycle is marked as Attack phase as it includes bot operation aimed at implementing attackers' malicious agenda. During attack phase zombie computer may launch DDoS attacks, start SPAM e-mail campaigns, perform distribution of the stolen identities, deploy click-fraud, manipulate online reputation systems and surveys, etc [1], [10], [11]. In this operational phase the bots can also implement propagation mechanism, such as scanning for vulnerable computers or distributing malicious software. The second and the third phase are functionally linked so they are usually altering one after another, once a vulnerable computer is successfully infected. However it should be noted that different phases within the botnet life-cycle can last for different time spans, and that the length of a specific phase can vary depending on the attack campaign the bot implements.

B. C&C channel

Command and Control (C&C) channel is the main carrier of botnet functionality and the defining characteristic of bot malware. The C&C channel represents a communication channel established between the botmaster and compromised computers. This channel is used by the attacker to issue commands to bots and receive information from the compromised machines [3]–[5]. The C&C channel enables remote coordination of a large number of bots, and it introduces the level of flexibility in botnet operations by creating the ability to change and update malicious botnet code. As the crucial element of the botnet phenomenon, the C&C channel is often seen as one of the most important indicators of botnet presence and thus one of the most valuable resources for botnet detection.

C&C communication infrastructure has been rapidly evolving over a recent years. As a result, several control mechanisms in terms of protocols and network architecture have been used to realize the C&C channel [3]–[5], [23]–[25]. On the basis of topology of the C&C network, botnets can be classified as botnets with centralized, decentralized or hybrid network architecture. The three types of botnet network topologies are illustrated in Figure 2.

Centralized botnets have centralized C&C network architecture, where all bots in a botnet contact one or several C&C servers owned by the same botmaster (Figure 2a). Centralized C&C channels can be realized using various communication protocols, such as IRC (Internet Relay Chat), HTTP and HTTPS. IRC-based botnets are created by deploying IRC servers or by using IRC servers in public IRC networks. In this case, the botmaster specifies a chat channel on a IRC server to which bots connect to in order to receive commands. This model of operation is referred to as the push model [26], as the botmaster "pushes" commands to bots. HTTP-based botnets are another common type of centralized botnets, that rely on HTTP or HTTPS transfer protocol to transfer C&C messages. In contrast to bots in IRC-based botnets, bots in HTTP-based botnets contact a web-based C&C server notifying their existence with system-identifying information via HTTP or HTTPS requests. As a response, the malicious server sends back commands or updates via counterpart response messages. This model of operation is referred to as the pull model [26], as bots have to "pull" the commands from the centralized C&C server. The IRC-and and the HTTP-based botnets are

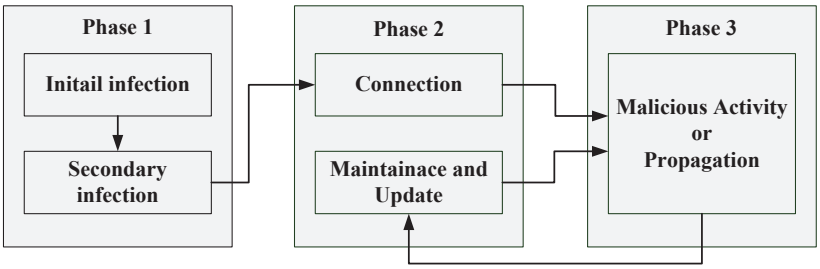

Fig. 1. Botnet life-cycle

easy to deploy and manage, and they are very efficient in implementing the botmasters' malicious agenda, due to the low latency of command messages. For this reason, the IRC- and the HTTP-based C&C have been widely used for deploying botnets. However, the main drawback of botnets with centralized network architecture is that they are vulnerable to the single point of failure. That is, once the C&C servers have been identified and disabled, the entire botnet could be taken down.

Decentralized botnets represent a class of botnets developed with the goal of being more resilient to neutralization techniques. Botnets with decentralized C&C infrastructure have adopted P2P (Peer-to-Peer) communication protocols as the means of communicating within a botnet [5], [23]. This implies that bots belonging to the P2P botnet form an overlay network in which the botmaster can use any of the bots (P2P nodes) to distribute commands to other peers or to collect information from them (Figure 2b). In these botnets, the botmaster can join and issue commands at any place or time. P2P botnets are realized either by using some of the existing P2P transfer protocols, such as Kademlia [27], Bittorent [28] and Overnet [29], or by custom P2P transfer protocol. While more complex and perhaps more costly to manage and operate compared to centralized botnets, P2P botnets offer higher resiliency, since even if the significant portion of the P2P botnet is taken down the remaining bots may still be able to communicate with each other and with the botmaster, thus pursuing their malicious purpose. However, P2P botnets have one major drawback. They cannot guarantee high reliability and low latency of C&C communication, which severely limits the overall efficiency of orchestrating attacks.

Some of the recent botnets [30] have adopted more advanced hybrid network architectures (Figure 2c), that combines principles of centralized and decentralized C&C network architectures. This class of botnets uses advanced hybrid P2P communication protocols that try to combine resiliency of P2P botnets with the low latency of communication of centralized botnets. The hybrid botnet architecture has been investigated by several groups of authors, such as Wang et al. [24] and Z. Zhang et al. [25]. The authors suggest that in order to provide both resilience and low latency of communication hybrid botnets should be realized as networks in which bots are interconnected in P2P fashion and organized in two distinct groups: the group of proxy bots and the group of working bots. Working bots would implement the malicious activity while proxy bots would provide the propagation of C&C messages from and to the botmaster. Working bots would periodically connect to the proxy bots in order to receive commands. Based on the work presented in [24], [25] this topology should provide a resilience to take down efforts as well as improvements in latency of C&C messages comparing to the regular P2P botnets.

C. Resilience techniques

One of the primary goals of the botnet operation is flying under the radar of botnet detection and neutralization systems. Therefore attackers equip their botnets with a diversity of resilience techniques capable of providing the stealthiness and robustness of operation. Implemented at network level, resilience techniques have a goal of providing secrecy and integrity of the communication, anonymity of the botmaster, and robustness of the C&C channel to take down efforts. Some of the most important means of providing secrecy of C&C communication are obfuscation of existing and development of custom communication protocols, as well as the encryption of the communication channel. Using these techniques the security and the integrity of communication are preserved, thus efficiently defeating detection methods that rely on content of the traffic payloads for detection. However usage of encrypted communication channels and obfuscated communication protocols can be considered suspicious and it can be used as a trigger for additional traffic analysis. Other commonly used techniques that provide resilience of botnet operation are Fast-flux and Domain Generation Algorithm (DGA) [17].

The basic idea behind Fast-flux is to have numerous IP addresses associated with a single fully qualified domain name, where the IP addresses are swapped in and out with extremely high frequency, by changing DNS records. Fast-flux [21] is widely used by the botnets to hide phishing and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies. This way the anonymity of C&C servers (and consequently botmaster) is protected, while providing more reliable malicious service. However it should be noted that by using Fast-flux, a specific botnet heuristic is formed that can be used for efficient detection of botnets [31].

DGA (Domain Generation Algorithm) [17], [22] i.e., domain fluxing is a technique that periodically generates a large number of domain names that can be used as rendezvous points with their controllers. Bots using the DGA generate large number of pseudo-periodical domain names that are queried, to determine addresses of the C&C servers. In order for mechanism to be functional the appropriate part of the DGA algorithm is also implemented by the attacker. The attacker registers pseudo-periodical domain names corresponding to

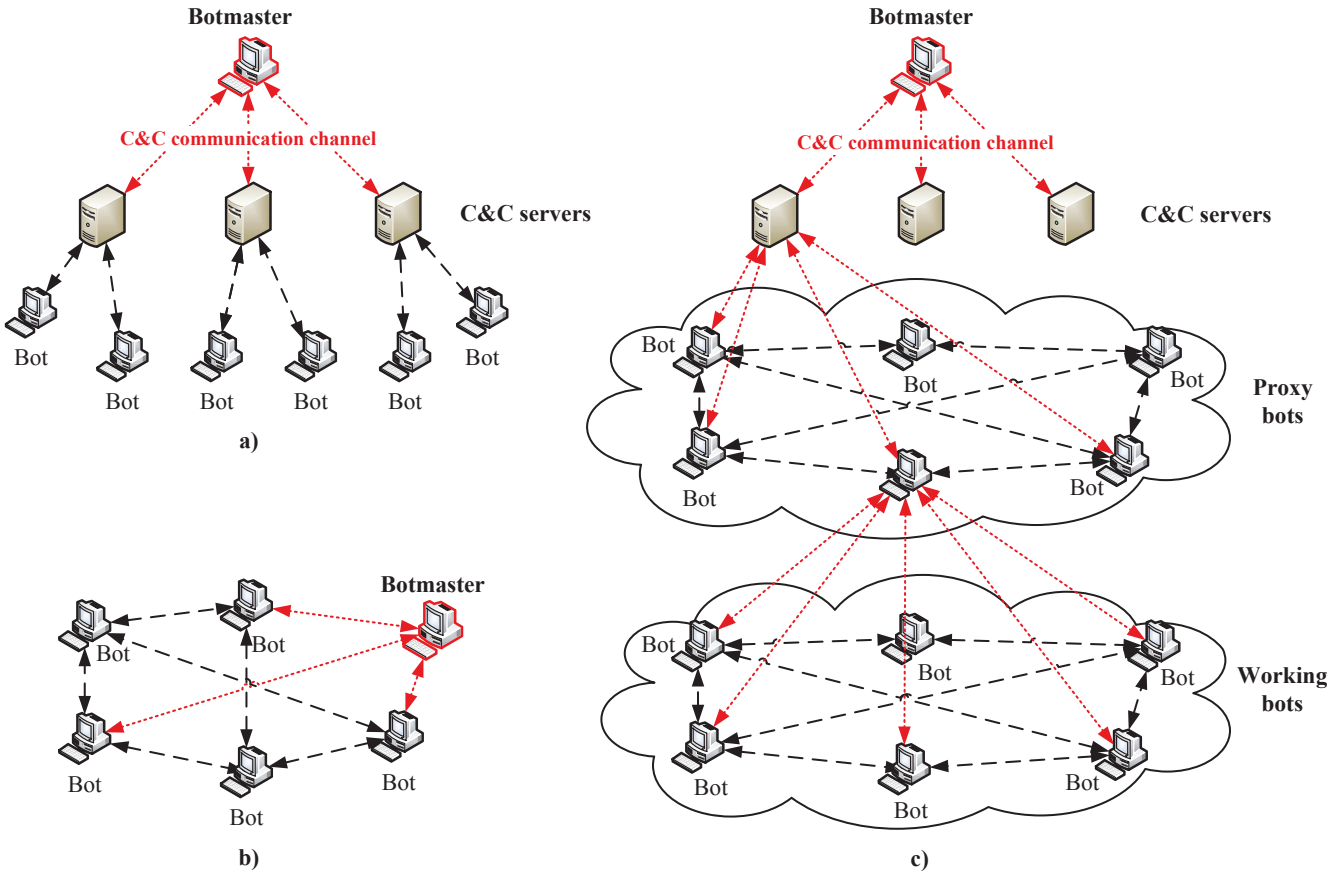

Fig. 2. Botnet architectures: a) centralized, b) decentralized and c) hybrid

the IP address of malicious servers. Although complex and hard to implement in efficient manner, the DGA algorithm has been improved over the years providing reliable mean of communication for some of the recent botnets [32]. The large number of domains makes it difficult for law enforcement to blacklist malicious domains or to detect the bots by detecting ones that contact the known malicious domain names. As in the case of Fast-flux, the DGA also introduces certain botnet heuristics that could be used for botnet detection [32]. However it should be noted the DGA is primary used as a back-up communication vector if primary communication channel fails. Using DGA as a backup strategy higher resilience and robustness of C&C communication is achieved.

In parallel with resilience techniques deployed at the network level, modern botnets also use an abundance of client level resilience techniques. These resilience techniques provide the robustness of bot malware to detection at the host computers [2], [33], [34]. Some of the most prominent techniques are code obfuscation techniques, such as polymorphism and metamorphism. These enable the bot code to mutate without changing the functions or the semantics of its payload. Hence, bot binaries in the same botnet are usually different from each other. Using these techniques bot malware evades conventional detection solutions that depend on signatures of malware binaries. Other client level resilience techniques are taint the bot malware behaviour at the client computer and attack the system for monitoring client level forensics [35]. Finally, one of the most challenging malicious technique deployed by the bot malware at the client level is a rootkit ability [36], [37]. Having the rootkit ability the malware is able to defeat majority of malware tracking systems implemented at the host computer. Client level resilience techniques have turned to be very effective in avoiding modern detection systems, thus posing the great challenges to detection at client level. As a result, the majority of the contemporary detection methods focus on the analysis of network traffic produced by bots, as the defining aspect of the botnet phenomena [1], [9].

III. Botnet Detection

From the early 2000s, when the first detection solutions were developed, many experimental systems have been reported in the literature, with various goals, and based on diverse technical principles and varying assumptions about bot behaviour and traffic patterns [9], [10], [14], [15]. Depending of the point of deployment detection approaches can generally be classified as client-based or network-based.

Client-based detection approaches are deployed at the client computer targeting bot malware operating at the compromised machine [18], [38]–[43]. These methods detect the presence of bot malware by examining different client level forensics, for instance, application and system logs, active processes, key-logs, usage of the resources and signature of binaries. Furthermore, the client-based detection can also include examination of traffic visible on the computer's network

interfaces [44]–[46].

Network-based detection, on the other hand, is deployed at an "edge" of the network (usually in routers or firewalls), providing botnet detection by analysing network traffic. This class of methods identifies botnets by recognizing network traffic produced by them within all three phases of bots life-cycle. These approaches are usually referred to as intrusion detection systems (IDS) or intrusion prevention systems (IPS) [9], [14].

In parallel with conventional network- and client-based detection methods a novel class of hybrid detection methods has emerged [46]–[50]. This class of methods concludes about the existence of botnets on the basis of observations gathered at both client and network levels. The main hypothesis behind hybrid approaches is that it is possible to provide significant improvements in performances of botnet detection by correlating findings from independent client- and network-based detection systems.

There are several conceptual differences between client- and network-based detection which make detection based on the traffic analysis often seen as a more promising solution. As mentioned in the previous section, client-based detection systems are highly vulnerable to the variety of client level resilience techniques. Attackers place a great and, most of all, continuous effort in making the presence of bot malware undetectable at the compromised machine [2], [33], [34]. Furthermore the detection systems that detect presence of bot malware at client computers are only able to identify the individual compromised hosts and the C&C servers contacted by it. Finally, an extensive deployment of the client-based detection systems is burdened by the practical challenges of deploying the detection system to a large number of clients machines. On the other hand, the network-based detection is targeting the essential aspects of botnet functioning, i.e. network traffic produced as the result of botnet operation. Network-based approaches assume that in order to implement its malicious functions botnets have to exhibit certain network activity. This assumption is supported by following reasoning. First, in order to make their operation more stealthy botnets have to limit the intensity of attack campaigns (sending SPAM, launching DDoS attacks, scanning for vulnerabilities, etc.) and taint and obfuscate the C&C communication channel. However this contradicts the goal of providing the most prompt, powerful and efficient implementation of malicious campaigns. Second, network level resilience techniques harden the detection but they also introduce the additional botnet heuristics that can be used for detection [31], [32]. The main advantage of network-based detection is the fact that it has wider scope then the client-level detection systems. The network-based detection is pushed further away from the actual hosts so it is able to capture the traffic from a large number of client machines. This provides the ability of capturing additional aspects of botnet phenomena, for instance, group behaviour of bots within the same botnet [51], [52], time dependency of bots activity and diurnal propagation characteristics of botnets [53].

A. Network-based detection

Network-based detection is based on analysis of network traffic in order to identify presence of compromised computers. This class of detection methods detects botnets by identifying traffic produced by botnets operating in all three phases of botnet life-cycle. The traffic is usually analysed on either packet level or flow level. Flows are usually defined as 5-tuple consisting of: source and destination IP addresses, source and destination ports and protocol identifier. The flow level analysis can generally catch a more finite characteristics of botnet-related communication, while the packet level analysis can provide more information on the attack vectors as it inspects the packet payload. Additionally, as the flow level analysis does not require access to the packet payload it is less privacy evasive comparing to the packet level analysis. Network-based detection can be classified based on several aspect, such as the point of implementation, the stealthiness of operation, and the basic principles of functioning.

Detection approaches based on traffic analysis can generally be deployed at different points in the network, where the main difference between methods is in the network scope they cover. By analysing traffic at the client machine only one compromised machine can be detected while implementing the detection system further from the client would include traffic from more hosts. However implementing the traffic monitoring in the higher network tiers also implies the need for processing larger amount of data.

Based on the stealthiness of functioning the methods can be classified as Passive or Active detection techniques. The passive detection approaches do not interfere with botnet operation directly, but operate based on observation only, which makes them stealthy in their operation and undetectable by the attacker. Active detection methods, on the other hand, are more invasive methods that actively disturb botnet operation by interfering with malicious activities or the C&C communication of the bots. Additionally, these techniques often target specific heuristics of the C&C communication or the attack campaign, providing higher precision of detection at the expense of flexibility and generality of the approach. The passive approaches on the other hand have an advantage of being able to detect wider range of botnet types, by deriving the pattern of malicious traffic from the observation only. Majority of botnet detection approaches are passive while only few as [54] are active.

In parallel with the classification of botnet detection based on the place of implementation or stealthiness of functioning the methods can be classified based on their functional characteristics as Signature- or Anomaly-based methods. Signature-based methods are based on recognizing characteristic patterns of traffic, also known as "signatures" [55]–[58]. The signature-based detection performs packet level traffic analysis by using deep packet inspection (DPI) to recognize signatures of malicious payloads. This class of detection techniques covers all three phases of botnet life-cycle and it is able to detect known botnets with high precision. The main drawback of signature-based approaches is that they are able of detecting only known threats, and that efficient use of these approaches requires constant update of signatures. Additionally these techniques are liable of various evasion techniques that change signatures of botnet traffic and malicious activities of bots, such as encryption and obfuscation of C&C channel, Fast-flux and DGA techniques, etc.

Anomaly-based detection is a class of detection methods that is devoted to the detection of traffic anomalies that can

indicate existence of malicious instances within the network [51], [59]–[63]. The traffic anomalies that could be used for detection differ from easily detectable as changes in traffic rate, latency, to more finite anomalies in flow patterns. This group of approaches can operate on both packet and flow level, targeting different botnet heuristics and using various anomaly detection algorithms. Some of the most prominent anomaly-based approaches detect anomalies in packet payloads [55], [59], DNS (Domain Name System) traffic [31], [61], [62], botnet group behaviour [51], [53], etc. The anomaly-based detection can be realized using different algorithms ranging from the statistical approaches, machine learning techniques, graph analysis, etc. In contrast to the signature-based approaches, the anomaly detection is generally able to detect new forms of malicious activity and it is more resistant to existing botnet resilience techniques. However some challenges in using anomaly-based detection still exist. This class of techniques requires the knowledge of anomalies that characterize botnet traffic. Additionally traffic produced by modern botnets is often similar to the "normal" traffic, resulting in many false positives. Finally anomaly detection methods often have to analyse a vast amount of data, which is difficult to perform in real-time, making the detection of a fine-grained anomalies in large-scale networks a prohibitive task. One of the novel and the most promising anomaly-based methods is the group of detection methods that rely on machine learning for detection of bot-related traffic patterns. The machine learning is used because it offers the possibility of automated recognition of bot-related traffic patterns without the need for traffic to exhibit specific anomalous characteristics. Additionally machine learning provide the ability of recognizing the patterns of malicious traffic without a priori knowledge about the malicious traffic characteristics.

B. Machine learning for botnet detection

The basic assumption behind machine learning-based methods is that botnets produce distinguishable patterns of traffic or behaviour within the client machine and that this patterns could be detected by employing some of the Machine Learning Algorithms (MLA) [18], [19].

Machine Learning (ML), is a branch of artificial intelligence, that has a goal of construction and studying of systems that can learn from data [64], [65]. Learning in this context implies ability to recognize complex patterns and make qualified decisions based on previously seen data. The main challenge of machine learning is how to provide generalization of knowledge derived from the limited set of previous experiences, in order to produce a useful decision for new, previously unseen, events. To tackle this problem the field of Machine Learning develops an array of algorithms that discover knowledge from specific data and experience, based on sound statistical and computational principles. Machine learning relies on concepts and results drawn from many fields, including statistics, artificial intelligence, information theory, philosophy, cognitive science, control theory and biology. The developed machine learning algorithms (MLAs) are at the basis of many applications, ranging from computer vision to language processing, forecasting, pattern recognition, games, data mining, expert systems and robotics. At the same time, important advances in the machine learning theory and algorithms have promoted machine learning to the principal mean for discovering knowledge from the abundance of data that is currently available in diverse application areas. One of the emerging application areas is botnet detection that relies on MLAs to detect the bot-related network traffic patterns.

Machine learning algorithms can be classified based on the desired outcome of the algorithm on two main classes:

1) Supervised learning
2) Unsupervised learning

Supervised learning [66] is the class of well-defined machine learning algorithms that generate a function (i.e., model) that maps inputs to desired outputs. These algorithms are trained by examples of inputs and their corresponding outputs, and then they are used to predict output for some future inputs. The Supervised learning is used for classification of input data on some defined class and for regression that predict continuous valued output.

Unsupervised learning [67] is the class of machine learning algorithms where training data consists of a set of inputs without any corresponding target output values. The goal in unsupervised learning problems may be to discover groups of similar examples within the input data, where it is called clustering, to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

In the case of the anomaly-based botnet detection, the machine learning represent the mean of classifying or clustering traffic by using some of the supervised and unsupervised machine learning algorithms. Traffic is analysed on both flow and packet level where different features of traffic are extracted. Extracted traffic features describe the traffic that characterizes specific host or server in the network, or the specific traffic flow. The more details on the traffic features used by the contemporary detection methods can be find in Section V.

In the supervised learning scenario, machine learning for botnet detection can be implemented as illustrated in Figure 3a. The supervised MLA is first trained using the training data, forming the function that maps inputs and corresponding outputs. The function, also referred to as a model is then used to classify the inputs from test data. In order to be used by the MLA both training and test data need to be appropriately pre-processed. Pre-processing is implemented by the Data Preprocessing unit that extracts the features from the available data and selects ones will be used within the MLA. Choosing the right features is one of the most challenging task of practical deployment of MLA. The features should be chosen in that way so they could capture targeted botnet heuristics. Some of the most popular supervised MLA used for botnet detection are: SVM (Support Vector Machines), ANN (Artificial Neural Networks), Decision tree classifiers, Bayesian classifier, etc.

In contrast to the supervised learning scenario, unsupervised learning scenario implies the use of unsupervised learning for the clustering of bot-related observations. The main characteristic of unsupervised MLAs is that they do not need to be trained beforehand. Unsupervised MLAs for botnet detection are deployed as illustrated in Figure 3b. These techniques pre-process available data by extracting and

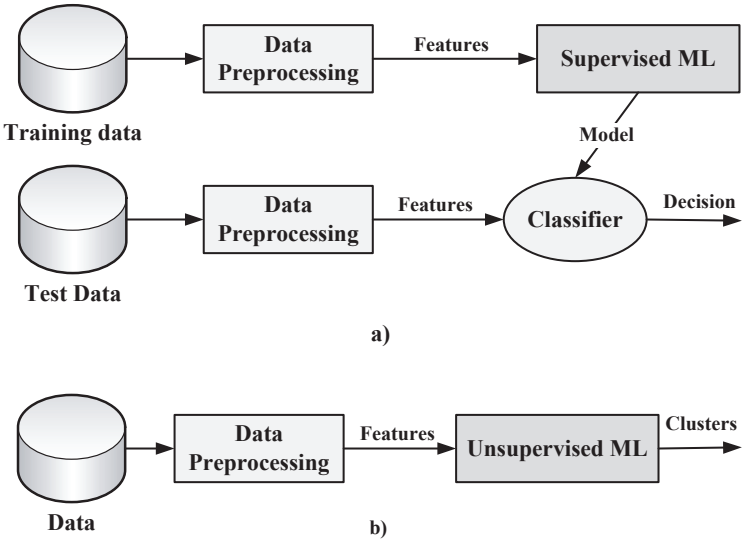

Fig. 3. Machine learning for botnet detection: a) supervised learning framework and b) unsupervised learning framework

selecting the features and then using the unsupervised MLA to cluster the observations, similar to each other, to the same cluster. The main challenges of successful implementation of these kind of learning scenario is choosing of appropriate features as well as determination of number of clusters. The most popular unsupervised learning approaches used for botnet detection are: K-means, X-means and Hierarchical clustering.

The presented scenarios for deployment of MLAs for botnet detection represent only the simplified illustration of botnet detection frameworks based on machine learning. Real-life implementations of data pre-processing usually include additional, more advanced processing in order to extract information that could successfully capture targeted botnet heuristics. In parallel with scenarios illustrated in the Figure 3 some of the modern machine learning-based approaches implement the detection through several phases, using combination of different MLAs or by deploying the MLAs in an adaptive manner. This way more fine grained, flexible, and adaptable detection can be achieved. More details on contemporary detection approaches based on machine learning, deployment of machine learning algorithms and performances they provide can be found in Section V.

IV. The principles of the analysis

Botnet detection approaches based on machine learning, as well as modern botnet detection approaches generally, have several goals that they try to achieve, such as:

1) Generality
2) Stealthiness
3) Timely detection
4) High detection performances
5) Robustness on evasion techniques

Through this paper we analyse the characteristics of contemporary machine learning-based botnet detection approaches and their ability to fulfil these goals. The analysis of detection methods is realized through two phases: the analysis of functional characteristics of methods and the analysis of performances of methods. The principles of the analysis are presented in more detail by the following chapters.

A. Characteristics of detection methods

The analysis of characteristics of detection methods is realized through the analysis of heuristics assumed by the approaches, the analysis of traffic features and MLAs used by the approaches, and assessment of generality, stealthiness and the ability of detection methods to provide timely detection.

The generality refers to the ability of covering the wide range of botnet types, regardless of botnet propagation mechanisms, implemented attack vectors, and the realization of the C&C communication channel. Different detection methods can target different phases of bot life-cycle i.e., the infection phase, the communication phase or the attack phase. Detection approaches that cover the communication phase can be directed at various communication protocols and network topologies (IRC, HTTP, P2P), while detection approaches that cover the attack phase can target different attack campaigns (SPAM, DDoS, etc.). Some of the methods rely on payload signatures (as described in Section III) of traffic limiting the generality of the method to known botnets. Additionally, the generality of the botnet detection depends on the bot-related heuristics assumed by the approach, and on how this heuristic relies to the real-world botnets. Detection methods that cover the specific type of botnets or the specific phase of bot life-cycle are generally more efficient then the methods that try to cover all types of botnets. However these detection techniques are at the same time less flexible to the changing nature of botnets phenomenon.

Stealthiness entails the ability of detection approach to function without being detected by the attacker, thus all passive techniques (as described in Section III) are stealthy in their operation. All detection addressed by this review are passive, thus fulfilling the stealthiness requirement.

Timely detection is a another much wanted characteristic of a detection system defined through the ability operating

efficiently and producing the detection results in "reasonable" time. The timely detection often entails a need for a detection method to operate in on-line fashion, thus being capable of processing large quantities of data efficiently. However it should be noted that the requirements of timely detection are not precisely defined, and that question of how prompt detection should be is still unanswered.

B. Performance Evaluation

The analysis of the performances of methods is realized through the analysis of the performance evaluation practices used by the methods and assessment of evasion techniques the methods are vulnerable to. The analysis of performance evaluation used within experimenting with detection methods is realized through assessment of evaluation scenarios, quantitative and qualitative aspects of evaluation data and examination of used performance metrics.

Testing and evaluation of the proposed approaches is typically realized using labelled traffic traces, i.e. traffic traces consisting of known malicious and non-malicious traffic traces [68]. Correctly labelled datasets are one of the main prerequisites of deterministic evaluation of detection performances. The malicious traffic represent traffic produced by botnets, while non-malicious traffic, often refereed to as "background" traffic, is a "clean" traffic that only contains traffic produced by non-malicious hosts. The labelled datasets is formed either by labelling previously recorded traffic trace or by combining the malicious and non malicious datasets. The labelling of the traffic can be done by using some of the existing IDS systems [56], [57] and signature-based botnet detection systems [55] or by checking IP and domain blacklists. However this way of obtaining labelled dataset is highly dependent on the precision of the labelling mechanism. Alternatively, malicious and non-malicious traffic traces can be obtained separately and than combined forming totally deterministic traffic traces. In this case, the malicious bot-related traffic traces can be obtained in the following scenarios:

1) Scenario 1: Bot-related traffic is captured by Honepots [69], [70] deployed by researchers themselves or by some third party.
2) Scenario 2: Bot-related traffic is generated within fully controllable network environments, where researchers have total control on both C&C servers and infected zombie machines. This scenario requires bot malware source code to be available. Having the source code, experiments can be realized in safe and totally controlled fashion.
3) Scenario 3: Bot-related traffic is generated in semi-controlled environments, where researchers have bot malware binary but not the bot malware code. In this scenario researchers deploy compromised machines by infecting them purposely with specific bot malware samples. Zombie computers are allowed to contact the C&C servers in order for bot-related traffic to be recorded. In order to limit any unwanted damage to the third parties on the Internet the traffic produced by the infected machines is filtered using different rate and connection limiting techniques as well as matching of the malicious signatures of bot traffic [69]. Although one of the simplest, this scenario rises many legal and ethical concerns.

Besides the way malicious traffic trace is obtained, the number of distinct bot malware samples used for evaluation of botnet detection methods is also very important for assessing the validity of obtained performance measures. Normally, the more bot malware samples of different types used within the evaluation the better. Using the traffic traces form different bot malware for training and testing could give a good indication if a method can generalize well or not.

Non-malicious traffic traces could be obtained in various ways: from self generated traffic using statistical traffic generators to the network traces recorded on LAN, enterprise, campus and in some cases even core ISP networks. However it should be noted that for the process of obtaining background traffic the primary concern is to make sure that the traffic traces are benign. This can be easily achieved on the controlled LAN network, while obtaining traffic from other "real-world" networks would need to include some kind of labelling as well. Additionally the traffic from one network to another vary, so choosing the right "background" traffic trace is also a very challenging task.

Understanding the performance metrics used is crucial to make a sound judgement of capabilities of the approach. Performance metrics used within the approaches can greatly vary but is typically express by some of the following metrics:

1) *True positives rate* (TPR) *i.e. Recall*: $TPR = recall = \frac{TP}{TP+FN}$
2) *True negative rate* (TNR): $TNR = \frac{TN}{TN+FP}$
3) *False positive rate* (FPR): $FPR = \frac{FP}{FP+TN}$
4) *False negative rate* (FNR): $FNR = \frac{FN}{TP+FN}$
5) *Accuracy*: $accuracy = \frac{TP+TN}{TP+FP+TN+FN}$
6) *Error*: $error = \frac{FP+FN}{TP+FP+TN+FN}$
7) *Precision*: $precision = \frac{TP}{TP+FP}$

Where true positive (TP) is a number of positive samples classified as positive, true negative (TN) is a number of negative samples classified as negative, false positive (FP) is a number of negative samples classified as positive, and false negative (FN) is a number of positive samples classified as negative. However it should be noted that not all of the approaches are evaluated using all of the performances metrics. The following sections presents more details on reported detection performances, evaluation practices and evaluation dataset used for the analysed detection methods.

C. Evasion techniques

Detection methods should be robust on evasion techniques in such a way that for detection to be evaded botnet should severely limit the efficiency of implementing its malicious agenda. The vulnerability of detection approaches to evasion techniques highly depend on the botnet heuristics used by

the detection method as well as technical principles on which method relies on. Rallying detection method on easily changeable botnet characteristics can lead to easy evasion, which would consequently limit the prospective use of the detection approach. Stinson et al. [35] have proposed a framework for systematic evaluation of robustness of detection methods on a series of evasion techniques. Similarly to the principles presented in [35] this paper considers several types of evasion techniques (ET) that directly affect detection approaches based on traffic analysis, such as:

1) ET1 - Evasion of host based detection: Evasion techniques that evade botnet detection at the client machine. This category includes a wide range of techniques, such as evasion by attacking process monitor and evasion by tainting bot malware behaviour at the client computer.
2) ET2 - Evasion by traffic encryption: Techniques that perform encryption of the traffic used within the C&C channel.
3) ET3 - Time-based evasion: Evasion techniques that try to avoid bot activity in specific time windows in which detection method operates, thus restricting the detection method from catching the right observations.
4) ET4 - Evasion by flow perturbation: The class of evasion techniques that change the patterns of traffic by changing the flow statistics.
5) ET5 - Evasion by performing only a subset of available attacks, thus limiting the available observation for the methods that are targeting the attack phase of botnet life-cycle.
6) ET6 - Evasion by restricting the number of attack targets, by targeting hosts at the same internal network, thus evading the methods that monitor traffic at network boundaries.
7) ET7 - Evasion of cross-host clustering by employing sophisticated schemes avoiding the group activities of bots within the same administrative domain.
8) ET8 - Evasion by coordination of bots out-of-band, by using Fast-flux and DGA algorithms as a mean of communicating, thus providing a level of privacy and resilience to malicious C&C servers.

The majority of the existing detection methods could be evaded by deploying some of the evasion techniques outlined here. However, different evasion techniques bear an implementation cost that varies from low to very high [35], often causing severe damage to the utility of the botnet. Therefore, the fact that detection system could be evaded does not necessarily mean that the cost of evasion will be justified. Please note that the paper does not address the complexities of evasion techniques and its effect on the overall utility of the botnet. Examination of the vulnerabilities of existing detection methods to the evasion techniques is presented in the following Section V.

V. State of the art: the analysis outlook

This section analyses contemporary machine learning-based botnet detection approaches, on the basis of the principles of analysis presented in Section IV. The methods are addressed in the chronological order starting from the some of the first machine learning-based detection approaches. Additionally, the methods are divided into three groups based on the point of implementation i.e. network-based, client-based and hybrid detection approaches. The review only addresses client-based and hybrid approaches that heavily rely on the network traffic analysis. Other client-based and hybrid botnet detection methods are not covered by this review.

The results of the analysis are summarized by the series of tables. The characteristics of the analysed detection approaches are summarized in Table I and Table II, where Table I gives an overview of how existing detection approaches fulfil the requirements of generality and timely detection, while Table II summarizes the MLAs and traffic features used by the approaches.

The analysis of the performance of the methods is illustrated in Table III and Table IV. The Table III gives a brief overview of evaluation practice and datasets used within the approaches as well as reported performances for analysed detection methods. However, it should noted that the results presented in the table should be taken with caution, as the values presented represent the bottom range of the performances of the methods. Additionally, the methods should not be directly compared using the reported metrics, as they used different evaluation practices and testing datasets. However, the presented performance metrics can still indicate the overall performances of the particular approach in identifying botnet traffic.

Table IV illustrates how different approaches tolerate most common evasion techniques, by indicating the strength of the indication (SF - strong factor and WF - weak factor) of the method being evaded by the evasion strategies presented in the Section IV. However, it should be noted that the indications given in the Table IV are based on the facts presented by the authors and that they should be used more as a guidelines than the precise measure.

A. Network-based detection methods

One of the first network-based botnet detection approaches that use machine learning was proposed by **Livadas et al.** [71] during 2006. The proposed approach evaluated the use of several MLAs for identifying the traffic originating from IRC-based botnets. The approach is realized in two stages. The first stage classifies traffic flows on either chat or non-chat flows, while the second stage further classify IRC chat flows on botnet or real chat flows. Both stages are realized using machine learning techniques. The first stage utilize machine learning in order to identify IRC chat flows within the total traffic, while the second stage use machine learning to classifies IRC flows on malicious or non-malicious ones. The efficiency of different machine learning techniques in identifying botnet traffic is evaluated by varying classification techniques, a set of characterization attributes and the size of the training set.

MLA and features used: The method used three different supervised MLAs for the realization of both classification phases: C4.5 decision tree classifier, Naive Bayes classifier and Bayesian network classifier [65]. The MLAs were assessed by using several flow level features such as: flow duration (numeric), maximum initial congestion window (numeric), indicator whether client or server initiated flow (categorical),

TABLE I. BOTNET DETECTION METHODS BASED ON MACHINE LEARNING - THE CHARACTERISTICS OF METHODS

Detection Method	Network / Host / Hybrid	Flow / Host -based	C&C Protocol Independent	Signature Independent	Individual **H**ost / **G**roup Activity / C&C **S**ervers	Detection Phase	On-line operation
Livadas et al. [71]	Network	Flow	IRC	x	H	2	-
Strayer et al. [72]	Network	Flow	IRC	x	H	2	x
G.Gu et al. [73]	Network	Host	x	x	G	2,3	-
Husna et al. [74]	Network	Host	x	-	H	3	-
Noh et al. [75]	Network	Flow	P2P	x	H	2,3	-
Nogueira et al. [76]	Network	Flow	x	x	H	2,3	x
Liu et al. [77]	Network	Host	P2P	x	G	2,3	-
Liao et al. [78]	Network	Flow	P2P	x	H	2,3	-
Yu et al. [79]	Network	Flow	IRC	x	H	2,3	x
Langin et al. [80]	Network	Host	P2P	x	H	2	-
H.Choi et al. [81]	Network	Flow	DNS	x	G	2,3	x
Sanchez et al. [82]	Network	Host	x	x	H	3	-
Chen et al. [83]	Network	Flow	x	x	H	2,3	x
Saad et al. [84]	Network	Flow	P2P	x	H	2	-
Zhang et al. [85]	Network	Flow	P2P	x	H	2	-
W.Lu et al. [86]	Network	Flow	IRC	-	H	2,3	-
Bilge et al. [87]	Network	Flow	x	x	S	2	x
Masud et al. [45]	Host	Flow	IRC	-	H	2,3	-
Shin et al. [44]	Host	Flow	x	x	H	2,3	-
Zeng et al. [46]	Hybrid	Flow	x	x	G	2,3	-

average byte per packet for flow (numeric), average bits per second for flow (numeric), average packets per second for flow (numeric), percentage of packets pushed in flow (numeric), percentage of packets in one of eight packet size bins (numeric), variance of packet inter-arrival time (numeric) and variance of bytes per packet for flow (numeric).

Performance evaluation: The approach was evaluated using bot-related traffic generated through a fully controlled experiment realized in accordance with the Scenario 2. Botnet traffic traces were obtained using only one bot malware sample (Kaiten bot). Background traffic was gathered from the campus network. As a result of evaluation a Bayesian network classifier showed potential in accurately classifying botnet IRC flows, with relatively high FNR (10-20%) and FPR (30-40%). Other two MLAs performed more poorly. The evaluation also showed that careful selection of the flow attributes used for the purpose of classification is of the most importance. This approach was one of the first that demonstrate the possibility of utilizing the machine learning in botnet identification. The method targets individual bots and the second phase of their life cycle, by analysing traffic on the flow level. The presented detection does not depend on the traffic payload providing detection of encrypted C&C channel. However as the method only targets IRC-based botnets its effectiveness in a real-world implementation is severely limited. In addition, the method is vulnerable on evasion by flow perturbation (strong indication).

Strayer et al. introduced a detection approach based on network behaviour and machine learning in 2008 [72]. The proposed framework represents an extension of Strayer's previous work [88] and work conducted by Livadas et al. [71]. Similar to the Livadas et al. approach, the framework utilizes several machine learning approaches in order to classify IRC traffic flows as malicious or non-malicious.

Strayer et al. approach can be divided into four stages. The first stage implements data pre-processing by filtering flows that are most likely not carrying C&C data. The filtering is based on prior knowledge of IRC bots behaviour patterns and flows characteristics. Implemented as a five level process, the filtering selects only TCP flows, eliminates scan attempts (TCP flows with only SYN or RST packets) high bit-rate flows (bulk data transfer) and brief flows (less than 2 packets or 60 seconds), and selects flows with small average packet length (less than 300 bytes). The pre-filtered flows are then sent to the second phase that implements MLAs in order to identify the suspicious flows. Flows classified as suspicious are passed to the third stage i.e., correlator stage. In the correlator stage the flows are clustered into group of flows with similar characteristics. This stage utilizes newly developed multi-dimensional flow correlation [72]. The correlated flows are then passed to the fourth stage that implements topological analysis using graph theory to determine flows with a common controller. Finally flows that share a common controller are investigated in order to determine if they belong to a botnet or not.

MLA and features used: Within the second stage, the method implements the classification of flows by applying three different supervised MLAs: C4.5 decision tree, Naive Bayes and Bayesian network classifier [65]. Several flow level features were used: flow start and end time (numeric), flow protocol (categorical), summary of TCP flags (categorical), total number of packets exchanged in flow (numeric), total number of bytes exchanged in flow (numeric), total number

of packets pushed in flow (numeric), flow duration (numeric), maximum congestion window (numeric), whether client or server initiated connection (categorical), average byte per packet for flow (numeric), average bits per second for flow (numeric), average packets per second for flow (numeric), percentage of packets pushed in flow (numeric), percentage of packets in one of eight packet size bins (numeric), variance of packet inter-arrival time (numeric) and variance of bytes per packet for flow (numeric).

Performance evaluation: Similar to the Livadas et al. approach [71], performances of the Strayer et al. approach have been evaluated through evaluation campaign using bot-related traffic generated within fully controlled experiments, as described by Scenario 2. For the testing only one bot code (Kaiten bot) was used, while background traffic was gathered from the campus network. Performances of the used MLA were evaluated by false positive (FPR) and false negative (FNR) rates. Naive Bayes have shown low FNR, but higher FPR, Bayesian Networks technique have shown low FPR, but higher FNR, while C4.5 decision provided relatively low values of both FNR and FPR. The evaluation also showed that training and performances of classifiers was quite sensitive to the used flow attributes, the training set, and the number of flows used for the training. The method targets individual bots within the botnet and second phase of their life cycle by analysing traffic at the flow level. The method is independent from signatures of traffic payload and the authors argue that the method is suitable for on-line detection. The presented approach shares limitations with the authors' previous work [71], [88]. It is only able to detect IRC botnets with centralized topology and it requires external judgement, either by human or machine, in order to alarm the existence of botnet. Additionally, the method can be evaded by evading classifiers, correlators and topology analysis. Classification can be evaded by performing flow perturbation (strong indication), correlators can be evaded by time-based evasion (strong indication) and the topology analysis can be evaded by deploying evasion of cross-host clustering (strong indication).

Gu et al. proposed BotMiner [73] as a novel mining-based approach in 2008. The proposed approach was one of the first to promise C&C communication topology and protocol independent detection and it is often regarded as one of the most prominent detection techniques. The approach is dedicated to the detection of the group activities of botnets by assuming that bots within the same botnet will be characterized by similar malicious activity and similar C&C communication patterns.

The architecture of the BotMiner detection system consists of five main components: A-Plane monitor, C-Plane monitor, A-Plane clustering, C-Plane clustering and Cross-plane correlator. A-Plane and C-Plane monitors are deployed on the edges of the network examining traffic between internal and external networks and employing appropriate pre-processing. The A-Plane monitor analyses the outbound traffic in order to detect the malicious activities of internal devices while the C-Plane monitor is responsible for tracking network traffic flows. Two monitoring components provide the network logs that are then transferred to the appropriate clustering entity. C-plane clustering and A-plane clustering components process the logs generated by the C-plane and A-plane monitors, respectively. The two clustering entities find the clusters of hosts with similar communication and attack traffic patterns. The results of these entities are then sent to the cross-plan correlation entities. The cross-plane correlator combines the results of the A-Plane and the C-Plane clustering and makes the final decision on which hosts are possibly members of the botnet.

MLA and features used: C-plane clustering is implemented as a two-step process. The first step performs the coarse-grained clustering using a simple clustering algorithm. The second step performs clustering in order to generate smaller and more precise clusters. Both steps are realized using X-means clustering algorithm. X-means [89] is an efficient algorithm based on K-means clustering algorithm. Different from K-means, the X means algorithm does not require the user to choose the number K of final clusters in advance. The first step uses eight features: mean and variance of number of flows per hour (numeric), number of packets per flow (numeric), average number of bytes per packet (numeric), average number of bytes per second (numeric). The second step of clustering uses 52 features: 13 quantiles of the each of the features used in previous step. A-plane clustering is also carried through a two-step clustering of activity logs. The first step clusters the whole list of clients by the type of their activity, while the second step further clusters clients according to specific activity features. The A-plane clustering uses relatively weak cluster features, but provides a possibility of using complex features that are more robust against evasion attacks.

Performance evaluation: BotMiner performances have been evaluated within experiments using bot-related traffic generated by all three scenarios, described in Section IV. Traffic traces produced by diverse types of botnets were used: IRC-based (Spybot, Sdbot and Rbot), HTTP-based (Bobax) and P2P botnets (Nugache and Storm). Background traffic was gathered from the campus network. The technique showed high efficiency in detecting different botnets, with the detection rate (TPR) higher than 99% and bounded FPR. The BotMiner implements traffic analysis at the host level and it is designed to target groups of compromised machines within a monitored network, by targeting the second and the third phases of botnet life-cycle. The technique is entirely independent of the C&C protocol, structure, and infection model of botnets. However, BotMiner has several limitations as well. The presented approach is vulnerable to several evasion tactics such as, evading the C-Plane monitoring, the A-plane monitoring and the cross-plane correlation entity. C- plane monitoring can be evaded by flow perturbation (strong indication). A-Plane can be evaded by performing only subset of attacks (strong indication), by targeting the hosts within the local network (week indication). Finally cross-plane correlation analysis can be evaded by time-based evasion (strong indication) and evasion of cross-clustering (strong indication).

Husna et al. [74] introduced a detection approach based on analysis of behaviour of spammers in 2008. The approach assumes that the majority of spammers are bots and that these compromised hosts can be detected based on the patterns of individual and group behaviour of hosts within the botnets. The method classifies a spammers behaviour based on the features contained in the header of e-mail messages. The method is independent from the content of the message itself.

MLA and features used: The proposed system is realized

through two phases of functioning. The first phase performs selection of features using PCA (Principal Component Analysis) [90] method. The PCA extracts features with high impact to the given dataset. As a result four host level features were selected: active time (numeric), content length (numeric), frequency (numeric) and time of arrival (numeric). The second phase of the approach performs clustering of host machines by employing one of the two clustering algorithms i.e., K-means or hierarchical clustering [67]. The clustering algorithms use previously extracted features as an input.

Performance evaluation: Performances of the Husna et al. approach have been evaluated within experiments using labelled dataset containing only spam e-mails as well as dataset containing both spam and non-spam e-mails. Performances of the used the method were expressed by precision of detection, number of true positives (TP) false positives (FP) and false negatives (FN). Although the majority of the e-mails were clustered correctly, K-means have showed a slightly advantage over the hierarchical clustering. For instance K-means was able to cluster e-mails with a precision of over 90% of, while hierarchical clustering had precision of over 77%. The method analyses traffic on host level targeting individual spamming botnets in the third phase of their life cycle. The technique is independent of the C&C protocol, structure, and infection model of botnets. However the approach also has a several disadvantages. First, it only targets spamming botnets leaving a broad range of botnet types uncovered. Additionally, the method is liable on several evasion techniques such as: evasion by traffic encryption (strong indication), time-based evasion (strong indication), evasion by restricting number of targets (strong indication).

Noh et al. (2009) proposed detection method specially developed in order to detect P2P botnets [75]. The method assumes that P2P bots are characterized by traffic flows that fallow distinctive activity pattern i.e., that P2P bots generate flows that have similarity patterns which can occur at irregular intervals. Using the observed P2P flow patterns, the authors develop a new detection framework that groups flows by similar behaviours and constructs a transition model for each of them. The authors argue that the bots could be successfully detected by comparing transition model for regular P2P traffic and botnet P2P traffic.

The proposed approach consists of four stages: Flow Grouping, Flow Compression, Flow Modelling and Detection stage. The Flow Grouping employs MLA in order to realizes clustering of monitored TCP and UDP connections. The second stage, known as Flow Compression stage computes the state value of each clustered flow and extracts the transition information. The state of the flow is defined by the value of seven flow level features. The third stage uses the information about transition of state value to construct a transitions matrix for flow modelling. The technique used to generate transition matrix is developed by the author and it is based on a Markov chain framework. Finally, the Detection Engine uses the likelihood ratio computed from the probability-based models in order to alert the existence of P2P botnets.

MLA and features used: The flow grouping stage implements clustering of flows. The clustering is realized through several stages. Monitored traffic is first pre-processed by segmenting the traffic on TCP and UDP flows, eliminating meaningless TCP packets and packets originating from retransmission generated due to flooding attacks. Segmented flows are then examined, and for every flow seven categorical features are extracted: indicators of UDP or TCP protocol, source and destination ports, number of connections, connection interaction indicator, packet count comparison, traffic volume comparison. Using the extracted feature vectors, flows are clustered using ROCK algorithm [91] that provides efficient clustering of categorical attributes. This MLA applies clustering by soft links, allowing flows to link to the cluster if they are linked to at least one of its flows.

Performance evaluation: The method has been evaluated using traffic traces of non-malicious and malicious P2P traffic. However it is not known how the author obtained the traces. Traces of three P2P botnets were used: SpamThru, Storm and Nugache botnets. The authors reported high TPR (over 95%) and low FPR (under 2.88%). The proposed system analyses traffic on flow level and targets individual bots operating in the second and third phases of botnet life-cycle. The approach is independent of packet content, and it is capable of detecting botnets using encrypted and obfuscated communication protocols. However the the approach is developed exclusively for detection of P2P botnets limiting its generality. The presented approach can be evaded by several evasion tactics such as: time-based evasion (strong indication), flow perturbation (strong indication), evasion by performing subset of available attacks (weak indication) and evasion by restricting number of targets (weak indication).

Nogueira et al. (2010) [76], [92] proposed a botnet detection approach that provides identification of botnet traffic using Artificial Neural Networks (ANNs) as classification algorithm. The novelty of the method is its ability to operate in on-line fashion and adapt to changes in botnet traffic patterns. The adaptability of the method is provided by adding one additional entity to the general framework of using supervised MLA for botnet detection (Figure 3a). The added entity is Intrusion Management System (IMS), that uses traffic examined by the detection system as well as results of the classification in order to make a conclusion about presence of malicious traffic. If the malicious traffic is detected by IMS and not by the trained neural network the process of retraining the ANN using novel observation is initiated. However the decision process performed by IMS is not automated. In order to to validate a decision made by the classifier the IMS requires external judgement (by human or automatic intervention system).

MLA and features used: Although considering several ANN models, the proposed approach implements identification tests based on a feed-forward back propagation network with three layers [66]. The input layer has $h + 1$ neurons, where h corresponds to the number of previous samples that are presented at the input together with the current sample, that is, the extent of temporal correlation that is considered. The number of neurons in the hidden layer is empirically selected such that the performance function (in this case, the mean square error) is minimized. The output layer has 1 neuron, since each output vector represents the existence of malicious or non-malicious traffic. Unfortunately authors did not provide more details on features used by the ANN so the more detailed analysis of the approach is not possible.

Performance evaluation: Performances of the method have

TABLE II. MACHINE LEARNING FOR BOTNET DETECTION - THE DETAILS ON USED MLAS

Detection Method	**S**upervised or **U**nsupervised	MLAs used	Number of features
Livadas et al. [71]	S	Comparison of three MLAs: C4.5 Tree, Naive Bayes and Bayesian Network classifiers	10
Strayer et al. [72]	S	Comparison of three MLAs: C4.5 Tree, Naive Bayes and Bayesian Network classifiers	16
G.Gu et al. [73]	U	Two level clustering by: First level: X-means clustering Second level: X-means clustering	 Level #1: 8 Level #2: 52
Husna et al. [74]	U	Comparison of two MLAs: K-means clustering and Hierarchical clustering	4
Noh et al. [75]	U	ROCK clustering algorithm	7
Nogueira et al. [76]	S	ANN (Artificial Neural Networks) in adaptive setup	NA
Liu et al. [77]	U	K-means clustering	NA
Liao et al. [78]	S	Comparison of three MLAs: C4.5 Tree, Naive Bayes and Bayesian Network classifiers	12
Yu et al. [79]	U	K-means in adaptive setup	4
Langin et al. [80]	S	SOM (Self Organizing Map)	8
H.Choi et al. [81]	U	X-means clustering	13
Sanchez et al. [82]	S	SVM (Support Vector Machine)	53
Chen et al. [83]	S	Least square SVM (LS-SVM)	4
Saad et al. [84]	S	Comparison of five MLA: SVM, ANN, Nearest Neighbours, Gaussian, and Naive Bayes classifiers	11
Zhang et al. [85]	U	Two level clustering: First level: BIRCH algorithm Second level: Hierarchical clustering	 Level #1: 4 Level #2: 2
W.Lu et al. [86]	U	Comparison of three MLAs: K-means, Un-merged X-means, Merged X-means clustering	256
Bilge et al. [87]	S	Comparison of three MLAs: C4.5 Tree, SVM, and Random forest classifiers	15
Masud et al. [45]	S	Comparison of five MLAs: SVM, C4.5 Tree, Naive Bayes, Bayes Network, and Boosted decision tree classifiers	20
Shin et al. [44]	S	Correlation of the findings of two MLAs: MLA #1: SVM MLA #2: One Class SVM (OCSVM)	 MLA #1: 7 MLA #2: NA
Zeng et al. [46]	U,S	Correlation of the findings of two MLAs: On network level: Hierarchical clustering On client level: SVM	 17 9

been evaluated within a controlled experiment using traffic traces of different malicious and non-malicious applications. The authors obtained malicious traffic traces similarly to the Scenario 2, where authors programmed the application performing various malicious activities using Sub Seven rootkit. The non-malicious "background" traffic is generated by various legitimate applications running on several local machines. On the mixed traffic traces the approach showed high detection rate (TPR over 87.56%) for both malicious and non-malicious traffic. The method analyses traffic on flow level and it is devoted to the detection of individual bots operating in the the second and third phases of botnet life-cycle. The method has several advantages such as: independence from topology and deployed communication protocol, ability to detect encrypted and obscured protocols, and low computational overhead. However the approach has several drawbacks such as need for external judgement in order to provide adaptive functioning. Additionally the approach is is vulnerable on flow perturbation as a evasion strategy (strong indication) and time based evasion (strong indication). It should be noted that due to the lack of details about features chosen for the classification, a final conclusion about evade-ability of approach cannot be made at this point.

Liu et al. (2010) proposed a P2P botnet detection framework based on unsupervised ML techniques and feature analysis of network streams [77]. The proposed framework is realized through three phases: detection of P2P nodes, clustering of P2P nodes and detection of botnet activity. All three phases imply certain assumptions regarding behaviour of both P2P hosts and P2P bots. The first step could be seen as a preprocessing step that analyses network streams in order to detect

P2P nodes within the network. The analysis is conducted by assuming that the P2P bots are characterized by a high degree of paroxysm and distribution. The degree of of paroxysm is determined by examining the number of connections over time while degree of distribution is determined by examining the ratio of connections to different subsets a node is connected to. The second stage clusters the nodes selected by the first phase on several groups of P2P nodes. Finally, the third phase of the approach finds similarities between suspicious actions that P2P nodes implement in order to estimate if P2P node is a member of P2P botnet or not. The authors assume that bots within the same botnet will show similarity in malicious actions.

MLA and features used: The second stage clusters the nodes by using K-means clustering algorithm [67]. The authors assume that the nodes from the same P2P network show steady communication for a long time period and high symmetry of communication, so the nodes are clustered using the following features of P2P traffic: quantity and frequency of data exchange between nodes and symmetry of flows between pair of nodes. However the authors did not indicate exact features used by the method.

Performance evaluation: The method has been evaluated through series of experiments within a controlled network environment, similar to the Scenario 3. The tests were carried on network traces containing flows from different common, non-malicious P2P applications as well as flows from several well-known P2P botnets (Storm, Slapper, Nugache). The approach has shown that efficient clustering of different P2P networks can be achieved within clustering time of 24 hours. Under these conditions proposed approach has shown the capability of identifying different P2P botnet traffic within the overall network traces. The method is performing traffic analysis on the host level. Furthermore, the framework targets the group activity of the bots within the botnet addressing both second and third phase of botnet life-cycle. The authors also argue that the approach is independent of botnet communication protocol and packet content, and that is capable of detecting polymorphic, undiscovered and cryptographic channel botnets. However the proposed approach is vulnerable to a number of evasion techniques: flow perturbation (strong indication), time-based evasion (strong indication) evasion of cross-clustering (strong indication) and evasion by performing only subset of attacks (strong indication).

Liao et al. (2010) [78] explored possibilities of identifying P2P botnet traffic using supervised MLAs. The authors followed scientific efforts of Livadas et al. [71] and Strayer et al. [72], [88], by employing the same MLA. However in contrast to the previous work, the authors explored the possibility of deploying these MLAs for the detection of P2P botnets.

Similarly to the general framework of using supervised MLA for identifying botnet detection, the approach can be observed as a two phase process (Figure 3a). Within the first phase data is pre-processed by extracting features of traffic flows. Several features are extracted such as: indicator of synchronous sessions (categorical), average number of bytes per flow (numeric), average length of packets in flow (numeric), standard deviation of number of bytes per flow (numeric), standard deviation of number of packets in flow (numeric), number of small sessions (numeric), percentage of small packets (numeric), average size of packet (numeric), standard deviation of packet size (numeric), average number of packets per flow (numeric), percentage of small packets per flow (numeric), number of small packets (numeric) and number of null packets (numeric). Where small packets have size between 63-399 bytes, while the sessions with flows and packets within one standard deviation are called "small session".

MLA and features used: Within the second phase the method implement three supervised MLAs: C4.5 decision tree, Naive Bayes and Bayes Network classifiers [65]. The performances of these MLAs in classifying botnet traffic flows were analysed and compared. The MLAs observe flows as a vectors of previously defined features.

Performance evaluation: For the purpose of testing botnet-related traffic traces were recorded in accordance with Scenario 3. The Storm bot was the only malware sample used within the testing. Non-malicious "background" data was generated by several computers running non-malicious applications. The C4.5 classifier has shown the best overall detection performances providing high accuracy of detection (over 98%) and very low false positive and false negative rates. The other two MLAs performed slightly worse, implying the need for the optimization of exploration parameters. The proposed method analyses traffic on flow level, targeting individual bots operating in the second and third phases of botnet life cycle. The method promises detection of botnets that is independent from packet content. However the presented approach has a limited scope as it is only targeting P2P botnets and it is generally vulnerable on evasion by flow perturbation (strong indication).

Yu et al. [79] proposed a novel on-line botnet detection approach in 2010. The newly introduced method promises on-line botnet detection by performing data-adaptive clustering of traffic flows. The method assumes that traffic flows from bots in the same botnet will exhibit high similarities between each-other. As the main goal of the approach is to provide timely detection through on-line operation the method operates in sliding windows. This way the method retains continuous network traffic and faithfully captures dynamic nature of botnet traffic.

MLA and features used: Within every sliding window the system performs the following procedure. First, the pre-processing stage extracts multi-dimensional feature streams from network traffic flows. The feature stream represents a multi-dimensional streaming time series that uniformly quantifies each traffic flow using the set of characteristics. Each feature stream i.e. vector of features, covers a certain time period, while each element in the feature vector is a value that describes the certain characteristic of the flow. The features used within the multi-dimensional vector are: number of bytes-per-packet for flow (numeric), average bits-per-second for flow (numeric), average packets-per-second for flow (numeric) and number of packet-per-flow (numeric). In order to cluster flows based on the formed feature streams the method employs a novel data-adaptive clustering. Initially, the method employs a classical clustering method, i.e., K-means [67], to divide all the feature streams into several clusters. For each cluster, a central feature stream is chosen to represent it. After forming

the clusters, the similarity between feature streams of the same cluster is computed. In contrast to the common implementation of the K-means clustering that entails distance measurement as a similarity measure, the proposed system uses correlation analysis of feature streams. If some cluster has a similarity measure higher then a predefined threshold it will be treated as a suspected cluster. Consequently, all hosts corresponding to the flows in the suspicious cluster will be regarded as suspected bot hosts. When the window slides, the network traffic is pre-processed once again and data-adaptive clustering refreshes the clusters. The clusters are only updated in two situations. First, if some feature streams become invalid in case of connection breaking, and second if the content of some feature streams change. For each of the two update scenarios special updating operations are developed. In the first scenario invalid feature streams are deleted and new ones are added, while in the second all existing clusters are re-constructed by efficient split and merge operations. This kind of data-adaptive clustering provides a great gain in terms of computational efficiency execution time comparing to the more conventional periodical re-clustering.

Performance evaluation: For the purpose of testing botnet traffic is obtained in accordance with the Scenario 2, where authors used Rbot botnet source code to set up a fully controllable botnet and record the "malicious" traffic. Non-malicious i.e. "background" traffic is recorded on a laboratory network originating from diverse types of legitimate applications. Using the test data set the approach expressed very high detection rate (100%), but also it exhibited high FPR (around 20%). The execution time of detection algorithm proved to be appropriate for on-line detection (in the manner of hours). The proposed detection systems is based on flow-based traffic analysis targeting individual bots that operate in the second and third phases of botnet life cycle. The method does not depend on content signatures and it is independent of C&C protocol and network topology. However the method also has its limitations, as it is vulnerable on evasion by flow perturbation (strong indication) due to flow-based analysis, and time-based evasion (strong indication) due to operation in sliding window.

Langin et al. (2010) proposed a P2P botnet detection framework that utilizes Self-Organizing Map (SOM) as a mean of clustering firewall logs data in order to discover unknown P2P bot and other network security issues [80]. The detection framework is based on the assumption that a properly configured firewall should deny access for P2P traffic originating from outside of the local network and log such a traffic for further inspection. The detection approach uses the firewall logs of the denied incoming traffic in order to determine if they are produced by a botmaster that is trying to connect to the potential bot within the local network. This way the proposed approach is able to detect the existence of bot even before it becomes involved in attack.

MLA and features used: The method uses Self-Organizing Map (SOM) [93] as a MLA for learning about existence of bots within the network. Firewall logs of denied incoming traffic are pre-processed in order to extract vector of features for local destination addresses. For every destination IP address following features are extracted: total number of firewall logs (numeric), total number of unique external IP address in the log entries (numeric), total number of unique destination ports in the log entries (numeric), the lowest destination port (numeric), the highest destination port (numeric), total number of ICMP protocol log entries (numeric) and total number of TCP protocol log entries (numeric) and total number of UDP protocol log entries (numeric). Formed feature-vectors are then forwarded to the mining entity. The mining entity uses SOM to determine if there is a suspicious host within the local network. SOM is implemented through two steps namely clustering and classification steps. Within the first clustering phase the SOM is trained. Training is realized by clustering the labelled data and marking the cluster that contain bot-related samples. The second stage clusters unlabelled, previously unseen data. Data clustered to suspicious clusters will be marked as suspicious as well, and the corresponding local host will be considered a potential bot.

Performance evaluation: The method was trained using firewall logs of incoming traffic that was denied at the point of campus network firewall. The test dataset contained 20 million firewall logs, for approximately 60000 local IP addresses. Additionally it is known that there were two P2P bot hosts operating within the campus network. After the training the methods was tested by the logs obtained during the 96 days at the point of campus network firewall. Using the described testing procedure the presented method showed its accuracy by identifying 18 suspect hosts, where some of them were infected with bots while others were improperly configured. The presented approach targets the individual bots covering both second of botnet life-cycle. The presented detection approach discovers knowledge of intrusions and other malignant network problems that is not available by other methods and it preserves privacy of user data by the use of non-local network data and the abstraction of the data in the SOM. However the method also has several disadvantages. First, the resources required for the self-training of the SOM and results interpretation is hard to obtain. Second, as trained on data specific to local network the SOM is not transferable to other locations.

Choi et al. (2011) developed a light-weight botnet detection approach based on analysis of botnet group activity within DNS traffic [81], [94]. The proposed method, referred to as BotGAD (Botnet Group Activity Detector) uses a small amount of data from DNS traffic to detect botnets. BotGAD is relying on several observations regarding the use of DNS services by botnets. First, when a bot is trying to lookup a C&C server or update server and second, when a bot performs DNS lookup of the victim. The authors assume that the bots within the botnet perform the DNS queries in a coordinated fashion, exhibiting group activity that generally appears intensively having a periodic and sporadic pattern. Additionally the authors assume that the botnets have relatively stable group activity i.e., that the botnet groups are consistent.

BotGAD is realized through five main parts: data collector, data mapper, correlated domain extractor, matrix generator, and similarity analyzer. The data collector receives and aggregates DNS traffics from the sensors. In order to provide better precision of botnet detection sensors are deployed throughout the whole monitored network. The data mapper then parses gathered DNS traffic and inserts DNS information into the hash map data structure. The hash map data structure consists of domain maps where every domain map has a domain name as a key and an IP map as a value. IP maps have an IP

address number as a key and the information list as a value. The information list has timestamps of each DNS query and DNS based features as a value. Information mapped this way is then transmitted to the matrix generator and correlated domain extractor. The matrix generator builds a matrix in order to measure a similarity score host contacting certain domains. Binary similarity matrix (dimension m by n) is defined for every domain, and it holds indications if certain IP addresses (one of m) queries the domain in some of observed time slots (n time slots). The correlated domain extractor classifies domain sets using the DNS based features stored in the hash maps. The similarity analyser is the final component of the BotGAD detection system, that calculates the similarity score of generated matrices. It also performs a hypothesis test to make a decision to detect botnet domains. Consequently the detected botnet domains are then summarized in a database.

MLA and features used: The correlated domain extractor employs X-means clustering [89] algorithm in order to detect correlated domains. The authors suggest 13 different features to be used for clustering. The features can be classified into three classes, depending on the aspect they describe: DNS lexicology, DNS query information and DNS answer features. DNS lexicology features are: number of domain tokens (numeric), average length of domain token (numeric) and blacklisted 2nd level domains (categorical). DNS query features are: number of queries sent (numeric), number of distinct sender IPs (numeric), number of distinct senders autonomous system numbers (ASNs) (numeric), query type (categorical) and estimated similarity of a domain (numeric). Finally DNS answer features are: number of distinct resolved IPs (numeric), number of distinct ASNs of resolved IPs (numeric), number of distinct countries of resolved IPs (numeric) and TTL value in DNS answers (numeric).

Performance evaluation: Performances of the method have been evaluated using three DNS traces originating from campus and two ISP networks. As these traces were on real network the authors used a combination of different approaches to verify and label the traces, such as: blacklist matching, web reputation search, IP address resolution and domain information investigation. Within the experiment the technique showed high detection rate (over 95%), very low FPR (lower than 0.31%) and low FNR (under 4.6%). BotGAD is directed at detection of group activity of botnet by covering the second and the third phases of the botnet life-cycle. The method does not depend on content signature and it is able to detect bots using encrypted protocols. As it uses a relatively small amount of traffic it can provide real-time detection. However the proposed method is vulnerable to various evasion techniques such as: time-based evasion (strong indication), evasion by flow perturbation (weak indication), evasion of cross-clustering (strong indication) and evasion by using out-of-band coordination of bots (strong indication).

Sanchez et al. (2011) [82] proposed the system for detection of spamming botnets by classifying the sender of e-mail as spamming hosts or legitimate mail server (LMS) machines. The main assumption made by the authors is that majority of spamming bots are end-user machines and not the legitimate mail server (LMS). Additionally the authors assume that such end-users would directly send the e-mail to the recipient and not through some LMS. The authors propose a system that deploys supervised MLAs in order to build the model of malicious spamming hosts using a set of features that cannot be easily manipulated by spammers.

MLA and features used: The method uses SVM (Support Vector Machine) [65] as machine learning algorithm for classification of hosts on LMS or bots. The paper focuses on the features of a sending machine that cannot be easily manipulated by a spammer, and are already available at or can be easily obtained by a recipient mail server. In particular, the paper considers two types of features associated with a sending machine: the operating system (OS) and the host name lexical structure of the sending machine. OS features used are indicator if the OS type can be determined (categorical) and indicator of the type of OS deployed at the remote machine (categorical). In addition to the OS features 51 host lexical features were used: number or dots on a host name (numeric), number of dashes on the local name portion of the host name (numeric), if a host name encodes IP as a part of the host name (categorical), if the reverse DNS lookup of the machine IP address resolves to a valid host name (categorical), indicator of some of the 13 common keywords found on the local name portion of a host name (categorical), indicator of some of the 34 common keywords found on the local name portion of a host name by the Internet draft (categorical).

Performance evaluation: The proposed approach has been evaluated using 3 different e-mail traces. One is recorded on a campus network containing only LMS records while the other two were spam traces gathered by spam traps. On the testing dataset the framework showed high detection rate (over 91%), low false positive (under 0.56%) and false negative rate (under 8.19%) for all of the applied MLA. The proposed framework is targeting individual bots covering the third phase of the botnet life cycle. The method is primarily developed for detection of spamming bots severely limiting the applicability of the approach. The proposed method is primarily vulnerable on evasion techniques by changing the naming convention of the host names, and by relying on legitimate mail servers for delivering the message.

Chen et al. (2011) [83] were one of the first to tackle problem of on-line adaptive botnet detection. The authors explored the possibility of implementing a detection method that would use MLAs in an adaptive manner and thus be able to adapt to the changes in characteristics of botnet traffic.

The system is composed of the following components: flow parser, graph feature extractor, external IP blacklists, and online classifier. The first two elements implement data preprocessing in the following way. The flow parser reconstructs all the packets that correspond to the same traffic flow, where a flow is defined as the unique 6-tuple of client and server IP-addresses, client- and server-ports, timestamp and transport layer protocol (TCP or UDP). On seeing each new flow, the system updates the aggregated statistics for the pair of IP-addresses which are communicating with each other. The graph feature extractor then maps the pairs of flows between the client and the server and computes simple aggregated counts for how many flows were exchanged between the IP-pair within a pre-configured time interval. At the end of this time-interval, the following information is passed on to the online learning module: timestamp for the end of the time (numeric) window, client IP-address (categorical), server IP-

TABLE III. DETAILS ON PERFORMANCE EVALUATION THE ANALYSED METHODS

Detection Method	Botnet-related data	Background data	Number of bot samples	Performances of the approaches
Livadas et al. [71]	Scenario 2	Campus	1	FPR (10-20%), FNR (30-40%)
Strayer et al. [72]	Scenario 2	Campus	1	FPR (< 30%), FNR (> 2.17%)
G.Gu et al. [73]	Scenarios 1,2,3	Campus	6	TPR (99%), FPR (1%)
Husna et al. [74]	Scenarios 1	NA	NA	Precision (> 88%), TP, FP, FN
Noh et al. [75]	NA	LAN	3	TPR(> 95%), FPR (< 2.88%)
Nogueira et al. [76]	Scenario 2	LAN	1	TPR (> 87.56%)
Liu et al. [77]	Scenario 3	LAN	3	TPR (53-100%)
Liao et al. [78]	Scenario 3	LAN	3	Accuracy (> 92%), FP, FN
Yu et al. [79]	Scenario 2	LAN	1	TPR(100%), FPR(< 20%)
Langin et al. [80]	Scenario 1	LAN	2	TPR (100%), FP
H.Choi et al. [81]	Scenario 1	Campus, ISP	NA	TPR (> 95.4%), FPR (< 0.32%), FNR (< 4.6%)
Sanchez et al. [82]	Scenario 1	Campus	NA	TPR (> 91%), FPR (<0.56%), FNR (< 8.91%)
Chen et al. [83]	Scenario 1	ISP	5	Error rate (< 7%)
Saad et al. [84]	Scenario 1	LAN	2	TPR (> 89%), Error rate (< 20%)
Zhang et al. [85]	Scenario 3	LAN, Campus	2	TPR (100%), FPR (< 0.2%)
W.Lu et al. [86]	Scenario 2	ISP	2	TPR (> 95%)
Bilge et al. [87]	Scenario 1	ISP	NA	TPR (> 87.8%), FPR (< 20.2%)
Masud et al. [45]	Scenario 2	LAN	2	Accuracy(> 95.2%), FPR (< 3.2%) FNR (< 6.5%)
Shin et al. [44]	Scenario 3	LAN	15	TPR (100%), FPR (< 1%)
Zeng et al. [46]	Scenario 2,3	LAN	6	FPR (< 0.16%), FNR (12.5%)

Comment: NA - not available values

address (categorical) and number of flows exchanged between the IP-pair (numeric). The online learning module then trains a classifier using the dataset updated for each incoming IP-pair.

MLA and features used: For the implementation of an adaptive and online botnet detection system the authors turned to the new algorithm known as least-square SVM (LS-SVM) [95]. LS-SVM is incremental in terms of both evolving feature sets as well as training samples. The algorithm uses a predefined number of training samples, N for training the model of malicious traffic. When m new samples are obtained the system excludes a corresponding m number of samples from the training set and uses a new training set in order to form the traffic model. The method considers several methods of excluding the older samples from the training set such as eliminating the oldest samples first or eliminating the samples with least influence of decision surface.

Performance evaluation: The proposed framework has been evaluated by an experiment using two labelled data sets obtained in accordance with the Scenario 1. The labelling was realized using the external IP blacklists. Based on the performed labelling bot-related traffic within the datasets was originating from several botnets namely: Conficker, Grum, Pong, Pushdo and Sality. Within the experiment the technique showed low error rate (under 7%) and the ability to operate in adaptive and on-line manner. The proposed system was compared with batch SVM that was periodically retrained, and it has shown a clear advantage in computational and time requirement. The proposed approach performs traffic analysis on flow level targeting individual bots by capturing traffic characteristics of both second and third phases of the botnet life-cycle. The approach is independent from the protocol and traffic content. However the presented approach is vulnerable on evasion by time-based evasion (strong indication) and evasion by flow perturbation (strong indication).

Saad et al. (2011) [84] conducted a study on P2P botnet detection using machine learning techniques. The authors explored the ability of commonly used MLAs to meet on-line botnet detection requirements, namely adaptability, novelty of detection and possibility of early detection. The author proposed a detection framework based on the identification of network traffic produced by P2P bots. The framework assumes that botnets have distinctive C&C communication patterns which can be used for efficient detection of bots existence within the network even before their attack occurs. Similarly to the general scheme of applying supervised MLA for botnet detection (Figure 3a), the proposed framework is realized through two phases. First phase, implements pre-processing of network traffic by extracting an extensive set of traffic features for each of the flow. Using the extracted set of features, the second phase implements supervised MLAs in order to classify traffic flows as non P2P traffic, P2P C&C flows, and normal P2P traffic flows.

MLA and features used: The features of traffic flows are analysed using five different supervised machine learning techniques. The machine learning techniques that were applied and compared by their performances are: SVM (Support vector Machine), ANN (Artificial Neural Network), Nearest neighbours classifier, Gaussian-based classifier, and Naive

bayes classifier [65]. The features used within the MLAs were: source and destination addresses of flow (categorical), source and destination ports of flow (numeric), transport layer protocol (numeric), payload size in bytes (numeric), average packet length per flow (numeric), total number of packets per flow (numeric), total number of bytes per flow (numeric), total ratio of incoming packets over the number of outgoing packets (numeric), total number of subset of packets of the same length over the total number of packets in the same flow (numeric), and total number of bytes of all the packets over the number of packets in the same flow (numeric).

Performance evaluation: The proposed framework has been evaluated within an experiment using bot-related traffic obtained in accordance with the Scenario 1. Bot-related traffic was originating from two P2P botnets namely Storm and Waledac. Non-malicious "background" traffic was recorded on a LAN network with host computers running different non-malicious applications. Each of the five used MLA were evaluated by training speed, classification-speed and accuracy of detection. The authors conclude that all five techniques provide detection rate grater than 89%. ANN and SVM require the most time to be trained and to perform classification. However, it should be noted that none of the five techniques were able to satisfy on-line detection requirements. Additionally the authors also argue that considering both the training and classification time, the SVM and the ANN are not suitable for online detection. The proposed approach is targeting individual bots by capturing traffic characteristics of the second phase of the botnet life-cycle. The approach is independent from the protocol and traffic content. However the approach is vulnerable on evasion by changing the traffic patterns by flow perturbation (strong indication) and time-based evasion (strong indication).

Zhang et al. (2011) [85] introduced a novel botnet detection system that is able to identify stealthy P2P botnets, even when malicious activities may not be observable. The proposed approach focuses on identifying P2P bots within a monitored network by detecting the C&C communication patterns that characterize P2P botnets, regardless of how they perform malicious activities.

The proposed approach is realized through two main phases. The first phase that identifies P2P hosts and the second that identifies P2P bots among P2P hosts. Identification of the P2P hosts is implemented through pre-processing of the traffic. Traffic pre-processing is realized by flow filtering and extracting of potential P2P flows. The filtering is realized by excluding the network flows whose destination IP addresses were previously resolved in a DNS response. The reason for this kind of filtering is that P2P clients usually contact their peers directly, by looking up IPs from a routing table for the overlay network, rather than resolving a domain name. The remaining traffic is then analysed and a number of statistical features which will be used to isolate flows related to P2P communications from unrelated flows, are extracted. For each host within the monitored network three flow sets are identified: flows related to successful outgoing TCP and UDP connections and flows related to failed outgoing TCP/UDP connections. The approach now assumes that P2P nodes often generate a large number of failed outgoing flows because they periodically probe their peers with ping/pong messages to maintain a view of the overlay network or search for content. Therefore hosts that generate at least one successful outgoing TCP or UDP connection, and more than a predefined number of outgoing failed TCP/UDP connections, are retained for further analysis.

MLA and features used: For every traffic flow four features are extracted: number of packets sent and received (numeric), and number of bytes sent and received (numeric). The generated features vectors are then used by the two-level clustering algorithm in order to cluster the flows. The two-level clustering is realized by combination of BIRCH algorithm [96] and hierarchical clustering [67]. BIRCH algorithm is used to form sub-cluster from the set of flows using Euclidean distance of predefined flow features. Then, for each sub-clusters, flows are aggregated in it and represented by the vector of average values of feature used within the first level of clustering i.e. the vector consisting of average number of packets sent and received per flows in the sub-cluster (numeric), and the average number of bytes sent and received per flows in the sub-cluster (numeric) . Newly formed vector of features is used to cluster the vectors to the clusters of similar vectors. Finally the flows contained in the sub-clusters of one cluster are grouped to the same cluster of flows. For each of generated clusters of flows the set of destination IP addresses related to the flows in the clusters is considered, and for each of this IPs theirs BGP prefix are considered. Finally, clusters with the number of distinct BGP prefixes smaller than some predefined threshold are discarded. Remaining clusters of flows are called fingerprint clusters, and they are used to identify final candidates for P2P hosts. A host is, however, considered as P2P host if its flows belong to at least one of the fingerprint clusters.

In the second phase, the proposed botnet detection system analyses the traffic generated by the candidate P2P clients and classifies them into either legitimate P2P clients or P2P bots. This is based on a number of assumptions. The authors assume that the P2P bots have more persistent functioning on the compromised system, eliminating P2P clients that are running P2P applications with short active time, compared to the underlying system. In order to further discriminate between legitimate persistent P2P clients and P2P bots, the approach uses the following observations. First, bots that belong to the same botnet use the same P2P protocol and network. Second the set of peers contacted by two different bots have a much larger overlap then a set of peers contacted by two P2P clients connected to the same legitimate P2P network.

Performance evaluation: The presented approach has been tested using bot-related traffic traces generated by controlled experiments realized in accordance with Scenario 3. For generation of these traces two malware samples were used: Storm and Waledac. Non-malicious data was gathered from campus network while traces of non- malicious P2P traffic were recorded on a LAN network operating with several hosts running non- malicious P2P applications. Within the controlled experiment the technique showed high efficiency in identifying both malicious and non-malicious P2P hosts, with the TPR of 100% and FPR of 0.2%. The technique also provided the similar detection performances for the scenarios where the host is simultaneously using both malicious and non-malicious P2P traffic. The proposed detection framework analyses traffic on flow level targeting individual bots covering the second

phase of the botnet life-cycle. Furthermore the method does not depend on content signatures or any transport layer heuristics. However the proposed detection approach is vulnerable on several evasion strategies such as: flow perturbation (strong indication), time-based evasion (strong indication), evasion by coordinating bots out of band (strong indication).

W. Lu et al. proposed a novel botnet detection approach based on the unsupervised MLA in 2011 [86]. The method represent continuation on scientific efforts started by the same group of authors a year earlier [97], [98]. Comparing to their previous work, the approach has explored the possibilities of applying a wider range of unsupervised MLAs. The proposed botnet detection framework first filters network traffic produced by known applications and then focuses on identifying botnet communication within the suspicious traffic.

MLA and features used: The proposed detection framework is realized by three phases: Feature Analysis, Clustering and Botnet Decision. In the first phase the method extracts the temporal-frequent attributes of network flows. The extracted features are 256-dimensional vectors that represent the temporal-frequent characteristics of the 256 ASCII binary bytes on the flow payload over a predefined time intervals. The vectors are generated using n-gram (i.e. *n=1*) of bytes distribution over one second intervals. These feature vectors are used in order to cluster corresponding flows within the Clustering phase. The Clustering phase implements three different clustering algorithms thus giving an insight on their performances in clustering botnet traffic. The clustering techniques used within the framework are K-means clustering [67] with fixed number of clusters for botnet detection, and two versions of X-means clustering [89]. The clusters formed by the clustering phase are then forwarded to the Botnet decision phase that marks the cluster with the lowest standard deviation as the botnet cluster.

Performance evaluation: The proposed approach has been evaluated using bot-related traces harvested from both Scenario 1 and Scenario 2. The botnet traffic traces were produced by two IRC-based botnets (Kaiten and one undefined). The non-malicious trace is gathered from a public ISP network. Performance evaluation can be divided into two stages: the performance testing of unknown traffic classification and the performance testing of the approach ability to discriminate malicious botnet traffic. The method exhibited high detection rate in identifying botnet traffic (over 95%), where different clustering techniques provide different detection performances. The merged X-means performed better than the other two approaches since it had a considerably higher detection rate than un-merged X-means and much lower FPR than K-means. The proposed framework is targeting individual bots covering both second and third phases of botnet life-cycles. The method is primarily developed for detection of IRC-based bots but authors argue that it can easily be extended to other types of botnets as well. The presented approach is liable on several evasion techniques such as: evasion by traffic encryption (strong indication), time-based evasion (strong indication) and flow perturbation (strong indication).

Bilge et al. (2012) [87] proposed Disclosure, a large-scale, wide-area botnet detection system that incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data for botnet detection. The authors argue that widely available NetFlow data can be successfully used in order to provide on-line botnet detection on large-scale networks.

The proposed method is realized using supervised learning and therefore implemented in accordance with a general scheme of using supervised MLA for botnet detection (Figure IIIa). The Netflow data is initially pre-processed by separating clients from servers, on the basis of the number of service ports opened by the servers. Then for the selected servers the number of features are extracted. The features are calculated using the NetFlow data that captures the traffic of certain servers. The features can be grouped in three classes: flow size features, client access pattern features, and temporal behaviour features. Flow size features include several features that are calculated based on the number of bytes going to/from the server: mean and standard deviation of number of bytes within a predefined time window (numeric), correlation of feature vector of flows in consecutive time windows (numeric), flow size within the time window (numeric). Client access pattern features are minimum, maximum, median and standard deviation of flow inter-arrival times in the predefined time window (all numeric). Additionally as a client access pattern features a number of unmatched flow for a particular server within the predefined time window is observed (numeric). Finally the system extracts a number of temporal features that characterize the variability of client flow volume as a function of time. Therefore temporal behaviour features are statistical features of the number of flows opened per specific server in predefined time interval.

MLA and features used: Using the extracted features the supervised MLA is trained. To build detection models for identifying C&C servers, the authors experimented with a number of machine learning algorithms, including the C4.5 decision tree classifier, SVM (Support Vector Machines), and Random forest algorithms [65]. However the random forest algorithm provided the best results in terms of on-line operation and precision of detection. Therefore it is used as a main MLA within the method .

Performance evaluation: The proposed approach has been evaluated using labelled traffic traces harvested in accordance with Scenario 1. The traces obtained were originating from both campus and ISP networks. Performance evaluation was realized by experimenting with different sets of features and it showed that the method can provide "reasonable" compromise between detection rate (TPR) and FPR. As an illustration the system is generally able to provide detection rates over 87% for the FPR under 20%. The proposed framework is targeting C&C servers by trying to detect the bots activity in the second phase of the botnet life-cycle. The method is independent from the C&C protocol, and packet content of the network traffic. Additionally, the authors argue that the method is able to provide on-line operation. Base on the presented the system is vulnerable on several evasion techniques such as: time-based evasion (strong indication), evasion by flow perturbation (strong indication), evasion by coordinating bots out of band (strong indication).

B. Client-based detection methods

Within this chapter two client-based detection systems are described. These systems strongly rely on network traffic

TABLE IV. THE OVERVIEW OF EVASION TACTICS FOR PARTICULAR DETECTION APPROACH

Detection Method	Evasion tactics							
	ET1	ET2	ET3	ET4	ET5	ET6	ET7	ET8
Livadas et al. [71]	-	-	-	SF	-	-	-	-
Strayer et al. [72]	-	-	SF	SF	-	-	SF	-
G.Gu et al. [73]	-	-	SF	SF	SF	WF	SF	-
Husna et al. [74]	-	SF	SF	-	SF	-	-	-
Noh et al. [75]	-	-	SF	SF	WF	WF	-	-
Nogueira et al. [76]	-	-	SF	SF	-	-	-	-
Liu et al. [77]	-	-	SF	SF	SF	-	SF	-
Liao et al. [78]	-	-	-	SF	-	-	-	
Yu et al. [79]	-	-	SF	SF	-	-	-	
Langin et al. [80]	-	-	-	-	-	-	-	-
H.Choi et al. [81]	-	-	SF	WF	-	-	SF	SF
Sanchez et al. [82]	-	-	-	-	-	-	-	-
Chen et al. [83]	-	-	SF	SF	-	-	-	-
Saad et al. [84]	-	-	SF	SF	-	-	-	-
Zhang et al. [85]	-	-	SF	SF	-	-	-	SF
W.Lu et al. [86]	-	SF	SF	SF	-	-	-	-
Bilge et al. [87]	-	-	SF	SF	-	-	-	SF
Masud et al. [45]	SF	SF	SF	SF	-	-	-	-
Shin et al. [44]	SF	-	SF	SF	WF	-	-	WF
Zeng et al. [46]	SF	-	SF	SF	-	WF	SF	-

Comment: SF - strong factor, WF - weak factor

analysis for the detection of bot presence at the point of compromised computer.

Masud et al. (2008) [45] proposed a client-based detection method that analyse both traffic produced by host computer and application execution at host computer. The method is based on correlating multiple log files and applying MLAs for detection of C&C traffic. The main assumption behind the proposed approach is that bots respond much faster than humans and that command latency is therefore much lower for C&C communication than in the case of non malicious applications.

The proposed detection is implemented in accordance with the general scenario of using supervised machine learning for botnet detection. The system monitors two types of bot-related forensics: network traffic traces and application execution logs. The network traffic traces include all network packets that are sent or received by the host, while application execution logs contain the start time of program executions on the host machine. Both network and application traces are correlated and pre-processed by the pre-processing entity. As a result of pre-processing several features of traffic flows are extracted. Using the extracted features supervised MLA learns the model of the malicious flows. The trained model is then used to classify novel flows as malicious or not-malicious.

MLA and features used: The approach implements five different classifiers: SVM (Support Vector Machine), Bayes network classifier, C4,5 decision tree, Naive Bayes and Boosted decision tree [65]. Different MLA are compared by their performances of classifying traffic flows as bot-related flows or not bot-related flows. The MLA algorithms use a number of features extracted by correlation of two log files. The features can be divided into two groups: Packet-based features that are calculated for every packet in the flow and Flow-based features calculated for every flow. Packet-based features are: BR (categorical) - indication of potential incoming command packet i.e. if the time difference between packet coming from and going to certain *IP:port* par is smaller than 100ms, BRtime (numeric) - time difference between BR packet and its corresponding outgoing packet, BRsize (numeric) - length of BR packet, BO (categorical) - indication of packet that originates from *IP:port* pair and that there is a packet going to *IP':port'* pair in a period shorter than 200ms, BoDestMatch (categorical) - indicator that payload of BO packet contains *IP'* address, BOtime (numeric) - time difference between BR packet and its corresponding outgoing packet, BA (categorical) - indicator for an incoming packet there is an application that is started within 3 seconds, BAtime (numeric) - time difference between arrival of BA packet and the launching of corresponding application, and BAMatch (categorical) - indicator that in the payload of BA packet contains name of corresponding application. The flow based features are: average and variance of length of the packets in kB, number of BA packets as percentage of total number of packets, average and variance of BAtime of all BA packets, number of BR packets as percentage of total packets, average and variance of BRtime of all BR packets, average and variance of BRsize of all BR packets, number of BO packets as percentage of total packets, and average and variance of BOtime of all BO packets. All flow-based features are numeric in nature.

Performance evaluation: The system uses data obtained from a controlled network environment as in Scenario 2. The

system is first trained with log files obtained from the infected host, and tested using the logs obtained from both infected and uninfected machines. For the purpose of evaluation, only two IRC bots were used: SDBot and RBot, where both training and testing data sets contained traces from both bot samples. On the testing data set the framework showed high accuracy of detection (over 95%), low false positive rate (under 3%) and low false negative rate (under 5%) for all of the applied MLA. However Boosted decision tree showed the best overall performances. The method analyses traffic on flow level by targeting individual bots and the communication phase of botnet life cycle. It does not put any restriction regarding topology and deployed communication protocol. However the method relies on access to payload content, so it could be defeated by encryption of C&C channel. Additionally, the method has not been tested on non-IRC bots so its ability to tackle modern bots has not been proved yet. Finally, the presented method is vulnerable to the various evasion techniques: evasion by traffic encryption (weak indication), evasion by flow perturbation (strong indication) and evasion by evading the client based detection (weak indication).

Shin et al. (2012) proposed EFFORT [44], a cooperative host-based detection framework, based on intrinsic characteristics of bots from both client and network levels. The proposed framework is implemented as a multi-module approach that correlates bot-related indications from different client and network level aspects, in order to provide more effective and efficient detection. The overall architecture of EFFORT consists of five modules (namely M1, M2,M3,M4, and M5). Module M1 provides an initial gathering of the observation that are then used in the modules M2, M3 and M4 in order to generate individual assessment of existence of bots existence. The observations are then forwarded to the module M5 that correlates them giving the final decision if the processes are malicious or not.

Module M1 finds bot-driven processes as well as bot-related network occurrences within the host computer. This module captures outgoing network connections produced by processes running as well as human-process interaction. Outgoing traffic is captured by observing DNS queries while human-process interaction is captured by recording system calls related to the keyboard and mouse actions. The two information are then combined creating a model that describes what network activities are characteristic for the observed process. The model employs three metrics: the time difference between the time when a process issues a DNS query and the time when a process produces a mouse or keyboard event, the source of the events, and the indication whether a process is running in the foreground at the time. The process event is regarded as generated from human, if the time difference is very small, the event is from actual physical devices, and the process is running in the foreground. Indications obtained by Module M1 are forwarded to the modules M2, M3 and M4 that perform detection of malicious processes. Module M2 classifies DNS traffic produced by processes as malicious or non-malicious. Module M3 classifies malicious behaviour of processes within the host machine. Module M4 is monitoring the traffic generated by the specific processes on the hosts network interface. The module employs a simple method of comparing incoming packets and bytes exchange rate between process and remote site. If the exchange ratio is smaller than some predefined threshold bot behaviour is indicated. After each module (Module M2-M4) makes its own decision, the correlation engine (Module M5) will combine these results and make a final decision using a weighted voting scheme.

MLA and features used: Module M2 employs supervised MLA in order to classify queried domain names as malicious or benign. SVM (Support Vector Machine) [65] is used, while several features are considered for the classification. Based on the domain black lists and whois service five features are considered: time difference between current date and domain expiration date - numeric value (most malicious domains are registered very recently), time difference between domain - numeric value (most malicious domains have short life time), number of domain registration - numeric value (malicious domains are typically registered to few name servers), and indicator whether a target domain can be found on blacklist or not - categorical value (malicious domains are likely on blacklist). Based on the search engines two additional features are used: indicator if the domain names are contacted by the process well indexed (thus returning many results) - categorical value, and indicator are the domain queried by the process frequently used in a malicious context in the top returned web pages - categorical value. Module M3 also uses supervised learning to classify malicious behaviour within the host computers. This module detects the bot malware presence by examining the usage of host computer resources. The module models benign processes using several heuristics such as: typically normal processes rarely access files in other user's folders and system directories, typically normal processes do not modify critical registries, and typically normal processes do not create a large number of sockets in a short time period. For modelling of benign behaviour of processes One Class SVM (OCSVM) [65] was used.

Performance evaluation: Performances of the method have been evaluated through a set of experiments where bot-related data was gathered in accordance with Scenario 3, while begin forensic is gathered from several uninfected machines. For testing purposes, 15 different real-world bot binaries were used, covering both centralized and decentralized botnets. The system managed to detect all bots used within experiments producing low FPR (less than 1%). The framework is targeting individual bots covering second and third phase of bot life cycle. Additionally, the method has a number of advantages such as, independence from topology and deployed communication protocol, and ability to detect encrypted and obscured protocols. However the method is vulnerable on several evasion techniques: time-based evasion (strong indication), host-based evasion (strong indication), time-based evasion (strong indication), evasion by flow perturbation (strong indication), evasion by performing a subset of attacks (weak indication) and coordinating bots out of band (weak indication).

C. Hybrid detection methods

Although several groups of authors [47]–[50] have proposed hybrid detection systems, to the best of our knowledge only Zeng et al. [46], [99] have actually implemented it. The system uses MLAs for inferring about bot existence on both network and client level, where client level detection entity heavily rely on traffic analysis.

Zeng et al. (2010) [46], [99] proposed a hybrid detection approach that provides botnet detection by combining client- and network-level information. The approach is based on the assumption that two sources of bot-related information will complement each other in making detection decisions. The framework first identifies suspicious hosts by discovering similar behaviours among different hosts using flow analysis, and then validates identified suspects to be malicious or not by scrutinizing their in-host behaviour. The approach is assuming that bots within the same botnet have similar traffic patterns i.e. traffic flows with similar pattern and that this occurrence can be captured by network-based detection. The flows that exhibit similar characteristics are marked as suspicious, and they are referred to as triggering flows. The approach then associates all subsequent flows with each triggering flow on a host-by-host basis, checking the similarity among those associated groups. If multiple hosts behave similarly in the trigger-action patterns, they are grouped into the same suspicious cluster as likely to belong to the same botnet. Whenever a group of hosts is identified as suspicious by the network analysis, the host-behaviour analysis results, based on a history of monitored host behaviours, are reported. A correlation algorithm finally assigns a detection score to each host under inspection by considering both network and host behaviours. The architecture of the proposed system is consisting of three components: host analyser, network analyser, and correlation engine.

MLA and features used: The network analyser consists of two modules: flow analyser and clustering. The flow analyser takes the NetFlow data from a router as input and searches for trigger-action botnet-like flow patterns among different hosts. The clustering is realized by unsupervised learning i.e., hierarchical clustering [67]. For every flow a set of 17 numeric features were used: the mean, variance, skewness and kurtosis of duration of flow, total number of bytes per flow, total number of packets per flow, the number of TCP flows, the number of UDP Flows, the number of SMTP flows, the number of unique IPs contacted and the number of suspicious ports. The feature vector is fed to the clustering module that groups similarly-behaving flows into the same cluster.

MLA and features used: The host analyser is deployed at every host and it consists of two modules: in-host monitor and suspicion-level generator. The in-host monitor is tracking run-time system-wide behaviour taking place in the registry, file system, and network stack on a host. To quantify suspicious level based on the observed host-based features, supervised MLA i.e. SVM (Support Vector Machine) is used. Nine different features are observed for this purpose: the indicator of DLL or EXE creation in system directory (categorical), the indicator of modification of files in system directory (categorical), the indicator of the creation of auto run key in registry (categorical), the indicator of the creation of process injection key in registry (categorical), the indicator of modification of critical registry keys (categorical), the number of ports opened (numeric), the number of suspicious ports (numeric), the number of unique IPs contacted (numeric) and the number of SMTP flows (numeric). Suspicious-level generator generates a suspicion-level by applying a machine learning algorithm based on the behaviour reported at each time window. The host analyser sends the average suspicion-level along with a few network feature statistics to the correlation engine, if required.

Performance evaluation: The system used bot-related data obtained from both Scenarios 2 and 3. For testing, 6 bot samples were used such as: IRC (rbot, spybot), HTTP (BobaxA, BobaxB) and P2P (Storm, Waledac) bots. Non-malicious traffic is recorded from the campus network. On the testing data set the framework showed low FPR (under 0.16%) and relatively low FNR (12.5%). The method generally targets group activities of botnets and it covers both communication and attack phase of botnet life-cycle. The approach is independent from the C&C protocol, topology and content of transmitted data. The main limitation of the approach is that it cannot detect individual hosts and that it operates in time windows. Based on the presented, the method is liable on following evasion techniques: time-based evasion (strong indication), evasion by flow perturbation (strong indication), host based evasion techniques (strong indication), cross-host clustering (strong indication) and restricting number of attack targets (weak indication).

VI. Discussion

Based of the analysis of detection approaches presented in the previous section several challenges of comparing the contemporary detection methods can be identified. First, different approaches assume different botnet traffic heuristics, providing different flexibility and generality of detection. Second, the approaches use different MLAs implemented in diverse ways and various setups. Third, the approaches are tested end evaluated using diverse evaluation datasets. Finally, the methods promise different performances that are expressed using somewhat different performance metrics. All of this make direct comparison of the approaches difficult and sometimes even impossible. Therefore, instead of comparing the methods, this section summarizes the characteristics and the capabilities of existing detection methods outlining the possibilities of using machine learning for identifying the presence of botnets and indicating the space for further improvements and future work.

A. Characteristics of the approaches

As presented in Table I, the majority of detection approaches addressed by this review are purely network-based, thus analysing network traffic recorded at an "edge" of a network. Depending on the point of implementation the majority of the approaches are suitable for operating on LAN networks [44]–[46], [75]–[80], [84], [85], some on campus networks [71], [73], [81], [82], [85], [88] while only a few on ISP networks [81], [86], [87]. The main difference between the points at which the methods are implemented is the exact scope of network that is visible form the point of traffic monitoring. For instance, a detection system implemented at the core network router has more comprehensive outlook on the behaviour of bots within a certain botnet, than the detection system implemented at a gateway connecting a local network to the Internet. By the same token, the client-based techniques addressed in this survey [44], [45] are only able to catch network traffic produced by the individual bots.

The detection methods usually target individual bots or group behaviour of bots within the botnet, while some methods alternatively target malicious C&C servers. The techniques that target the group behaviour of bots [46], [73], [77], [81] or

the C&C servers [87] cannot detect individual bots within the monitored network and they usually require to be implemented in the higher network tiers from which a broader network scope is visible. However, this methods provide more comprehensive insight on dynamics of the botnet behaviour and regularities of botnet traffic.

As the majority of the approaches operate at the LAN network targeting individual compromised hosts, there is a clear need for the development of approaches that can operate on core ISP networks, thus covering the larger network scope and consequently being able to catch activity of higher number of bots. As applying detection systems on high-speed networks requires detection algorithm that is capable of processing large quantities of data in an efficient manner, special emphasis should be place on development of computationally and time efficient detection algorithms.

The analysed detection methods are based on diverse botnet traffic heuristics, consequently having different characteristics regarding generality of operation. The methods target either the communication, the attack or both phases of botnet life-cycle, by modelling characteristics of traffic produced during them. However, none of the techniques covers the infection phase, which makes the machine learning-based techniques somewhat lacking behind some of the existing IDSs [56], [57] and anomaly-based botnet detection systems [55], [59]. Additionally, the techniques that only cover the attacks phase of the botnet life-cycle [74], [82] are dependent on the specific attack campaigns thus detecting only botnets that implement certain malicious activity (in this case botnets sending SPAM messages).

Some of the methods are independent of C&C communication [44], [46], [73], [74], [76], [82], [83], [87], while other methods target specific types of botnets, such as IRC-based [45], [71], [72], [79], [86] and P2P-based [75], [77], [78], [80], [84], [85] botnets by relying on specific heuristics of IRC and P2P C&C channels. Additionally, some of the techniques such [81] detect botnets by malicious use of DNS services, limiting its scope to botnets that rely on DNS for discovering addresses of C&C servers.

The approaches addressed by the survey are generally independent from the payload content, therefore having a significant prevalence over conventional signature based detection methods [55]–[57]. Only three techniques [45], [74], [86] rely on content of the payload thus being easily defeated by the encryption or the obfuscation of the packet payload.

Timely detection by on-line operation is promised by only a couple of approaches. However, as indicated in Section IV, the requirements of timely detection are not precisely defined, and the question of how prompt detection should be is still unanswered. Some of the contemporary detection approaches provide the on-line functioning by operating in a time window and then being periodically trained using the new training set or by periodically updating the clusters of the observation [76], [81], [88]. Others, as [79], [83] use adaptive MLAs in order to provide the adaptive and on-line detection. Finally, some methods such as [87] argue that they are capable on-line detection, based on the fact that they are able to process the data in the shorter time than the actual length of the traffic traces. One of the important goals of future detection systems is to define requirements of timely detection and develop methods that are able to operate in on-line fashion. These systems should also be able to adapt to the changes in network traffic patterns and they should improve their efficiency so they could be implemented in core networks.

B. MLAs used within the approaches

The existing techniques use a variety of machine learning algorithms deployed in diverse setups. In total 16 different MLAs were used within the 20 analysed approaches. As presented in Table II, supervised and unsupervised MLAs are evenly distributed between the methods. Some of the approaches experimented with more than one MLA providing the good insight on how the assumed heuristics hold in different learning scenarios as well as what are the performances of different MLAs [45], [71], [74], [78], [84], [86]–[88]. Additionally, some authors used MLAs in more advanced setups, where clustering of observation is realized through two level clustering schemes [73], [85] or where the findings of independent MLAs were correlated in order to pinpoint the malicious traffic pattern [44], [46], [73]. Several authors used the same MLAs within their detection systems [45], [71], [78], [84], [88] so their direct comparison is possible, to the large extent.

The MLAs analyse traffic on both packet and flow levels using an array of host- and flow-based network traffic features. As indicated in the Section III, the processes of extracting and choosing the right features set are crucial to the performances of the detection system. Therefore, one of the most important goals of the future work is optimization of the chosen feature set so it can capture the targeted botnet traffic heuristics. Additionally, within detection systems that require the processing of high traffic loads significant resources should be dedicated to the extraction of the features that will be used within the MLAs [87]. Therefore, the future work should also be directed at development of time and computationally efficient data pre-processing.

The performances of MLAs differ so some of the supervised MLAs algorithms, such as decision tree classifiers perform very well in terms of time and computational requirements of building the model and classifying new traffic observation. Others, such as ANN and SVM although providing the precise classification are more performance demanded. Performances of data clustering using unsupervised MLAs highly depend on the number of features and the number of clusters used within algorithms. However, the approaches addressed by this paper do not provide the sufficient information on the performances of MLAs and pre-processing of data adding to the difficulties of assessing the scalability of the approaches. Therefore, the future work should be directed at exploring the efficiency of different classification and clustering algorithms on extensive botnet traffic dataset.

C. Performances evaluation

Due to the challenges of obtaining training and testing data, the evaluation of the proposed botnet detection systems is one of the most challenging tasks within the development of detection methods. As it can be seen from the Table III the botnet traffic traces used within the approaches were obtained

both by labelling traffic traces [74], [81], [83], [86], [87] as well as by combining malicious and non malicious traffic traces. Malicious botnet traffic traces were obtained through all three scenarios (as presented in Section IV), while the background data is usually recorded on the campus or LAN networks. However, the use of the background traffic recorded on the LAN or campus network raises concerns regarding the representativeness of such data and the validity of performance evaluation. This is due to the fact that the network traffic produced in the controlled LAN environment do not capture the traffic patterns of "usual" usage of the Internet, while traffic on campus networks is usually filtered so some of the traffic protocols and network services are blocked.

In addition, the malicious traffic samples are usually recorded from a very limited number of bot samples. For instance, the performances of only four detection approaches were evaluated on the traffic traces produced by more than 5 bot sample [44], [46], [73], [83], while the maximal number of samples used for evaluation was 15 in case of [44]. The rest of the methods were tested with less than 4 bot malware samples. The poor number of bot samples can be explained due to many legal, ethical and practical limitations of getting the botnet traffic traces. However, the small number of used bot samples indicates the need for more thorough testing where more comprehensive set of malware samples would be used, in order to prove that the detection system is not only able to model traffic of couple of botnets but also to generalize the inferred knowledge.

In addition to the diverse evaluation practices, the performances of the approaches are also expressed using different performance metrics. As the methods use different evaluation practices and testing datasets, they cannot be directly compared using the presented metrics. However, the presented performance metrics still indicate the overall performances of the particular approach in identifying botnet traffic, showing the great perspective of using machine learning for botnet detection.

As illustrated in Table IV, the proposed approaches are more or less liable on the different evasion techniques. Generally, a majority of the the methods are resistant to evasion by encryption of botnet traffic. Only three approaches [45], [74], [86] that use features of packet payloads are vulnerable on this evasion strategy. However, a majority of techniques are liable on evasion by flow perturbation, due to the fact that they analyse traffic on the flow level. The techniques that operate in the time window and especially the ones that promise the on-line operation are vulnerable on time-based evasion. The analysed client-based [44], [45] and hybrid [46] based techniques are vulnerable on evasion techniques that target the monitoring of the internals of a host computer. As this study does not address the complexities of evasion techniques and its effect on the overall utility of the botnet, the future work could be directed at more thorough analysis of evasion techniques and vulnerability of modern detection systems to them, covering effect of evasion techniques to both detection systems and overall utility of the botnet.

VII. Conclusion

Botnet detection based on machine learning has been the subject of interest of the research community resulting in the numerous detection methods that are based on different botnet heuristics, that target different types of botnets using diverse machine learning algorithms and that consequently provide varying performances of detection. This paper presents a review of some of the most prominent contemporary botnet detection methods that use machine learning as a tool of identifying botnet-related traffic. The presented study addresses 20 detection methods, proposed over the last decade. The methods have been analysed by investigating bot-related heuristic assumed by the detection systems and machine learning techniques used in order to capture botnet-related knowledge. Furthermore, the methods have been examined by analysing their characteristics, performances, and limitations. The analysis of these detection approaches indicates a strong ability of this class of approaches to be used for identifying botnet network traffic. However, the study also indicates several aspects of machine learning-based approaches that could be further improved.

First, the modern machine learning-based detection systems should aim at achieving data-adaptive, on-line and efficient detection. The approach should be able to adapt to the changing patterns of botnet traffic, and it should operate in the on-line fashion in order to provide timely detection and fulfil requirements of prompt botnet neutralization. Finally, the detection methods should be time and computationally efficient so they could be successfully deployed at core networks, covering larger network scopes and providing more thorough insight on traffic produced by botnets.

Second, the comprehensive testing and evaluation of the proposed detection systems is needed, where more comprehensive data sets would be used, covering malicious traffic originating from higher number of botnets and non-malicious traffic that capture "true" nature of the Internet traffic. Additionally, evaluation should not only concentrate on the assessment of the accuracy of detection but also on the assessment of the performances of data pre-processing and the machine learning algorithms, so qualified judgement on detection performances and the scalability of methods could be made.

References

[1] Hogben, G. (ed.), Botnets: Detection, measurement, disinfection and defence, Tech. rep., ENISA (2011).

[2] M. Egele, T. Scholte, E. Kirda, C. Kruegel, A survey on automated dynamic malware-analysis techniques and tools, ACM Comput. Surv. 44 (2) (2008) 6:1–6:42. doi:10.1145/2089125.2089126.

[3] P. R. Marupally, V. Paruchuri, Comparative analysis and evaluation of botnet command and control models, in: Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, AINA '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 82–89. doi:10.1109/AINA.2010.171.

[4] H. Zeidanloo, A. Manaf, Botnet command and control mechanisms, in: Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on, Vol. 1, 2009, pp. 564 –568. doi:10.1109/ICCEE.2009.151.

[5] D. Dittrich, S. Dietrich, P2p as botnet command and control: a deeper insight, in: Proceedings of the 3rd International Conference On Malicious and Unwanted Software (Malware 2008), 2008, pp. 46–63.

[6] M. A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, A multifaceted approach to understanding the botnet phenomenon, in: Internet Measurement Conference, 2006, pp. 41–52. doi:10.1145/1177080.1177086.

[7] M. A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging, in: Proceedings of the first conference on First Workshop on Hot Topics

in Understanding Botnets, HotBots'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 5–5.

[8] S. Liu, J. Gong, W. Yang, A. Jakalan, A survey of botnet size measurement, in: Networking and Distributed Computing (ICNDC), 2011 Second International Conference on, 2011, pp. 36 –40. doi:10.1109/ICNDC.2011.15.

[9] S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, Botnets: A survey, Computer Networks 1 (0) (2012) –. doi:10.1016/j.comnet.2012.07.021.

[10] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, M. Karir, A survey of botnet technology and defenses, in: Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology, 2009, pp. 299 –304. doi:10.1109/CATCH.2009.40.

[11] C. Li, W. Jiang, X. Zou, Botnet: Survey and case study, in: Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on, 2009, pp. 1184 –1187. doi:10.1109/ICICIC.2009.127.

[12] N. Ianelli, A. Hackworth, Botnets as a vehicle for cyber crime, Tech. rep., CERT Coordination Centre (2005).

[13] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, J. Zhang, Botnet: classification, attacks, detection, tracing, and preventive measures, EURASIP J. Wirel. Commun. Netw. 2009 (2009) 9:1–9:11. doi:10.1155/2009/692654.

[14] M. Feily, Shahrestani, A survey of botnet and botnet detection, Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on (2009) 268–273doi:10.1109/SECURWARE.2009.48.

[15] H. Zeidanloo, M. Shooshtari, P. Amoli, M. Safari, M. Zamani, A taxonomy of botnet detection techniques, in: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, Vol. 2, 2010, pp. 158 –162. doi:10.1109/ICCSIT.2010.5563555.

[16] Z. Zhu, G. Lu, Y. Chen, Z. Fu, P. Roberts, K. Han, Botnet research survey, in: Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International, 2008, pp. 967 –972. doi:10.1109/COMPSAC.2008.205.

[17] L. Zhang, S. Yu, D. Wu, P. Watters, A survey on latest botnet attack and defense, in: Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, 2011, pp. 53 –60. doi:10.1109/TrustCom.2011.11.

[18] M. Masud, L. Khan, B. Thuraisingham, Data Mining Tools for Malware Detection, Taylor & Francis Group, 2011.

[19] S. Dua, X. Du, Data mining and machine learning in cybersecurity., Boca Raton, FL: CRC Press. xxii, 234 p. $ 89.95 , 2011. doi:10.1201/b10867.

[20] J. Nazario, T. Holz, As the net churns: Fast-flux botnet observations, in: Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on, 2008, pp. 24 –31. doi:10.1109/MALWARE.2008.4690854.

[21] T. Holz, C. Gorecki, K. Rieck, F. C. Freiling, Measuring and detecting fast-flux service networks, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008, 2008, pp. 8–8.

[22] M. Antonakakis, J. Demar, C. Elisan, J. Jerrim, Dgas and cybercriminals: A case study, Tech. rep., Damballa (2012).

[23] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, D. Dagon, Peer-to-peer botnets: overview and case study, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 1–1.

[24] P. Wang, S. Sparks, C. C. Zou, An advanced hybrid peer-to-peer botnet, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 2–2.

[25] Z. Zhang, B. Lu, P. Liao, C. Liu, X. Cui, A hierarchical hybrid structure for botnet control and command, in: Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on, Vol. 1, 2011, pp. 483 –489. doi:10.1109/CSAE.2011.5953266.

[26] G. Gu, Correlation-based botnet detection in enterprise networks, phd thesis, Ph.D. thesis, Georgia Institute of Technology (July 2008).

[27] P. Maymounkov, D. Mazières, Kademlia: A peer-to-peer information system based on the xor metric, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS '01, Springer-Verlag, London, UK, UK, 2002, pp. 53–65.

[28] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, The bittorrent p2p file-sharing system: Measurements and analysis, in: M. Castro, R. Renesse (Eds.), Peer-to-Peer Systems IV, Vol. 3640 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 205–216.

[29] K. Kutzner, T. Fuhrmann, Measuring large overlay networks âĂŤ the overnet example, in: P. MÃijller, R. Gotzhein, J. Schmitt (Eds.), Kommunikation in Verteilten Systemen (KiVS), Informatik aktuell, Springer Berlin Heidelberg, 2005, pp. 193–204.

[30] G. Sinclair, C. Nunnery, B.-H. Kang, The protocol: The how and why, in: Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on, 2009, pp. 69 –77. doi:10.1109/MALWARE.2009.5403015.

[31] R. Perdisci, I. Corona, D. Dagon, W. Lee, Detecting malicious flux service networks through passive analysis of recursive dns traces, in: Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 311–320. doi:10.1109/ACSAC.2009.36.

[32] Damballa, A new iteration of the tdss/tdl4 malware using dga-based command-and-control, Tech. rep., Damballa (2012).

[33] I. You, K. Yim, Malware obfuscation techniques: A brief survey, in: Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, 2010, pp. 297 –300. doi:10.1109/BWCCA.2010.85.

[34] J. Marpaung, M. Sain, H.-J. Lee, Survey on malware evasion techniques: State of the art and challenges, in: Advanced Communication Technology (ICACT), 2012 14th International Conference on, 2012, pp. 744 –749.

[35] E. Stinson, J. C. Mitchell, Towards systematic evaluation of the evadability of bot/botnet detection methods, in: Proceedings of the 2nd conference on USENIX Workshop on offensive technologies, WOOT'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 5:1–5:9.

[36] B. Blunden, The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System, Jones and Bartlett Publishers, Inc., USA, 2009.

[37] T. Arnold, T. A. Yang, Rootkit attacks and protection: a case study of teaching network security, J. Comput. Sci. Coll. 26 (5) (2011) 122–129.

[38] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, J. Nazario, Automated classification and analysis of internet malware., in: C. KrÃijgel, R. Lippmann, A. Clark (Eds.), RAID, Vol. 4637 of Lecture Notes in Computer Science, Springer, 2007, pp. 178–197.

[39] E. Stinson, J. C. Mitchell, Characterizing bots' remote control behavior, in: W. Lee, C. Wang, D. Dagon (Eds.), Botnet Detection, Vol. 36 of Advances in Information Security, Springer, 2008, pp. 45–64.

[40] L. Liu, S. Chen, G. Yan, Z. Zhang, Bottracer: Execution-based bot-like malware detection, in: Proceedings of the 11th international conference on Information Security, ISC '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 97–113.

[41] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, X. Wang, Effective and efficient malware detection at the end host, in: Proceedings of the 18th conference on USENIX security symposium, SSYM'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 351–366.

[42] U. Bayer, P. M. Comparetti, C. Hlauschek, C. KrÃijgel, E. Kirda, Scalable, behavior-based malware clustering., in: NDSS, The Internet Society, 2009, pp. 5–5.

[43] Y. Park, Q. Zhang, D. Reeves, V. Mulukutla, Antibot: Clustering common semantic patterns for bot detection, in: Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual, 2010, pp. 262 –272. doi:10.1109/COMPSAC.2010.33.

[44] S. Shin, Z. Xu, G. Gu, EFFORT: Efficient and Effective Bot Malware Detection, in: Proceedings of the 31th Annual IEEE Conference on Computer Communications (INFOCOM'12) Mini-Conference, 2012, pp. 71–80.

[45] M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, K. Hamlen, Flow-based identification of botnet traffic by mining multiple log files, in: Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on, 2008, pp. 200 –206. doi:10.1109/ICDFMA.2008.4784437.

[46] Y. Zeng, X. Hu, K. Shin, Detection of botnets using combined host- and network-level information, in: Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on, 2010, pp. 291 –300. doi:10.1109/DSN.2010.5544306.

[47] H. Wang, Z. Gong, Collaboration-based botnet detection architecture, in: Proceedings of the 2009 Second International Conference on Intelligent Computation Technology and Automation - Volume 02, ICICTA '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 375–378. doi:10.1109/ICICTA.2009.326.

[48] H. Wang, J. Hou, Z. Gong, Botnet detection architecture based on heterogeneous multi-sensor information fusion, Journal of Networks 6 (12) (2011) 1655–1661.

[49] M. Szymczyk, Detecting botnets in computer networks using multi-agent technology, in: Proceedings of the 2009 Fourth International Conference on Dependability of Computer Systems, DEPCOS-RELCOMEX '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 192–201. doi:10.1109/DepCoS-RELCOMEX.2009.46.

[50] R. F. Erbacher, A. Cutler, P. Banerjee, J. Marshall, A multi-layered approach to botnet detection., in: H. R. Arabnia, S. Aissi (Eds.), Security and Management, CSREA Press, 2008, pp. 301–308.

[51] A. Karasaridis, B. Rexroad, D. Hoeflin, Wide-scale botnet detection and characterization, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 7–7.

[52] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, N. Feamster, Building a dynamic reputation system for dns, in: Proceedings of the 19th USENIX conference on Security, USENIX Security'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 18–18.

[53] D. Dagon, C. Zou, W. Lee, Modeling botnet propagation using time zones, in: In Proceedings of the 13 th Network and Distributed System Security Symposium NDSS, 2006, pp. 7–7.

[54] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, W. Lee, Active botnet probing to identify obscure command and control channels, in: Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 241–253. doi:10.1109/ACSAC.2009.30.

[55] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, BotHunter: Detecting malware infection through IDS-driven dialog correlation, in: Proceedings of the 16th USENIX Security Symposium, San Jose, California, USENIX Association, 2007, pp. 167–182.

[56] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer Networks 31 (23Ãćâ Ćňâ ĂIJ24) (1999) 2435 – 2463. doi:10.1016/S1389-1286(99)00112-7.

[57] M. Roesch, Snort - lightweight intrusion detection for networks, in: Proceedings of the 13th USENIX conference on System administration, LISA '99, USENIX Association, Berkeley, CA, USA, 1999, pp. 229–238.

[58] J. Goebel, T. Holz, Rishi: identify bot contaminated hosts by irc nickname evaluation, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 8–8.

[59] G. Gu, J. Zhang, W. Lee, BotSniffer: Detecting botnet command and control channels in network traffic, in: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), 2008, pp. 1–1.

[60] A. Ramachandran, N. Feamster, D. Dagon, Revealing botnet membership using dnsbl counter-intelligence, in: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2, SRUTI'06, USENIX Association, Berkeley, CA, USA, 2006, pp. 8–8.

[61] R. Villamarin-Salomon, J. Brustoloni, Identifying botnets using anomaly detection techniques applied to DNS traffic, in: Proceedings of 5th IEEE Consumer Communications and Networking Conference (CCNC 2008), 2008, pp. 476–481.

[62] R. Villamarín-Salomón, J. C. Brustoloni, Bayesian bot detection based on dns traffic similarity, in: Proceedings of the 2009 ACM symposium on Applied Computing, SAC '09, ACM, New York, NY, USA, 2009, pp. 2035–2041. doi:10.1145/1529282.1529734.

[63] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Y. Zhao, Online botnet detection based on incremental discrete fourier transform, JNW 5 (5) (2010) 568–576. doi:http://dx.doi.org/10.4304/jnw.5.5.568-576.

[64] T. M. Mitchell, Machine Learning, 1st Edition, McGraw-Hill, Inc., New York, NY, USA, 1997.

[65] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[66] S. Kotsiantis, I. Zaharakis, P. Pintelas, Supervised machine learning: A review of classification techniques, Frontiers in Artificial Intelligence and Applications 160 (2007) 3.

[67] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323. doi:10.1145/331499.331504.

[68] A. J. Aviv, A. Haeberlen, Challenges in experimenting with botnet detection systems, in: Proceedings of the 4th conference on Cyber security experimentation and test, CSET'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 6–6.

[69] T. Honeynet Project (Ed.), Know Your Enemy: Learning about Security Threats, 2nd Edition, Addison Wesley Publishing, 2004.

[70] N. Provos, T. Holz, Virtual honeypots: from botnet tracking to intrusion detection, 2nd Edition, Addison-Wesley Professional, 2009.

[71] C. Livadas, R. Walsh, D. Lapsley, W. Strayer, Usilng machine learning technliques to identify botnet traffic, in: Local Computer Networks, Proceedings 2006 31st IEEE Conference on, 2006, pp. 967 –974. doi:10.1109/LCN.2006.322210.

[72] W. T. Strayer, D. Lapsely, R. Walsh, C. Livadas, Botnet detection based on network behaviour, in: W. Lee, C. Wang, D. Dagon (Eds.), Botnet Detection, Vol. 36 of Advances in Information Security, Springer, 2008, pp. 1–24.

[73] G. Gu, R. Perdisci, J. Zhang, W. Lee, Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection, in: Proceedings of the 17th conference on Security symposium, 2008, pp. 139–154.

[74] H. Husna, S. Phithakkitnukoon, S. Palla, R. Dantu, Behavior analysis of spam botnets, in: Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, 2008, pp. 246 –253. doi:10.1109/COMSWA.2008.4554418.

[75] S.-K. Noh, J.-H. Oh, J.-S. Lee, B.-N. Noh, H.-C. Jeong, Detecting p2p botnets using a multi-phased flow model, in: Digital Society, 2009. ICDS '09. Third International Conference on, 2009, pp. 247 –253. doi:10.1109/ICDS.2009.37.

[76] A. Nogueira, P. Salvador, F. Blessa, A botnet detection system based on neural networks, 2010 Fifth International Conference on Digital Telecommunications 0 (2010) 57–62. doi:http://doi.ieeecomputersociety.org/10.1109/ICDT.2010.19.

[77] D. Liu, Y. Li, Y. Hu, Z. Liang, A p2p-botnet detection model and algorithms based on network streams analysis, in: Future Information Technology and Management Engineering (FITME), 2010 International Conference on, Vol. 1, 2010, pp. 55 –58. doi:10.1109/FITME.2010.5655788.

[78] W.-H. Liao, C.-C. Chang, Peer to peer botnet detection using data mining scheme, in: Internet Technology and Applications, 2010 International Conference on, 2010, pp. 1 –4. doi:10.1109/ITAPP.2010.5566407.

[79] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Data-adaptive clustering analysis for online botnet detection, in: Computational Science and Optimization (CSO), 2010 Third International Joint Conference on, Vol. 1, 2010, pp. 456 –460. doi:10.1109/CSO.2010.214.

[80] C. Langin, H. Zhou, S. Rahimi, B. Gupta, M. Zargham, M. Sayeh, A self-organizing map and its modeling for discovering malignant network traffic, in: Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on, 2009, pp. 122 –129. doi:10.1109/CICYBS.2009.4925099.

[81] H. Choi, H. Lee, Identifying botnets by capturing group activities in DNS traffic, Journal of Computer Networks 56 (2011) 20–33.

[82] F. Sanchez, Z. Duan, Y. Dong, Blocking spam by separating end-user machines from legitimate mail server machines, in: Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS '11, ACM, New York, NY, USA, 2011, pp. 116–124. doi:10.1145/2030376.2030390.

[83] F. Chen, S. Ranjan, P.-N. Tan, Detecting bots via incremental ls-svm learning with dynamic feature adaptation, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, ACM, New York, NY, USA, 2011, pp. 386–394. doi:10.1145/2020408.2020471.

[84] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, P. Hakimian, Detecting p2p botnets through network behavior

analysis and machine learning, in: Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on, 2011, pp. 174 –180. doi:10.1109/PST.2011.5971980.

[85] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, X. Luo, Detecting stealthy P2P botnets using statistical traffic fingerprints, in: 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks (DSN), Hong Kong, IEEE/IFIP, 2011, pp. 121–132.

[86] W. Lu, G. Rammidi, A. A. Ghorbani, Clustering botnet communication traffic based on n-gram feature selection, Computer Communications 34 (2011) 502–514.

[87] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, C. Kruegel, Disclosure: detecting botnet command and control servers through large-scale netflow analysis, in: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC '12, ACM, New York, NY, USA, 2012, pp. 129–138. doi:10.1145/2420950.2420969.

[88] W. Strayer, R. Walsh, C. Livadas, D. Lapsley, Detecting botnets with tight command and control, in: Local Computer Networks, Proceedings 2006 31st IEEE Conference on, 2006, pp. 195 –202. doi:10.1109/LCN.2006.322100.

[89] D. Pelleg, A. Moore, et al., X-means: Extending k-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, Vol. 1, San Francisco, 2000, pp. 727–734.

[90] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometrics and Intelligent Laboratory Systems 2 (1) (1987) 37 – 52, <ce:title>Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists</ce:title>. doi:10.1016/0169-7439(87)80084-9.

[91] S. Guha, R. Rastogi, K. Shim, Rock: A robust clustering algorithm for categorical attributes, in: In Proc.ofthe15thInt.Conf.onDataEngineering, 2000, pp. 7–7.

[92] P. Salvador, A. Nogueira, U. Franca, R. Valadas, Framework for zombie detection using neural networks, in: Internet Monitoring and Protection, 2009. ICIMP '09. Fourth International Conference on, 2009, pp. 14 –20. doi:10.1109/ICIMP.2009.10.

[93] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.

[94] H. Choi, H. Lee, H. Kim, Botgad: detecting botnets by capturing group activities in network traffic, in: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, COMSWARE '09, ACM, New York, NY, USA, 2009, pp. 2:1–2:8. doi:10.1145/1621890.1621893.

[95] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural processing letters 9 (3) (1999) 293–300.

[96] T. Zhang, R. Ramakrishnan, M. Livny, Birch: A new data clustering algorithm and its applications, Data Mining and Knowledge Discovery 1 (2) (1997) 141–182.

[97] W. Lu, M. Tavallaee, G. Rammidi, A. A. Ghorbani, Botcop: An online botnet traffic classifier, in: Proceedings of the 2009 Seventh Annual Communication Networks and Services Research Conference, CNSR '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 70–77. doi:10.1109/CNSR.2009.21.

[98] W. Lu, M. Tavallaee, A. A. Ghorbani, Automatic discovery of botnet communities on large-scale communication networks, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS '09, ACM, New York, NY, USA, 2009, pp. 1–10. doi:10.1145/1533057.1533062.

[99] Y. Zeng, On detection of current and next-generation botnets, phd thesis, Ph.D. thesis, The University of Michigan (January 2012).